

Cooperative retransmission implementation in ns 2

by
Ali Azami

Master thesis in
Information and communication technolog

University of Agder
Grimstad 25.mai 2010

Abstract

From the theoretical perspective cooperative communications are reasonably well understood. But in practical realizations of cooperative communication system are still quite limited, the articles in this special issue are devoted to implementation aspects related to cooperative retransmission implementation in ns 2.system..Cooperative retransmission in wireless networks has attracted many research attentions which is the basic idea when a receiver cannot decode a frame, the retransmission is handled not by its original source but rather by a neighbor that overheard the transmission successfully, in this situation may helper have a better channel to the destination and when it comes to this topic, three key issues need to be addressed, namely when to cooperate, whom to cooperate with and how to protect cooperative retransmission are addressed here. The fundamental of cooperative communications lie in the physical layer. However, the notation of cooperation is available in various forms at different protocol layer. To facilitate access to the physical layer information and adaptation to mobility, it is natural to introduce the notion of cooperation in the layer directly above the physical, namely, the medium access control MAC layer; we consider the case of multiple cooperating neighbors from a MAC-layer perspective and assume a receiver that can only decode one transmission at a time, while multiple retransmission by several neighbors that had heard the frame, will cause a collision, according to our protocol which we are particularly interested in MAC protocol design for cooperative communication based on 802.11 CSMA/CA, and a cooperation is initiated only if the direct transmission fails. An optimal relay node is selected in a distributed manner according to instantaneous relay channel conditions in the network,

Preface

This master thesis is a part of the fulfillment of my Master's degree in Information communication technology at Grimstad University I have been a student at Grimstad University from autumn 2005 to spring 2010. I spent the first three years studying for a Bachelor degree in Computer and network security, and the last two to earn more added mobile communication in my Master's degree. The essences of my study are computer security and mobile communication. Since more and more mobile communication and computer systems are used in our daily life, important issues are keeping these systems safe and secure and a great challenge. An effective means to achieve this is to maintain by computer simulation. Course like mobile communication and computer, networks security have provided me with a good knowledge and inspiration to explore these topics further. This project has given me more knowledge to what kind of challenges we are facing regarding mobile communication and information about some key elements to find cooperative retransmission implementation in ns-2 and have better simulation in wireless network.

Grimstad, 25. Mai 2010

Ali Azami

Acknowledgement

I would like to thank my co-supervisor and supervisor, Xin He and Professor Frank Li, for their inspiration and support throughout the entire project period. They have been of great help, and given me constructive comments and ideas. Without their help, this project would probably not have been carried out at all. I am very grateful for all the support that I have received.

Finally, I would like to thank information communication technology and all the other members of University at Grimstad; they kindly and friendly behave for their pleasant company during the entire project:

Contents:

Cooperative retransmission implementation in ns 2.....	1
Abstract.....	2
Preface.....	3
Acknowledgement	4
1. Introduction.....	9
1.1 Data link layer and physical layer.....	9
1.1.1 Data link.....	10
1.1.2 Medium Access Control	10
Reliable Data Delivery.....	10
Distributed Coordination Function	12
Point Coordination Function.....	14
1.1.3 Physical Layers	14
Original IEEE 802.11 Physical Layer.....	15
1.3 Motivation of cooperative retransmission why do we need	16
1.4 Cooperative transmission.....	16
2. IEEE 802.11 DCF scheme description	17
3. Cooperative retransmission MAC description.....	18
3.1 Review of DCF	18
3.2 System model and description	19
3.2.1 Cooperative Basic scheme	20
3.2.2 Step one, the direct transmission sequence.....	21
3.2.3 Cooperative RTS/CTS scheme	23
3.2.4 Step two, the cooperative retransmission sequence.....	24
4. Introduction to ns-2 software	24
4.1 What is NS-2 ?.....	24
4.2 NS-2 Architecture.....	25
4.2.1 Wireless node design in ns-2	25
4.2.2 Outgoing packets	25
4.2.3 Incoming packets	27
4.2.4 Propagation model in ns-2	27
5. Cooperation Implement in ns 2.....	27
5.1 finding error frames	28
5.2 Testing.....	30
6. Simulation description	31
6.1 Simulation environment description	32
6.2 Simulation 802.11a	33
6.3 Simulation 802.11p.....	36
6.4 Simulation results part	39
6.5 validation of results part.....	43
7. Conclusions.....	50

8. Appendix.....	51
Appendix A.....	51
Files in ns-2 simulation have been used	51
Appendix B.....	94
IEEE 802.11 Standard.....	94
A briefly defines key terms used in the IEEE 802.11 standard	95
IEEE 802 Services	95
9.References.....	96
10.Abbreviations.....	100

List of figures:

Figure 1.1 Relationship of layers and addresses in TCP/IP..... 9

Figure 1.2 MAC layer in IEEE 802.11 standard..... 11

Figure 1.3 IEEE 802.11 MAC timing (Basic access method) 13

Figure 1.4 IEEE 802.11 MAC timing (DCF superframe construction)..... 14

Figure 1.5 IEEE 802.11 physical layer has two sub-layers, PLCP and PMD 15

Figure 2.1 IEEE 802.11 DCF basic access scheme 17

Figure 2.2 IEEE 802.11 DCF RTS/CTS scheme..... 18

Figure 3.1 System model for cooperative retransmission..... 20

Figure 3.2 Cooperative basic access schemes..... 21

Figure 3.3 Direct transmissions between source and destination 22

Figure 3.4 Cooperative RTS/CTS schemes. 23

Figure 3.5 Cooperative re-transmission..... 24

Figure 4.1 Schematic representation of a wireless node (CMU Monarch implementation)
..... 26

Figure 5.1 Simulation with 5% error data rate (**\$err set rate_ 0.05**) 30

Figure 5.2 Simulation with 20% error data rate (**\$err set rate_ 0.20**) 31

Figure 6.1 Node constructions in TCL Script..... 32

Figure 6.2 Extension of MAC and PHY modules 33

Figure 6.3 Screen shot 802.11a simulation..... 35

Figure 6.4 Delay and throughput analysis of 802._11a simulation 36

Figure 6.5 Delay and throughput analysis of 802._11p simulation 38

Figure 6.6 Screen shot cooperative simulation 42

Figure 6.7 Delay and throughput analysis cooperative simulation..... 43

Figure 6.8 (Enhanced) Delay and throughput analysis cooperative simulation 46

Figure 6.9 delay and throughput analysis cooperative simulation with teen nodes..... 47

Figure 6.10 delay and throughput analysis cooperative simulation with teen nodes..... 48

Figure 6.11 delay and throughput analysis cooperative simulation with teen nodes..... 49

List of tables:

Table 6.1 Simulation parameters	39
Table 6.2 Parameters for IEEE 802.11	39
Table 1.1 IEEE 802.11 Standards	94
Table 1.2 A briefly defines key terms used in the IEEE 802.11 standard	95
Table 1.3 IEEE 802.11 define nine service.....	95

1. Introduction

Institute of Electrical and Electronics Engineers (IEEE) has defined the specification for a wireless LAN, called IEEE 802.11, which covers the physical and data link layers. Wireless communication is one of the fastest-growing technologies. The demand for connecting devices without the use of cables is increasing everywhere. That is present in the workplace, the home, educational institutions, cafes, airports and street corners, wireless LANs are now one of the most important access network technologies in the Internet today. Although many technologies and standards for wireless LANs were developed in the 1990s, one particular class of standards has clearly emerged as the winner the IEEE 802.11 wireless LAN, also known as Wireless Fidelity (Wi-Fi). To familiarize the reader with the necessary background, section 1.1 and 1.2 briefly introduces data link layer, physical layer

1.1 Data link layer and physical layer

TCP/IP protocol suit contain relatively independent protocols which can be mixed and matched depending on the needs of the system. Four levels of addresses in the TCP/IP protocol suite are used in an internet that consist of Physical addresses, logical addresses, port addresses, and specific addresses as shown in figure 1.1 and each address is related to a specific layer in the TCP/IP architecture [1]

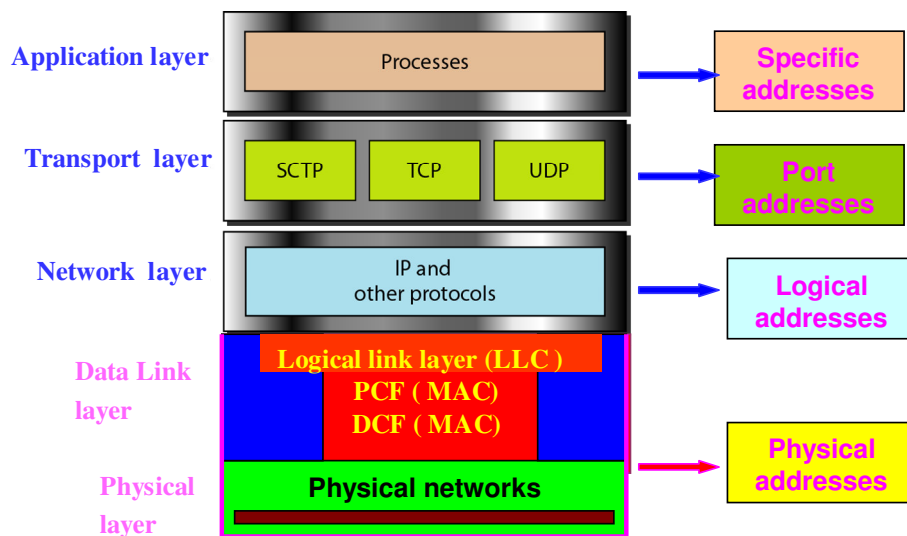


Figure 1.1 Relationship of layers and addresses in TCP/IP

1.1.1 Data link

The two main functions of the data link layer are data link control and media access control. The first, data link control, deals with the design and procedures for communication between two adjacent nodes: node-to-node communication. The second function of the data link layer is media access control, or how to share the link.[1]

We can consider the data link layer as two sub-layers that the upper sublayer call Logical link control, LLC, is responsible for data link control, and the lower sub-layer that call Media Access Control, MAC. is responsible for resolving access to the shared media. If the channel is dedicated, we do not need the lower sublayer. Figure 1.1 shows these two sub-layers in the data link layer. :[1, 2] The architecture is based on used in the IEEE has actually made this division for LANs. The upper sublayer that is responsible for flow and error control is called the logical link control (LLC) layer; the lower sublayer that is mostly responsible for multiple access resolution is called the media access control (MAC) layer. When nodes or stations are connected and uses a common link, called a multipoint, we need a multiple-access protocol to coordinate access to the link. This is similar for multipoint networks and many formal protocols have been devised to handle access to a shared link:[1, 2]

1.1.2 Medium Access Control

The IEEE 802.11 MAC layer covers three functional areas: reliable data delivery, access control, and security.[8, 9] The MAC sub-layer is responsible for the channel access procedures, protocol data unit (PDU) addressing, frame formatting, error checking, and fragmentation and reassembly of MAC Service Data Unit (MSDUs) The MSDU is the Service Data Unit that is received from the Logical Link Control sub-layer which lies above MAC as shown in figure 1.1 above sub-layer in a protocol stack. The Logical Link Control (LLC) and MAC sub-layer are collectively referred to as the Data Link Layer [2, 3, 4, 5, 7]

Reliable Data Delivery

A wireless LAN using the IEEE 802.11 physical and MAC layers is subject to considerable unreliability. Noise, interference, and other propagation effects result in the loss of a significant number of frames and a number of MAC frames may not successfully be received. This situation can be dealt with by reliability mechanisms at a higher layer, such as TCP. There is not a good option to use timers for retransmission at higher layers. It is therefore more efficient to deal with errors at the MAC level. For this purpose, IEEE 802.11 includes a frame exchange protocol in such away when a station receives a data frame from another station; it returns an acknowledgment (ACK) frame to the source station. This exchange is treated as an atomic unit, not to be interrupted by a transmission from any other station. If the source does not receive an ACK within a short period of time, either its data frame or returning ACK was damaged, the source retransmits the frame.[8, 9], and the basic data transfer mechanism in IEEE 802.11 involves an exchange of two frames. For more reliability that is possible a four-frame exchange may be used, in this situation, a source first issues a Request to Send (RTS) frame to the destination. The destination then responds with a Clear to Send (CTS). After

receiving the CTS, the source transmits the data frame, and the destination responds with an ACK. The RTS alerts all stations that are within reception range of the source that an exchange is under way; these stations hold back from transmission in order to avoid a collision between two frames transmitted at the same time. In similarly ways, the CTS alerts all stations that are within reception range of the destination that an exchange is under way. The RTS/CTS portion of the exchange is a required function of the MAC but may be disabled.[5, 6, 8, 9, 12]

Distributed access protocols, which, like Ethernet, distribute the decision to transmit over all the nodes using a carrier sense mechanism; and centralized access protocols, which involve regulation of transmission by a centralized decision maker A distributed access protocol makes sense for an ad hoc network of peer workstations and may also be attractive in other wireless LAN configurations that consist primarily of bursty traffic. A centralized access protocol is natural for configurations in which a number of wireless stations are interconnected with each other and some sort of base station that attaches to a backbone wired LAN; it is especially useful if some of the data is time sensitive or high priority.. [5, 6, 8, 9, 12]

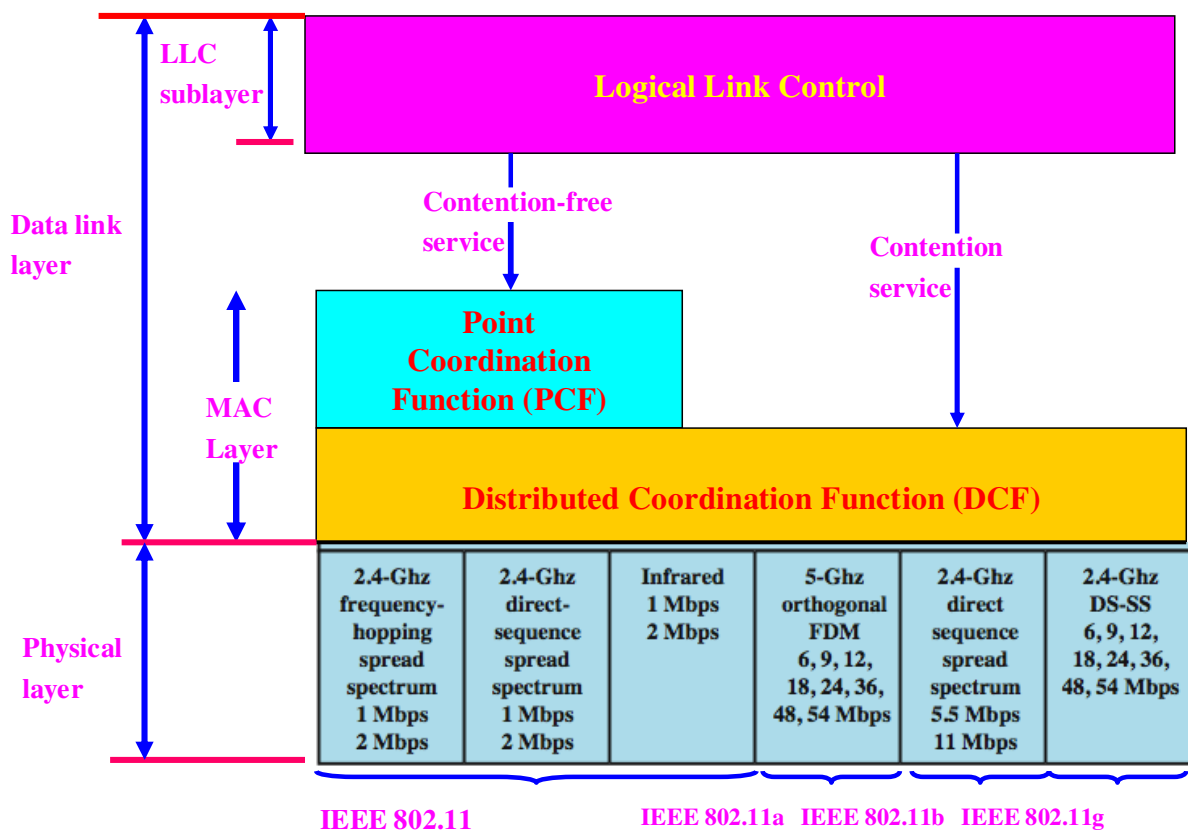


Figure 1.2 MAC layer in IEEE 802.11 standard

Figure 1.2 illustrates the MAC layer in IEEE 802.11 standard. The lower sub-layer of the MAC layer is the distributed coordination function (DCF) which uses a contention algorithm to provide access to all traffic and asynchronous traffic directly uses DCF. The

point coordination function (PCF) is a centralized MAC algorithm used to provide contention-free service. PCF is built on top of DCF and exploits features of DCF to assure access for its users. [5, 11]

Distributed Coordination Function

The DCF sub-layer makes use of a simple carrier sense multiple access (CSMA) algorithm. The stations always listen to the medium when they have a MAC frame to transmit. If the medium is idle, the station may transmit; otherwise all stations must wait until the current transmission is complete before transmitting. The distributed coordination function does not include a collision detection function for example carrier sense multiple access with collision detection (CSMA/CD) because collision detection is not practical on a wireless network. A transmitting station cannot effectively distinguish incoming weak signals from noise and the effect of its own transmission when the dynamic range of the signals on the medium is very large. DCF includes a set of delays that amounts to a priority scheme to ensure the smooth and fair function of algorithm and start by considering a single delay known as an inter-frame space (IFS). There are three different IFS values and the algorithm is best explained by initially ignoring this detail which using an IFS, the rules for CSMA access are as follows

If a station has a frame to transmit senses the medium and if the medium is idle, it waits for a short period to see if the medium remains idle for a time equal to IFS. If so, the station may transmit immediately.

If the medium is busy and the current transmission is not over or because the medium becomes busy during the IFS idle time, the station defers transmission and continues to monitor the medium until the current transmission is over.

The station must delays other IFS , if the current transmission is over, once the medium remains idle for this period, then the station backs off a random amount of time and again senses the medium. If the medium is still idle, the station may transmit. If the medium becomes busy, during the backoff time, the backoff timer is halted and resumes when the medium becomes idle.

In situation when the transmission is unsuccessful, which is determined by the absence of an acknowledgement, then it is assumed that a collision has occurred. Binary exponential backoff provides a means of handling a heavy load. Repeated failed attempts to transmit result in longer and longer backoff times, which causes to smooth out the load without such a backoff, the following situation could occur:

Collision can happened when two or more stations attempt to transmit at the same time. These stations then immediately attempt to retransmit, causing a new collision. This situation lead to coordination and the preceding scheme is refined for distributed coordinate function (DCF) to create access priority-based scheme by the simple mean of using three values for IFS

Short Interframe Space (SIFS): The shortest IFS, used for all immediate response actions, [5, 7,11]

Point Coordination Function Interframe Space (PIFS): A midlength IFS, used by the centralized controller in the PCF scheme when issuing polls [5, 7, 11]

Distributed Coordination Function Interframe Space (DIFS): The longest IFS, used as a minimum delay for asynchronous frames contending for access [5, 7, 11]

As shown in figure 1.3 IEEE 802.11 MAC timing (Basic access method) and the use of this time values. Here first consider the SIFS. Any station using SIFS to determine transmission opportunity has, in effect, the highest priority, because it will always gain access in preference to a station waiting an amount of time equal to PIFS or DIFS. The SIFS is used in the following circumstances: [5, 7, 11]

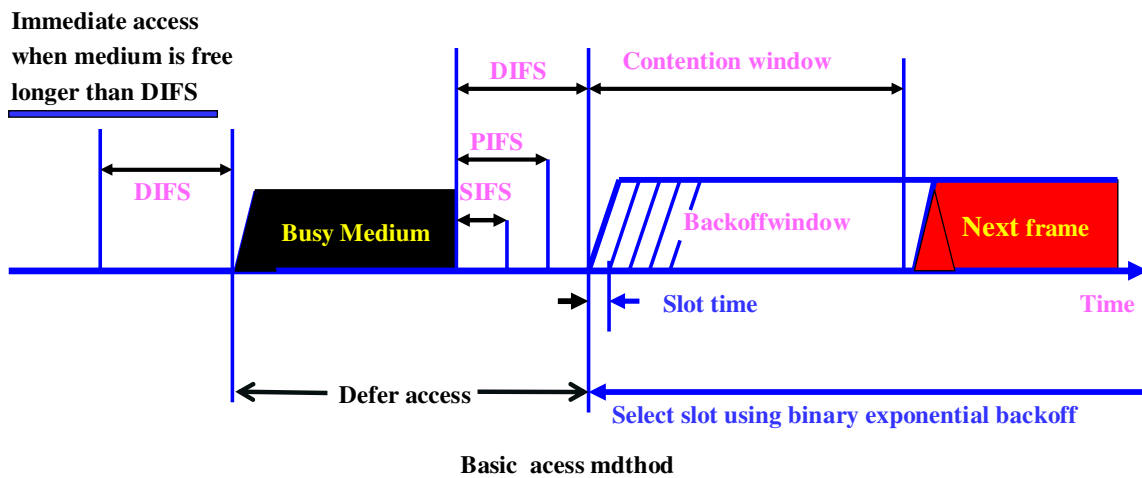


Figure 1.3 IEEE 802.11 MAC timing (Basic access method)

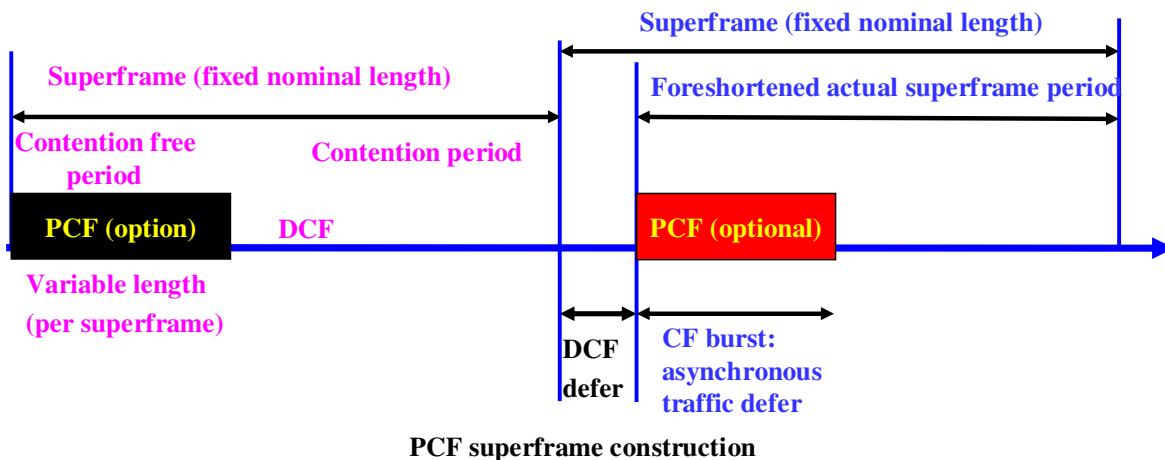


Figure 1.4 IEEE 802.11 MAC timing (DCF superframe construction)

Acknowledgment (ACK): When a station receives a frame addressed only to itself not multicast or broadcast, it responds with an ACK frame after waiting only for an SIFS gap. [5, 7, 11]

Clear to Send (CTS): A station can ensure that its data frame will get through by first issuing a small Request to Send (RTS) frame. The station to which this frame is addressed should immediately respond with a CTS frame if it is ready to receive. All other stations receive the RTS and defer using the medium. [5, 7, 11]

Poll response: The point coordinator makes use of PIFS when issuing polls.

Point Coordination Function

The point coordination function (PCF) is an optional capability that can be used to provide connection-oriented, contention-free services by enabling polled stations to transmit without contending for the channel. This is an alternative access method implemented on top of the DCF is called PCF. The operation consists of polling by the centralized polling master. The point coordinator makes use of PIFS when issuing polls. Because PIFS is smaller than DIFS, the point coordinator can seize the medium and lock out all asynchronous traffic while it issues polls and receives responses. [5, 7, 11]

At the beginning of a superframe, the point coordinator may optionally seize control and issue polls for a given period of time. This interval varies because of the variable frame size issued by responding stations. The remainder of the superframe is available for contention-based access. At the end of the superframe interval, the point coordinator contends for access to the medium using PIFS. If the medium is idle, the point coordinator gains immediate access and a full superframe period follows. [5, 8]

The PCF function is performed by the point coordinator (PC) in the AP within a BSS. Stations within the BSS that are capable of operating in the contention-free poll (CFP) are known as CF-aware stations. The method by which polling tables are maintained and the polling sequence is determined by the PC is left to the implementor. [7, 8]

1.1.3 Physical Layers

The IEEE 802.11 LAN has several physical layers defined to operate with its MAC layer. Each physical layer is divided into two sub-layers that correspond to two protocol function as shown in figure 1.5. The physical layer convergence procedure (PLCP) is the upper sub-layer, and it provides a convergence function that maps the MPDU into a format suitable for transmission and reception over a given physical medium. The physical medium dependent (PMD) sub-layer is concerned with the characteristics and methods for transmitting over the wireless medium. [7, 8]

Figure 1.5 shows that MPDU is mapped into PLCP frame that consists of three parts.

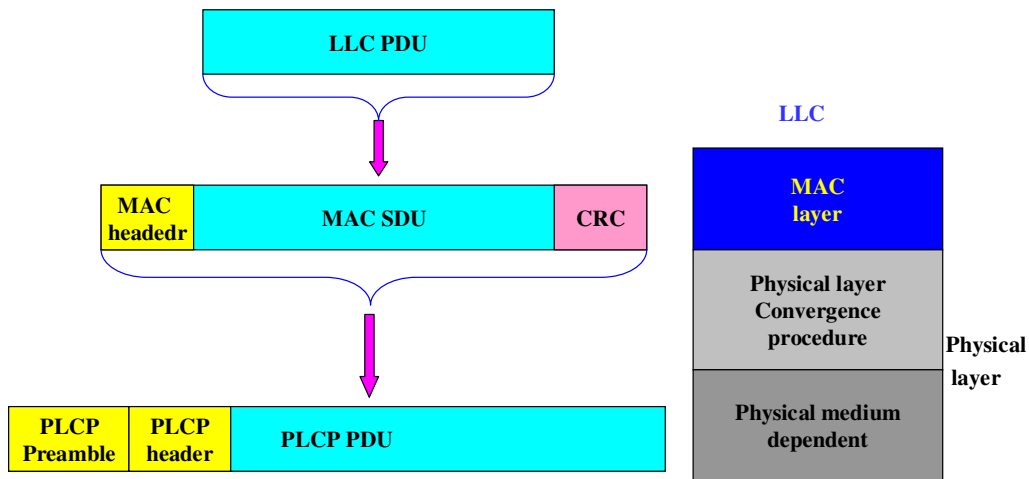


Figure 1.5 IEEE 802.11 physical layer has two sub-layers, PLCP and PMD

The specific structure of each PLCP depends on the particular physical layer definition. There are three original physical layers that have been defined for IEEE 802.11.[7, 8, 9]

Original IEEE 802.11 Physical Layer

In the original 802.11 physical layer standard, three physical media are defined which the first uses direct sequence spread spectrum (DSSS) operating in the 2.4-GHz ISM band, at data rates of 1 Mbps and 2 Mbps. In some countries requires no licensing for the use of this band for example in the United States. And numbers of channel available depend on the bandwidth allocated by the various national regulatory agencies. In the DSSS scheme up to three non-overlapping channels and each with a data rate of 1 Mbps or 2 Mbps. can be used and each channel has a bandwidth of 5 MHz. The encoding scheme that is used is Differential Binary Phase Shift Keying (DBPSK) for the 1-Mbps rate and DQPSK for the 2-Mbps rate.[9, 10, 13]

Frequency-hopping spread spectrum (FHSS) is the second alternative and operating in the 2.4-GHz ISM band, at data rates of 1 Mbps and 2 Mbps. FHSS system makes use of a multiple channels, with the signal hopping from one channel to another based on a pseudonoise sequence. In the case of the IEEE 802.11 scheme, 1-MHz channels are used..[9, 10, 13]

Infrared is the third alternative at 1 Mbps and 2 Mbps operating at a wavelength between 850 and 950 nm. The IEEE 802.11 infrared scheme is omnidirectional rather than point to point and a range of up to 20 m is possible. The modulation scheme for the 1-Mbps data rate is known as 16-PPM Pulse Position Modulation..[9, 13]

1.3 Motivation of cooperative retransmission why do we need

Today cooperative communication is an active area of research, and wireless devices are evolving into multipurpose systems with data extensive applications running on them, by this such applications require strong error protection and high speed connectivity. Those requirements, and along with the exploding growth of wireless networks and limited spectrum resources, have created capacity crunch and high interference in today's wireless networks. This causes in situation that a move towards the development of new wireless techniques which can achieve a more efficient use of the available spectrum or energy consumption when emerging techniques such as Multi-Input Multi-Output systems increase the spectrum efficiency in terms of the number of bits per hertz of bandwidth, because of the size, cost and power constraints posed by portable wireless devices its usage is limited. An alternative approach called cooperative communications promises to deliver some of the benefits of multi-input multi-output within the given constraints. Cooperative communication refers to the collaborative processing and retransmission of the overheard information at those stations surrounding the source. Because of the cooperation takes full advantage of the nature of the wireless channel and create diversity, in particular transmission diversity, thereby achieving tremendous improvements in system robustness, capacity, delay, interference, and coverage range. However the notion of cooperation is available in various forms at different protocol layers and the fundamentals of cooperative communications lie in the physical layers to facilitate access to the physical layer information and adaptation to mobility, it is natural to introduce the notion of cooperation in the layer directly above the physical layer, namely the medium access control layer, the benefit of cooperative communication can be exploited to the maximum and evaluating the performance these protocols have also become extremely important . Since two software's simulation can give us to different simulation results and it is difficult to conduct physical experiments on the performance of these systems such as wireless networks. Here network simulators, such as ns-2 have been used to evaluate the performance of network protocols and some modification will taken place, however, the network simulators is employed in the research community in many years:

1.4 Cooperative transmission

In wireless networks, cooperative transmission is used as a means to combat channel fading. In cooperative communications, relays are assigned to help a source node to deliver its information to its destination node. In addition in wireless networks the transmitted signals that can be overheard by some nodes which do nothing but wait for their own transmission during other nodes transmission in the network. Recently cooperative communication schemes have been introduced to exploit the broadcasting nature of wireless transmission and many research on cooperative communication techniques is being developed to allow stations to cooperate in their transmissions in order to improve the communication between all network and since transmission in the wireless channel is overheard by neighboring station which can process these signals and retransmit them in order to facilitate better reception.. In these schemes, some overhearing nodes are also involved in the signal transmission by relaying the received signal from the source to the destination. Cooperative communications can

achieve spatial diversity because signals bearing the same information go through uncorrelated channels introduced by cooperating nodes.

Cooperative transmission can obtain spatial diversity without using multiple antennas, thus achieving more reliable transmission or consuming less power

When cooperative diversity is utilized from the Medium Access Control (MAC) layer in distributed wireless networks, three key issues need to be addressed, namely when to cooperate, whom to cooperate with and how to protect cooperative transmissions.[14]

Since wireless channels vary with time, a source node may not always need help from a relay node, Therefore it is more sensible that cooperation is only initiated when it is necessary and beneficial and secondly, one or more appropriate relay nodes need to be selected among multiple potential helpers in the network [14]

A cooperative MAC protocol named CoopMAC is developed for wireless local area networks (WLAN) [14, 64]. Their protocol utilizes the possible opportunities where a two hop relaying link may support higher data rate than the direct link.

2. IEEE 802.11 DCF scheme description

The IEEE 802.11 standard for wireless networks incorporates two medium access methods, which one of two is mandatory called, Distributed Coordination Functions (DCF) method and the other is optional, Point Coordination Functions (PCF) method. Wireless stations communicate with each other using a contention based channel access method CSMA/CA in the DCF. On the top of DCF is built PCF, which is only possible in infrastructure mode. The access-point of cell acts as a coordinator called the point coordinator (PC) for that cell. The PC grants contention free channel access to individual nodes by polling them for transmission.

The DCF is an asynchronous data transmission function and best suits delay insensitive traffic and PCF best suits delay sensitive traffic. Basically the DCF provides an equal opportunity to access the shared wireless channel to the contending stations, this equal access opportunity results in an equal share of the bandwidth or throughput

The DCF defines two techniques for packet transmissions. The default is a two-way handshaking mechanism called basic access method as shown in figure 2.1

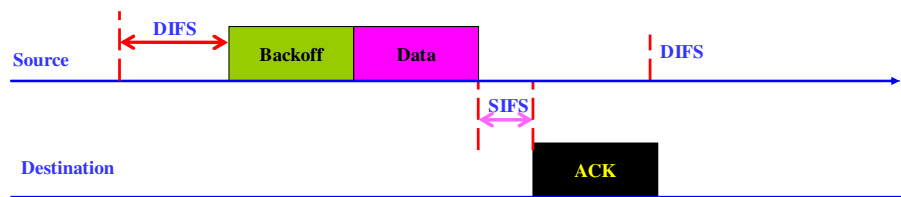


Figure 2.1 IEEE 802.11 DCF basic access scheme

Another optional technique is a mechanism known as a Request To Send / Clear To Send (RTS/CTS) method as shown in figure 2.2. The use of the RTS and CTS frames can improve performance in two important ways. The hidden station problem is mitigated since a long DATA frame is transmitted only after the channel has been reserved. Because the RTS and CTS frames are short, a collision involving an RTS or CTS frame will last only for the duration of the short RTS or CTS frame. Once the RTS and CTS frames are correctly transmitted, the following DATA and ACK frame should be transmitted without collision.

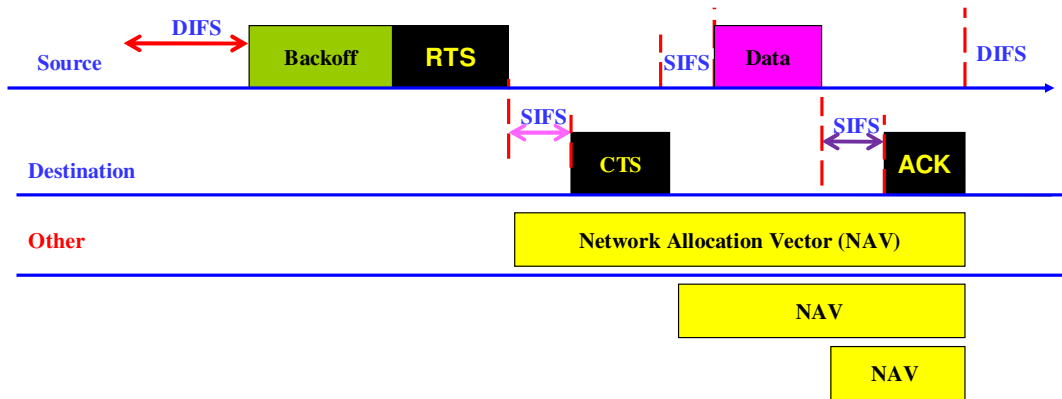


Figure 2.2 IEEE 802.11 DCF RTS/CTS scheme

Although the RTS/CTS exchange can help reduce collisions, it also introduces delay and consumes channel resources. For this reason, the RTS/CTS exchange is only used if at all to reserve the channel for the transmission of a long DATA frame. In practice, each wireless station can set an RTS threshold such that the RTS/CTS sequence is used only when the frame is longer than the threshold. For many wireless stations, the default RTS threshold value is larger than the maximum frame length, so the RTS/CTS sequence is skipped for all DATA frames sent.

3. Cooperative retransmission MAC description

IEEE 802.11 WLANs provide flexible access to the Internet; have been widely researched and deployed for several years. Medium access control (MAC) protocol is one of the key components in WLAN, which employs distributed coordination function (DCF) as the primary mechanism to access the medium. Distributed coordinate function is based on the carrier sense multiple access with collision avoidance (CSMA/CA) protocol that use a binary exponential backoff (BEB) algorithm to provide access to heterogeneous traffic.

3.1 Review of DCF

We briefly here review the process of the basic access mode of DCF. If a station has a packet to send, monitors the channel activity. If the channel is idle for a period of time equals to a distributed inter-frame space (DIFS), the station transmits. Otherwise, the

station defers and monitors the channel until it is measured idle for a DIFS. At this point, the station generates a random backoff interval before transmitting; this method causes to minimize the probability of collision with packets being transmitted by other stations. Distributed coordinate function use an exponential backoff algorithm and the backoff time that is uniformly chosen which in the range $(0, CW-1)$, here the CW is called contention window CW which is set equal to CW_{min} known as minimum contention window at the first transmission. After each unsuccessful transmission, CW will be doubled until it reaches the maximum value $CW_{max} = (2^m) W$, where W equals to CW_{min} , the backoff counter is decremented as long as channel is sensed idel and frozen when a transmission is detected o n the channel and the station transmits when the backoff counter reaches zero.. In RTS/CTS based access mode, nodes transmit data using special short RTS and CTS frames prior to the transmission of the data packet in favor of shortening the collision time interval. After CTS is correctly received, the source node is allowed to transmit packet. That is still possible a collision may occur within two or more short RTS/CTS, compared to the in case where large data packet may collide and more bandwidth is wasted, but in situation the RTS/CTS will be less bandwidth wasted, if collision happen.

3.2 System model and description

In figure 3.1 shown initial network system in nam that start at random position and will be illustrate how the proposed cooperative scheme works. In below you can see a screenshot of a nam window where the most important nodes source (S) and destination (D) are labeled in addition the system network consist of several arbitrarily potential helper nodes, 1, 2, 3,,7 . In the first case each packet with starts from source S to the intended destination D and other nodes in the network can hear from the source node and destination and will as members of relay nodes or communicate with each other.

In the model, we assume the wireless channels to be strongly correlated in the time domain but independent in the spatial domain. That is to say, the direct link is highly temporally correlated and the time diversity is limited when Automatic-Repeat-Request (ARQ) is executed on the same channel; but the channels between S and D and the channels between each relay candidate and D are assumed to be independent of each other, hence the full spatial diversity can be achieved by data retransmission on another channel.[14]

The cooperative retransmission is only initiated when the direct transmission fails. The relay candidates will contend for channel access if they have corrected decoded the data packets they have captured from the direct link and the relay selection criterion is satisfied. An optimal relay node will be automatically selected to forward the data packet to the destination according to our proposed scheme [14]

In addition our model can be extended to multi-hop scenarios, and can be establish different link between the potential relays nodes as a new virtual single hop (e, g, direct

link between nodes three and four). In other word the nodes in our model are not stationary,

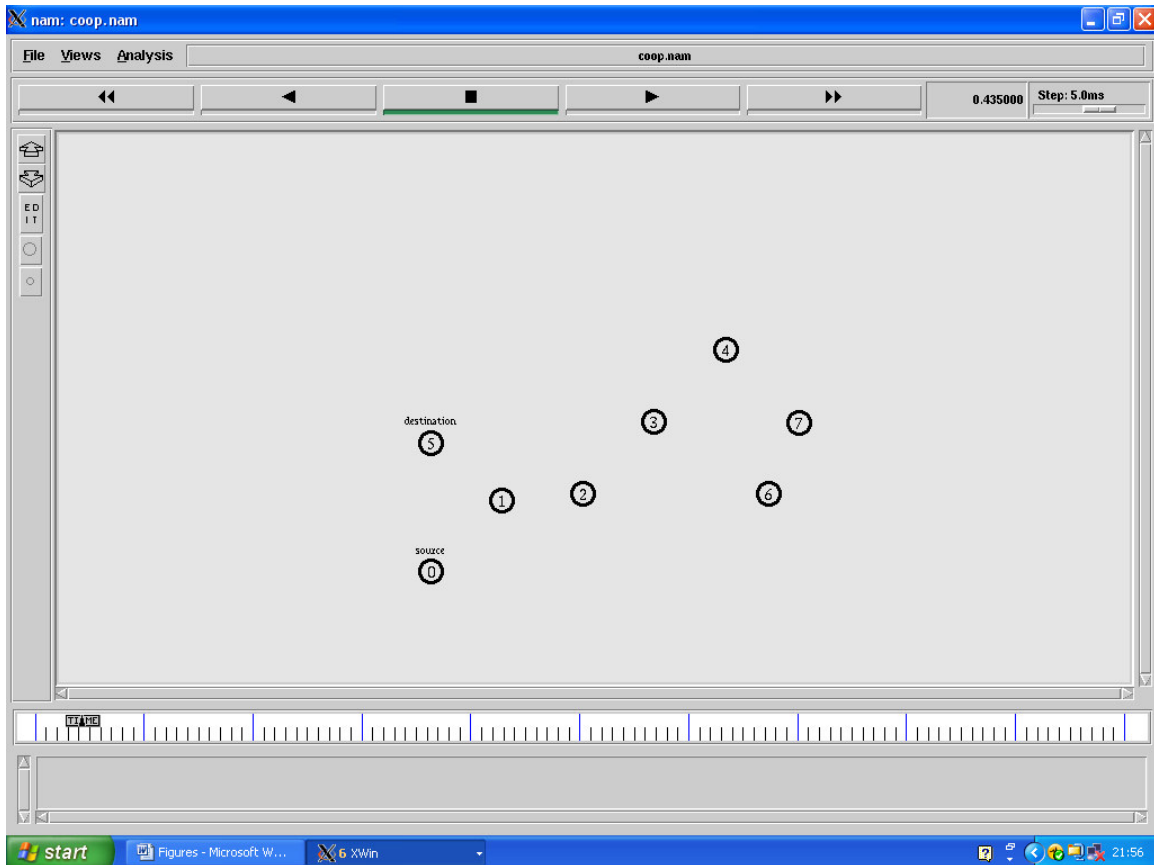


Figure 3.1 System model for cooperative retransmission

3.2.1 Cooperative Basic scheme

In figure 3.2 cooperative basic schemes are illustrated and for packet transmission there are there cases that may occur:

- (a) First step, direct transmission attempt between Source and destination is successfully
- (b) Second step, cooperative transmission attempt from Rely node to D is successfully
- (c) Third step, transmission failure

In the first step source sends out its data packet to destination according to the original basic access scheme in 802.11. As shown in figure 3.2 (a) when the transmission succeeds, the message sequence will proceed exactly the same as the original scheme

In the second cause relay node will automatically forward its received data packet, without waiting for DIFS in this situation relay node creates ACK timeout.. In figure 3.2 (b), we can see a short frame (CAV) add to the relay packet that is called Cooperative Allocation Vector which protect the ongoing cooperative retransmission sequences In order to guarantee a reliable cooperative transmission through relay node an ACK will be relayed to source since the direct transmission from source to destination was not successful, that is likely an ACK frame would also fail to reach source for this reason two step ACK is designed in this scheme. In figure 3.2 (c) , if even the cooperative retransmission fails, source has to wait for a longer ACK timeout which is $2*(SIFS+T_{ACK})$ to initiate the next transmission [14, 16]

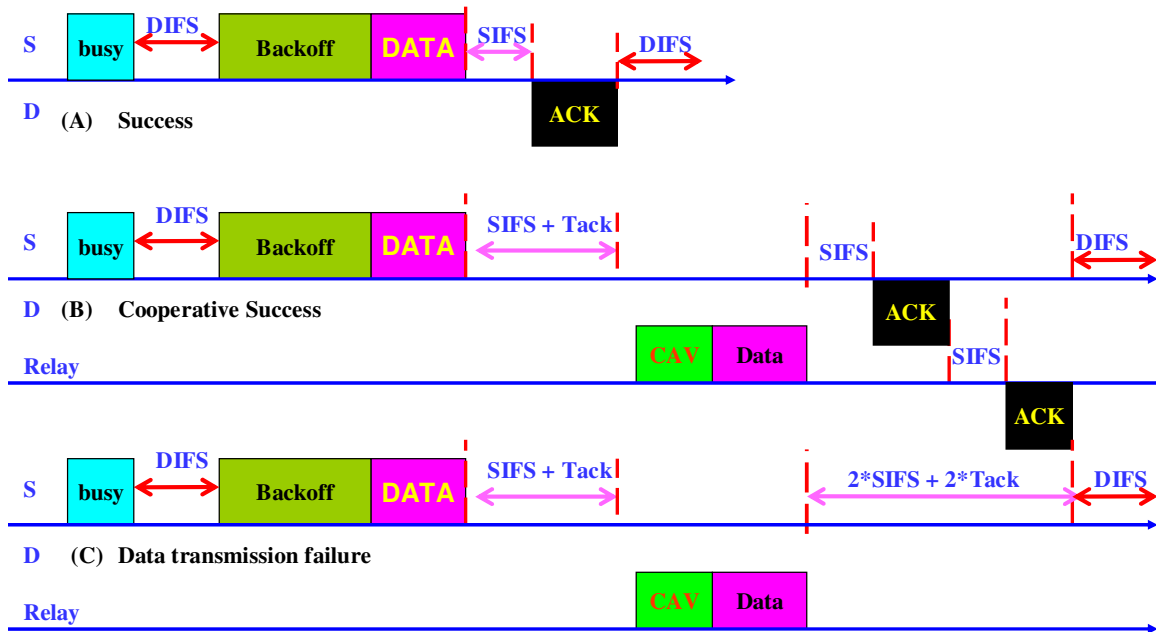


Figure 3.2 Cooperative basic access schemes.

A Cooperative Allocation Vector (CAV) packet that is introduced in above scheme reserve the channel for the following cooperative retransmission sequences and information in the cooperative allocation vector packet can be used by other nodes to set their network allocation vector values. The cooperative allocation vector contains information about the length of the time, duration from the beginning of the relay packet to the end of the second ACK packet. The CAV frame has the same frame format as RTS's, which is not used in the basic access scheme. It is transmitted at a data rate from the basic data set in order to protect the whole transmission packet exchange over a larger area. As shown in figure 3.2 all ongoing message are well protected and can not be interrupted by nodes.[14, 16]

3.2.2 Step one, the direct transmission sequence

In figure 3.2 below you can see a nam window starts direct transmission from source to destination D as the first step, node S sends out its data packet to D and here according to

Media access control protocol, DCF using RTS/CTS (IEEE 802.11 DCF RTS/CTS).. Furthermore, the RTS and CTS frames carry information about the length of the current frame exchange any station in the domain can read this information and then update Network Allocation Vector (NAV) These information contains the time duration during the channel remains busy and can mitigates hidden problem by detecting one of the RTS/CTS frames, in addition it can avoid collision by delay the transmission, another important RTS / CTS fames function is for later potential cooperative transmission and the candidate relay nodes that have received both the RTS frame and the CTS frame can decode the DATA packet from the source. When DATA packet is correctly decoded and condition for the relay selection is satisfied, the relay node contends to forward their packet to destination when that is necessary. The message procedure will proceed exactly in the same way as the original DCF scheme, when the initial transmission succeeds. Otherwise the scheme will move to the cooperative retransmission term.

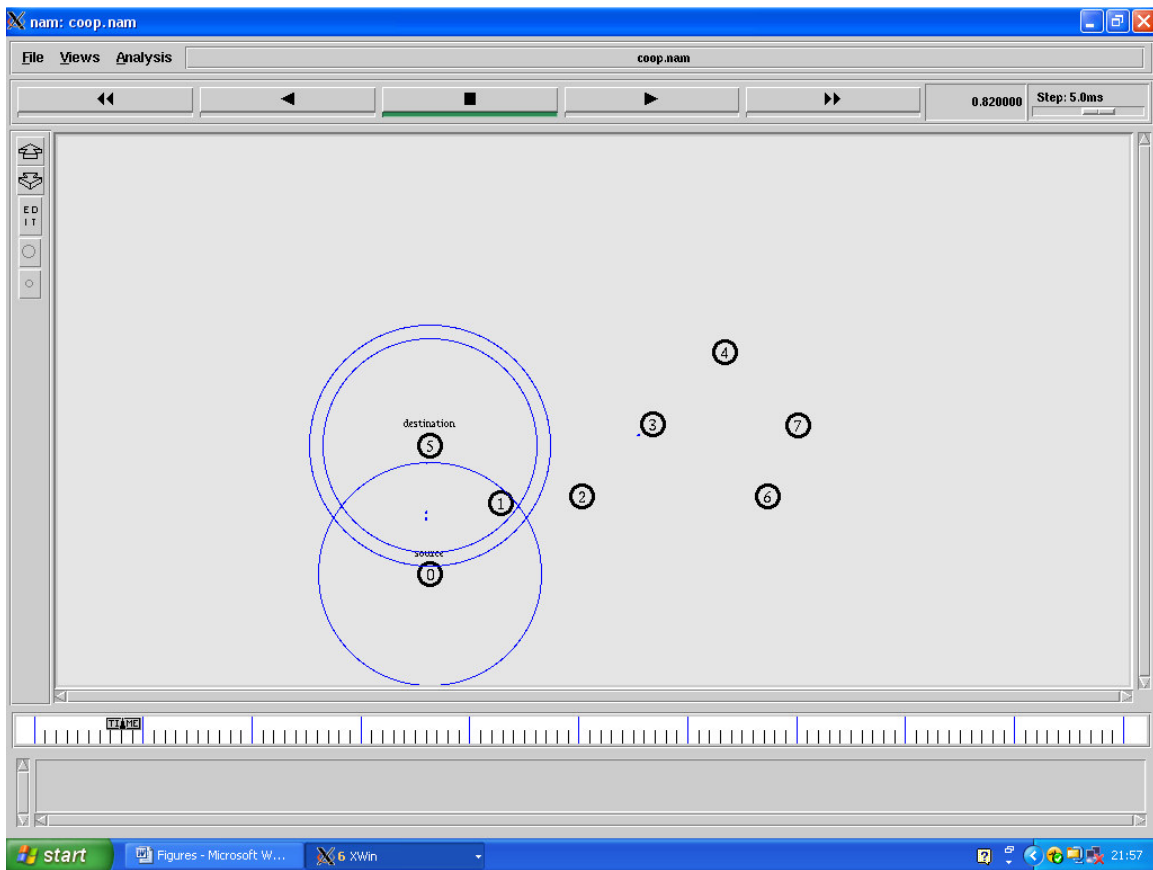


Figure 3.3 Direct transmissions between source and destination

3.2.3 Cooperative RTS/CTS scheme

Figure 3.4 solutions for the RTS/CTS scheme. In RTS/CTS transmission, source and destination perform two handshakes. If the direct transmission fails, the relay node will start cooperative retransmission automatically as shown in step (B) in figure 3.4. To deal with the hidden terminal problem, another control frame named Clear for Relay to Send (CRS) is introduced in the cooperative RTS/CTS scheme. The cooperative allocation vector (CAV) and clear for relay to send (CRS) frame contain the time duration that will be consumed by their cooperative retransmission sequences respectively to reserve the channel between relay and destination that have the same format as RTS and CTS frames and will be transmitted at the same rate as the RTS and CTS frames respectively, in order to protect the cooperative transmission packet exchange in the sensing range [14, 16]

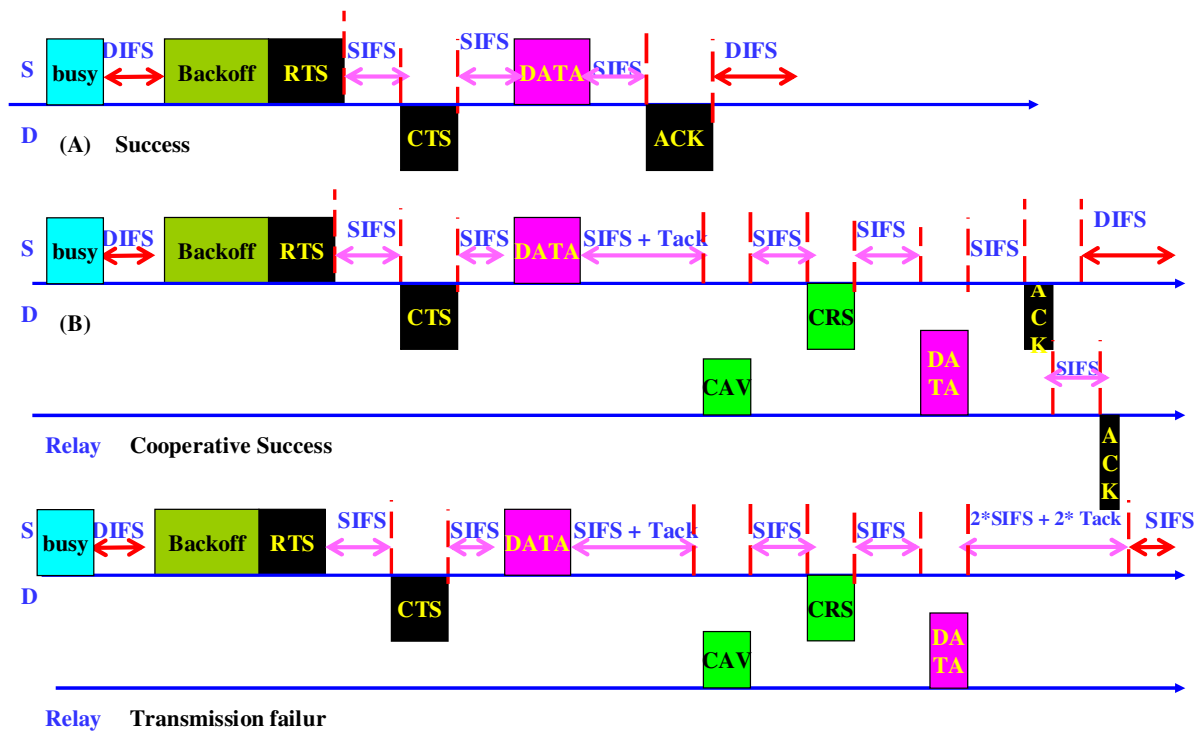


Figure 3.4 Cooperative RTS/CTS schemes.

.It might be considered to send another CRS from the source node to reserve the channel for the second ACK sent from the relay node. However, the assumption in [14, 16] model implies that the relay node is generally close to the source node and their sensing ranges could overlap to a large extent. Hence the CRS frame from the source node is no longer necessary.[14, 16]

3.2.4 Step two, the cooperative retransmission sequence

In figure 3.4 below, the cooperation is initiated when the potential cooperative link can satisfy the target data rate which the direct link cannot satisfy. Therefore, it will be automatically

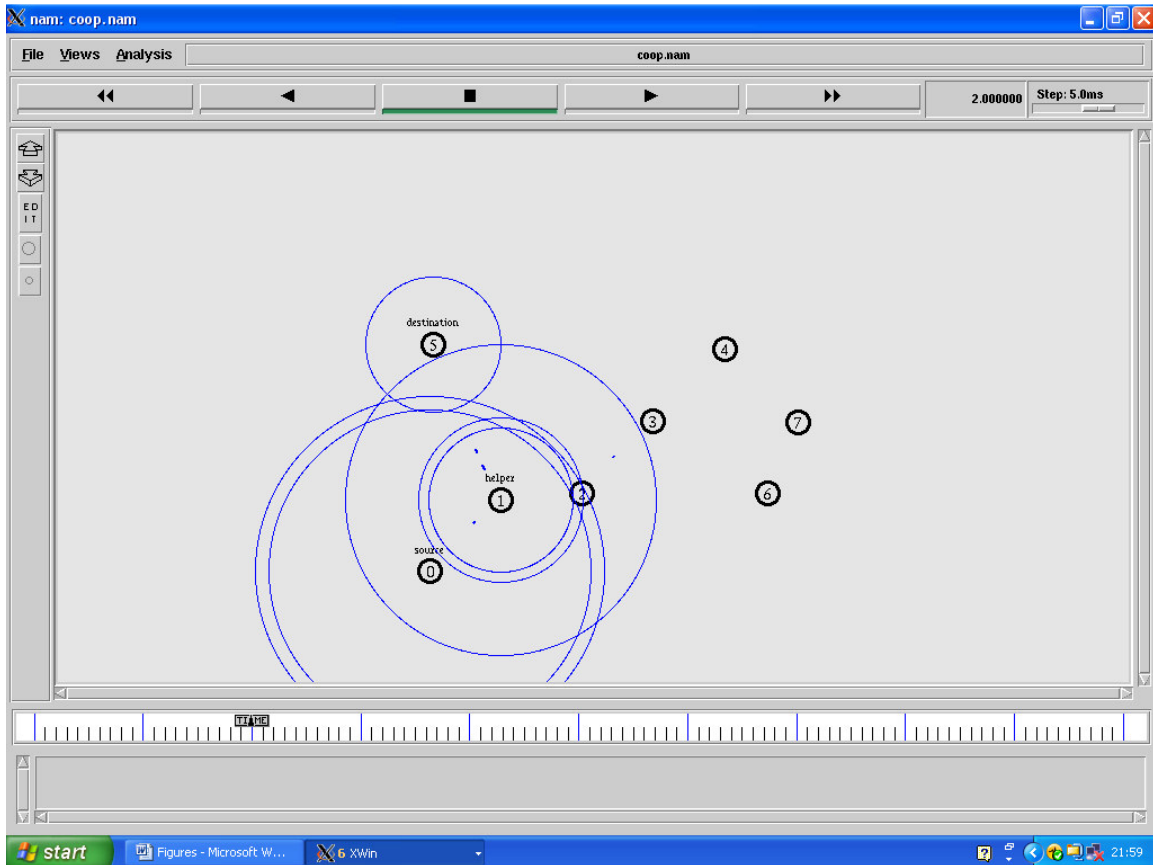


Figure 3.5 Cooperative re-transmission

4. Introduction to ns-2 software

4.1 What is NS-2 ?

Network Simulator Version 2, also known as NS-2 which is an event driven packet level network simulator developed as part of the project Virtual Internet Testbed (VINT), that was a collaboration of many institutes including UC Berkeley, AT&T, XEROX PARC and ETH. Version 1 of NS was developed in 1995 the Defense Advanced Research Project Agency (DARPA) supported NS through the Virtual Inter-Network Testbed project, and with version 2 released in 1996. Version 2 included a scripting language

called Object-oriented Tcl (OTcl).. It is an open source software package available for both Windows 32 and Linux platforms. [15]. Currently the National Science Foundation (NSF) has joined the ride in development and a group of research developers in the community are constantly working to keep NS 2 strong and can be use for many different purposes [15]

4.2 NS-2 Aechitecture

The main objective of this section is to provide the readers with insights into the NS2 architecture and here is a brief introduction to the network simulator ns-2 is a popular and widely used simulator for wireless and wired network simulation. It supports various protocols at the application layer, and mainly TCP and UDP at the transport layer and includes models for simulation of wired and wireless physical layers. Network simulation 2 offers an implementation for the IEEE 802.11 protocol, regarding its wireless physical layer implementation. Research in the field of wireless communication and network today using ns 2 to evaluate and understand the performance of new protocol and new wireless physical layer, that is obviously very important for many research group using simulation tools such as ns-2 and since that is an open software and can be expanded indeed when evaluating the performance of wireless protocols on a complex topology such as 802.11 in a multi-hop scenario, our interest lies in cooperative MAC design in wireless networks and retransmission implementation in ns 2 physical layer makes use of radio-propagation model, Propagation/ TwoRayGround

4.2.1 Wireless node design in ns-2

The wireless networking support in ns-2, originally developed as extentions to ns-2 by the CMU Monarch group and each wireless node basically is an independent entity that is responsible for computing its own position and velocity as a function of time, this wireless node consists of series of component and follow the ISO network stack, The network layer is implemented as a routing agent. All these entities are linked up together inside the wireless node ant the link-layer includes medium access control (MAC) protocols needed in such environment, and in addition address resolution protocol (ARP) for the MAC address translation in any outgoing IP packet. The physical layer includes radio propagation models, radio interfaces with adjustable parameters such as transmission power and receiver sensitivity and antennas models Figure 4.1 shows the basic schematic layout of a typical wireless node [17]

4.2.2 Outgoing packets

A source agent attached to a wireless node, when packets generated by a source are sent to the node's entry point, If packets generated by the routing agent such as routing request, which are directly passed down to the link layer which address classifier and compares the destination address with the local address, if it does not match, forwards the packet to the routing agent by default.

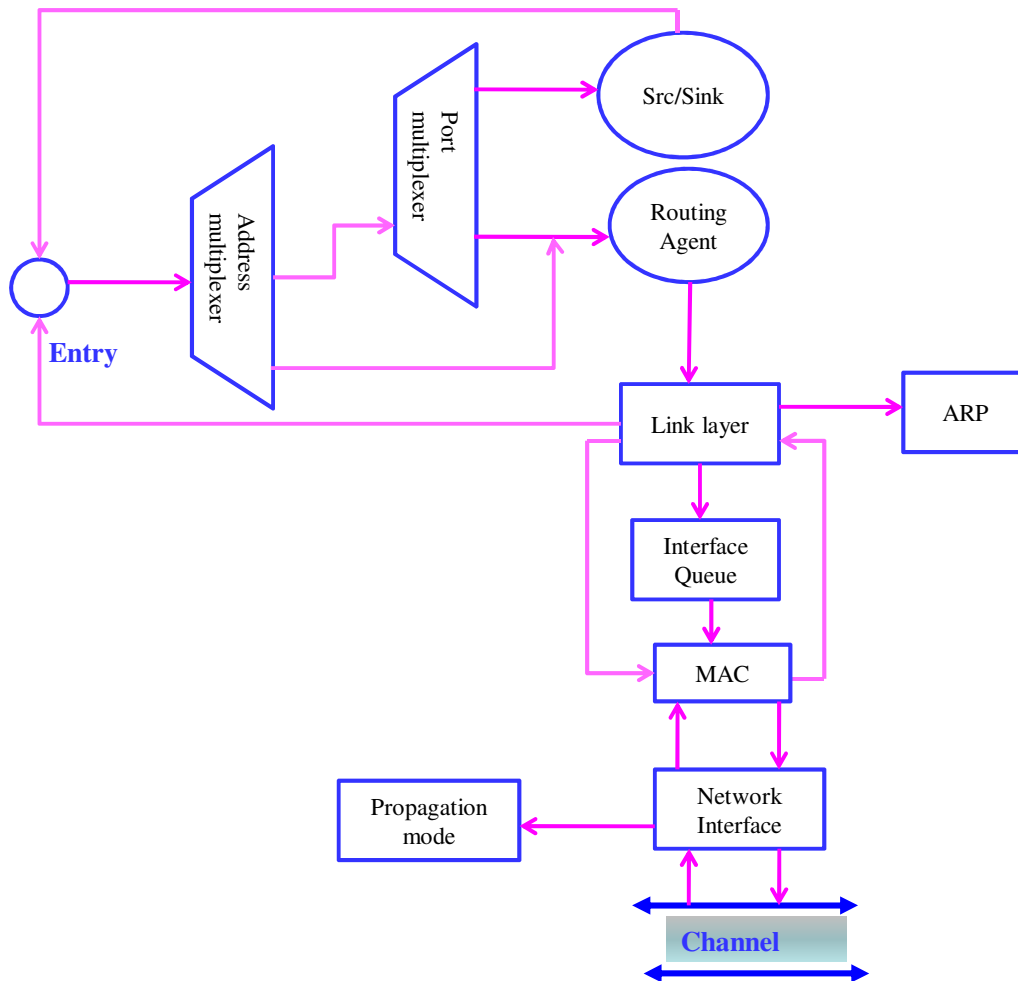


Figure 4.1 Schematic representation of a wireless node (CMU Monarch implementation)

The task for routing agent is for setting the next hop address before handling the packet down to the link layer. The link layer queries for ARP to map the IP address to a MAC address. When ARP does not have a valid mapping then it sends an ARP request to resolve the IP address to the MAC address then the packet is queued in the interface queue (IFq) when the MAC address of the next hop is known. The MAC object takes packets from the queue and sends them when appropriate. At the physical layer the node position and transmission power are recorded in a metadata packet header. This is called the common header in ns-2 terminology, before the packet is transmitted onto the wireless channel. After a propagation delay depending on the distance between the sending and receiving nodes a copy of the packet is distributed to all the other interfaces on the same channel .[18, 19, 20]

In a TCL application file, users can configure the mobility scenario for each ordinary wireless node. The node's mobility scenario is specified by assigning the node's initial location together with the moving speed and moving direction change covering the whole simulated time span. To ensure all "interface nodes" are always at the same place, it can be configured one by one with the same mobility scenario. The way in configuring an "interface node" is the same as that in configuring an ordinary node.

4.2.3 Incoming packets

A propagation model is used to determine the power at which the interface is receiving the packet. While transmitting or receiving a packet a single signal strength value is computed. The physical layer object determines then whether the packet has actually been successfully received by this node or not. Every incoming packet received at the MAC layer without error or collision on the medium is forwarded to the node's entry point. If the packet is at its final destination, the address classifier passes it directly to the port classifier which delivers it to the proper sink agent or the packet is to be forwarded by the routing agent in other case

4.2.4 Propagation model in ns-2

The Propagation Model implementation in NS-2 provides three basic RF signal attenuation formulas: two-ray ground, free space and shadowing. Two-ray ground is the most commonly used for calculating received power of a frame. This calculation is usually done in a deterministic manner. That is, with a particular transmission power, there is only one possible received power value at any particular distance. The result of using Propagation Model this way is that reception and interference ranges are perfect circles. Rayleigh fading is available, and if used, makes received power value non-deterministic.

5. Cooperation Implement in ns 2

NS-2 is a widely used simulator by wireless communication researchers and the accuracy and precision of ns-2 is critical for researchers of wireless networks. If ns-2 lacks accuracy or precision, many of the proposed theories, protocols, and may not be properly supported by the simulation results and thus not be substantive. However, the wireless simulation core in NS-2 has remained largely unchanged till now,

Our objective for the present work is to be able to simulate a medium access control (MAC) protocol for wireless networks and retransmission implementation in ns-2. Therefore we need a little modification into the some files such as (channel.c., wireless-phy.cc, , wireless-phy.h, mac802_11.cc and Packet.h and make one file co.tcl for details refer to Appendix A) that can be employed into the wireless physical layer in ns-2., support for bit error rate (BER) and packet error rate (PER) computation, and support for cumulative interference.

5.1 finding error frames

In the ns-2 implementation, if the receiving power of a frame is between the receiving and carrier sense thresholds (i.e., RXThresh_ and CXThresh_ respectively in the implementation) since the distance of the sender is between the transmission and carrier sense ranges, the error flag of the frame will be marked and consequently, the frame will be treated as a bit-error frame and the station that receive the frame will defer its transmission by EIFS(Extended Inter Frame Space) to provide enough time for other stations to notice the transmission error , the current ns-2 implementation of the wireless two-ray ground propagation model decides that a frame transmitted from a station between the transmission and carrier sense ranges of the RX node is an error frame. This is not correct assumption, and causes an RX node receives many error frames in simulation. However an EIFS will cause the RX node to have lower transmission opportunity than other wireless stations because the duration of the EIFS for example SIFSTime + 8xACKSize + aPreambleLength + aPLC-PHeaderLength + DIFS, roughly 364us) is longer than the Distributed Inter Frame Space DIFS of 50us [28].

In ns-2 implementations, the unit of error can be specified in terms of frames or bits, or it can be time-based, to support a wide variety of models. The class ErrorModel is inherited from the class Connector; the receiving agents will know the frame is an error frame and handle it accordingly. The ErrorModel also defines an additional TCL method unit to specify the unit of error as well as ranvar to specify the random variable for generating error. Detailed implementations can be found in errormodel.cc and errormodel.h while a simple example of usage in TCL is shown below (blue color)

That is (co.tcl) file for simulation,

```
# =====
# Define options
# =====
set val(chan)    Channel/WirelessChannel ;# channel type
set val(prop)    Propagation/TwoRayGround ;# radio-propagation model
set val(ant)     Antenna/OmniAntenna    ;# Antenna type
set val(ll)      LL                      ;# Link layer type
set val(ifq)     Queue/DropTail/PriQueue ;# Interface queue type
set val(ifqlen)  50                      ;# max packet in ifq
set val(netif)   Phy/WirelessPhy        ;# network interface type
set val(mac)     Mac/802_11             ;# MAC type
set val(nn)      8                      ;# number of mobilenodes
set val(rp)      AODV                   ;# routing protocol
set val(x)       650
set val(y)       650
```

#To create a simulator object. This can be done with the command:

```
set ns [new Simulator]
```

```
set f [open coop.tr w]
```

```
$ns trace-all $f
```

```
set namtrace [open coop.nam w]
```

```
$ns namtrace-all-wireless $namtrace $val(x) $val(y)
```

```
.....
```

```
set chan_8 [new $val(chan)]
```

```
# CONFIGURE AND CREATE NODES
```

```
proc UniformErr {} {  
set err [new ErrorModel]  
$err unit packet  
$err set rate_ 0.05  
$err ranvar [new RandomVariable/Uniform]  
$err drop-target [new Agent/Null]  
return $err}
```

```
$ns node-config -adhocRouting $val(rp) \  
  -llType $val(ll) \  
    -macType $val(mac) \  
    -ifqType $val(ifq) \  
    -ifqLen $val(ifqlen) \  
    -antType $val(ant) \  
    -propType $val(prop) \  
    -phyType $val(netif) \  
    #-channelType $val(chan) \  
    -topoInstance $topo \  
    -agentTrace OFF \  
    -routerTrace ON \  
    -macTrace ON \  
    -movementTrace OFF \  
    -channel $chan_1 \  
    -IncomingErrProc UniformErr
```

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

5.2 Testing

We conduct simulations to show how the impacts the result of simulation of wireless networks to consider the network configuration shown in figure 3.3 and figure 3.5 in which we employed 5% error data bits and other for 20% error data bit which the result, respectively, shown in figure 5.1 and 5.2 This scenario, which is common in wireless networks, result shown in figures below (5.1 and 5.2) in terms of `co_packet_lost.data`:

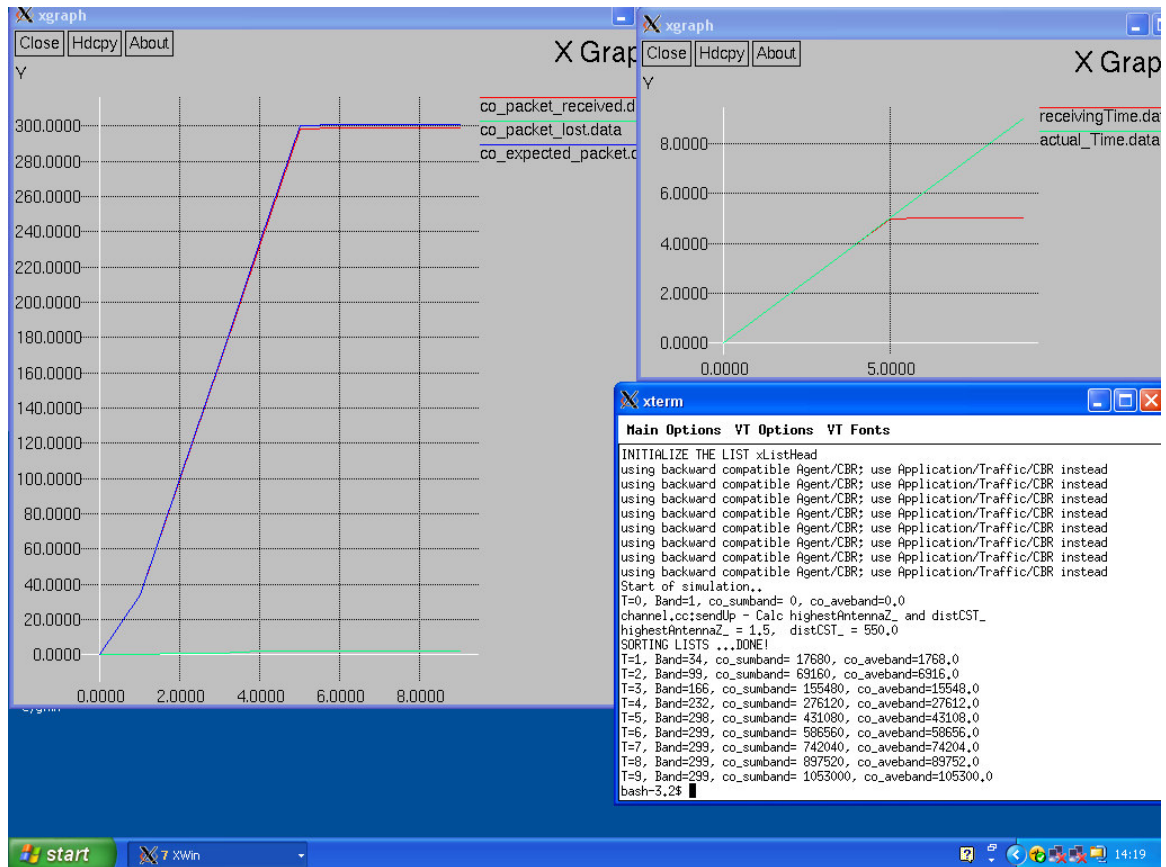


Figure 5.1 Simulation with 5% error data rate (`$err set rate_ 0.05`)

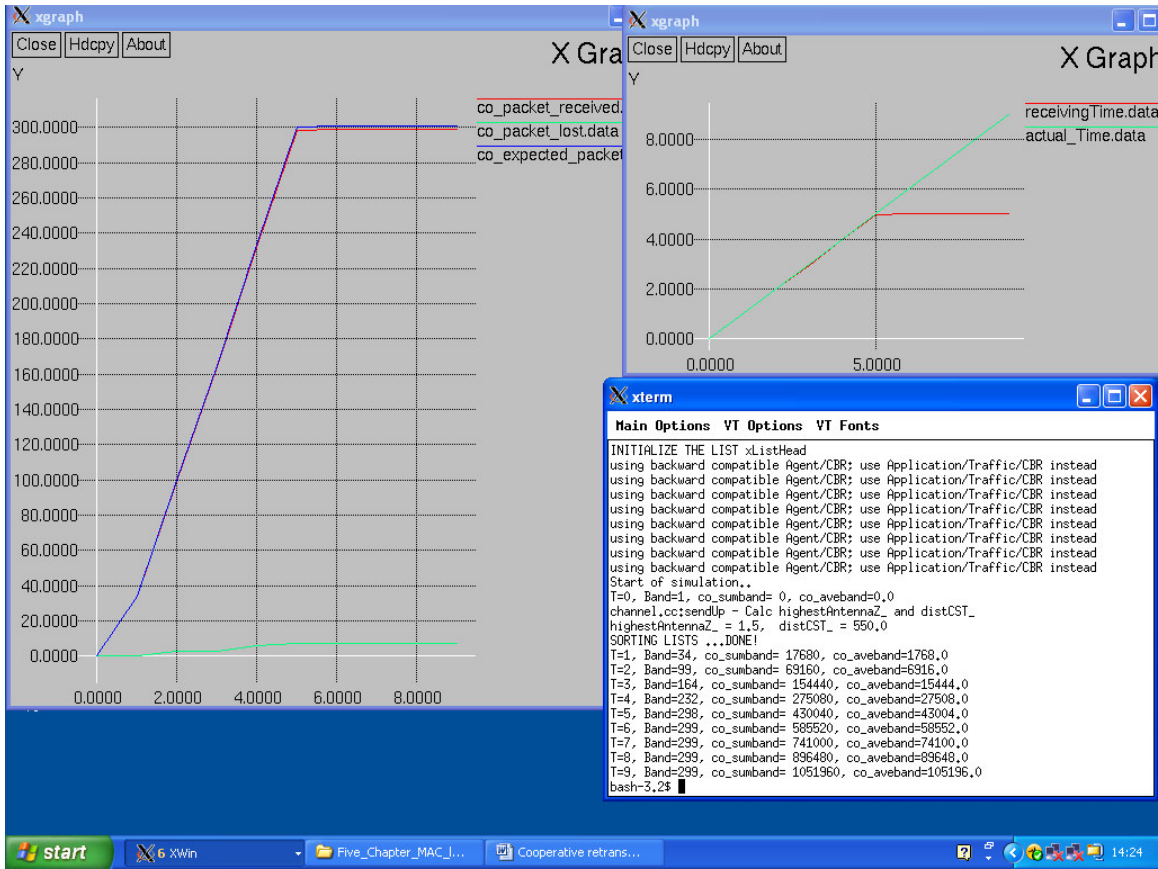


Figure 5.2 Simulation with 20% error data rate ($\$err$ set rate_ 0.20)

6. Simulation description

In wireless LAN IEEE 802.11g has become the de facto standard and most of the 802.11g functionality is implemented in hardware or firmware; modifying or extending such communication protocols running at the data link layer are mostly implemented in hardware or firmware because of the amount of micro- second timing requirements.. Under these circumstances, modifying or extending existing protocols will be difficult. A simulation model of the standard is need which allows us to find the efficient evaluation of the system performance. Several network and protocol simulator have been release both open source and commercial, but we are interest in ns-2 simulation which support 802.11a/b/g [21]

The recently developed new modeling of IEEE 802.11 for NS-2, which introduces two new modules: Mac802_11Ext and WirelessPhyExt. The extensions are based on Mac802_11 and WirelessPhy, but did a major modification to the original code, aiming at a significantly higher level of simulation accuracy. [27]

The new modeling includes the following key features:

Structured design of MAC functionality modules: transmission, reception, transmission coordination, reception coordination, backoff manager and channel state monitor [27]

Cumulative SINR computation

MAC frame capture capabilities

Multiple modulation scheme support

Packet drop tracing at PHY layer

By simply choosing these two modules and replacing MAC-802_11 and WirelessPhy used in a running TCL script, users are able to get the above benefits in wireless communication studies [27]

6.1 Simulation environment description

We use discrete event simulator ns 2.[21]. In order to provide an IEEE 802.11g model, we extended the simulation environment described in section five. We have modified, PHY and MAC layer parameters in order to simulate 802.11g standard specification [22] All these factors can bind the achievable throughput. Some parameters in Physical layer like path loss, fading; interference and noise have been taken into account because of their important effects in simulation results..[24].

The original 802.11 have been enhanced and all NS-2.31 users have the freedom to choose for their requirements and implementation to use in their study. Figure 6.1 shows the protocol stack of and NS mobile node in a NS2 simulation. That is recommended Mac-802_11Ext and WirelessPhyExt have to be used together:

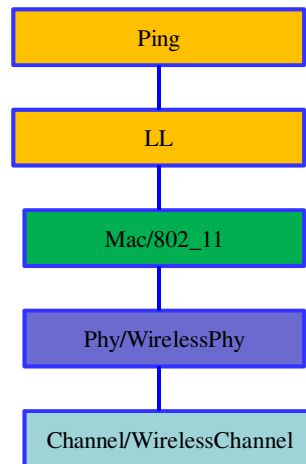


Figure 6.1 Node constructions in TCL Script

These two, Mac-802_11Ext and WirelessPhyExt are inherited classes from Mac-802_11 and WirelessPhy which are the modifications in the design and coding and functionalities of PHY, MAC changed in separated of tasks, that the major change in WirelessPhyExt implements network interface logic and monitoring the received radio power and

bandwidth is different with achievable throughput. Mac-802_11Ext task is in processing of MAC frame and maintaining the logic view of channel states and managing the back off procedure

As shown in figure 6.2 the second most distinguished change is a structured design for the internal function modules which each function module matches with IEEE standard's design and allows an easy extension or simplification and the implementation of every module is relatively independent of each other

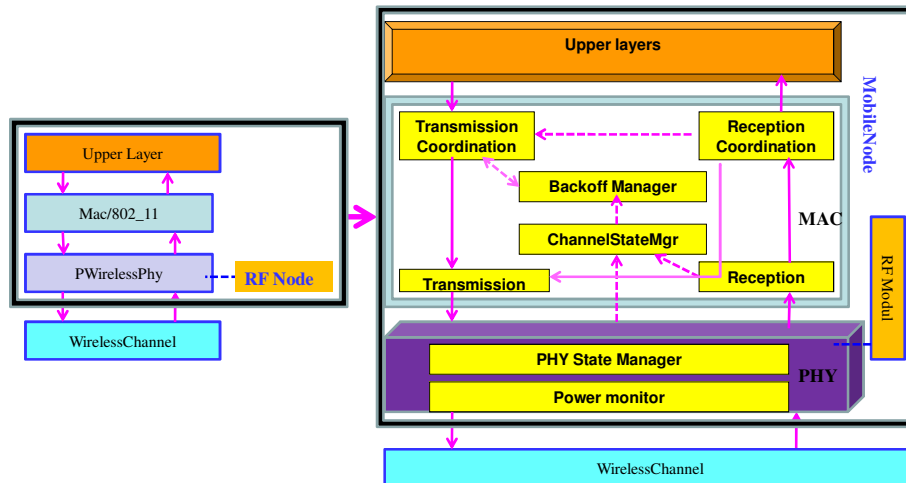


Figure 6.2 Extension of MAC and PHY modules

6.2 Simulation 802.11a

In simulation 802.11a the payload length is set to be 500 bytes. The sizes of RTS, and CTS, frames automatically are set to be 37, and 31 bytes respectively. All parameter in the physical layer and mac-802.11Ext shown in below (blue color): and configured according to the 802.11a standard. Here MAC-802_11Ext and WirelessPhyExt are implemented and tested under NS-2.35, they are introduced as new protocol modules, given the TCL class name **Mac/802_11Ext** and **Phy/WirelessPhyExt**.and we don't have the freedom to choose Mac/802_11, and Phy/WirelessPhy in my case it needs to be pointed out that Mac-802_11Ext and WirelessPhyExt have to be used together.

802_11a.tcl with 802.11a parameters

```
# =====
# Define options
# =====
set val(chan)      Channel/WirelessChannel ;# channel type
set val(prop)      Propagation/TwoRayGround ;# radio-propagation model
set val(ant)       Antenna/OmniAntenna    ;# Antenna type
set val(ll)        LL                      ;# Link layer type
set val(ifq)       Queue/DropTail/PriQueue ;# Interface queue type
set val(ifqlen)    50                      ;# max packet in ifq
```

```

set val(netif)    Phy/WirelessPhy    ;# network interface type
set val(mac)      Mac/802_11        ;# MAC type
set val(nn)       8                  ;# number of mobilenodes
set val(rp)       AODV               ;# routing protocol
set val(x)        650
set val(y)        650

```

```

#The Antenna height of transmitter and receiver is 1.5m.
#The propagation model is TwoRayGround model.
# set up the antennas to be centered in the node and 1.5 meters above it
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
Antenna/OmniAntenna set Gt_ 1.0    ;# Transmit antenna gain
Antenna/OmniAntenna set Gr_ 1.0    ;# Receive antenna gain

```

#802.11a parameters

```

Phy/WirelessPhyExt set CStresh_ 6.31e-12 ;#-82 dBm Wireless interface
sensitivity
                                ;#(sensitivity defined in the standard)
Phy/WirelessPhyExt set Pt_ 0.001
Phy/WirelessPhyExt set freq_ 5.18e+9
Phy/WirelessPhyExt set noise_floor_ 2.512e-13 ;#-96 dBm for 10MHz bandwidth
Phy/WirelessPhyExt set L_ 1.0 ;#default radio circuit gain/loss
Phy/WirelessPhyExt set PowerMonitorThresh_ 1.259e-13 ;#-99dBm power
monitor
                                ;#sensitivity
Phy/WirelessPhyExt set HeaderDuration_ 0.000020 ;#20 us
Phy/WirelessPhyExt set BasicModulationScheme_ 0
Phy/WirelessPhyExt set PreambleCaptureSwitch_ 1
Phy/WirelessPhyExt set DataCaptureSwitch_ 0
Phy/WirelessPhyExt set SINR_PreambleCapture_ 2.5118; ;# 4 dB
Phy/WirelessPhyExt set SINR_DataCapture_ 100.0; ;# 10 dB
Phy/WirelessPhyExt set trace_dist_ 1e6 ;# PHY trace until distance of 1 Mio.
km
                                ;#("infinty")
Phy/WirelessPhyExt set PHY_DBG_ 0

Mac/802_11Ext set CWMin_ 15
Mac/802_11Ext set CWMax_ 1023
Mac/802_11Ext set SlotTime_ 0.000009
Mac/802_11Ext set SIFS_ 0.000016
Mac/802_11Ext set ShortRetryLimit_ 7
Mac/802_11Ext set LongRetryLimit_ 4

```

```

Mac/802_11Ext set HeaderDuration_      0.000020
Mac/802_11Ext set SymbolDuration_      0.000004
Mac/802_11Ext set BasicModulationScheme_ 0
Mac/802_11Ext set use_802_11a_flag_    true
#Mac/802_11Ext set RTSThreshold_       2346
Mac/802_11Ext set MAC_DBG              0

```

In figure 6.5 node zero (source) communicate with node 5 and node 3 to node 2 and all station are capable to transmitting and receiving the packets Nodes are move (except source) within a specified region and communicate among themselves through one another. As we can see in code above in this case MAC-802_11Ext and Wireless-PhyExt are implemented and tested, but we are going to use (**Mac/802_11, Wireless-Phy**) in section cooperative simulation 6.4)

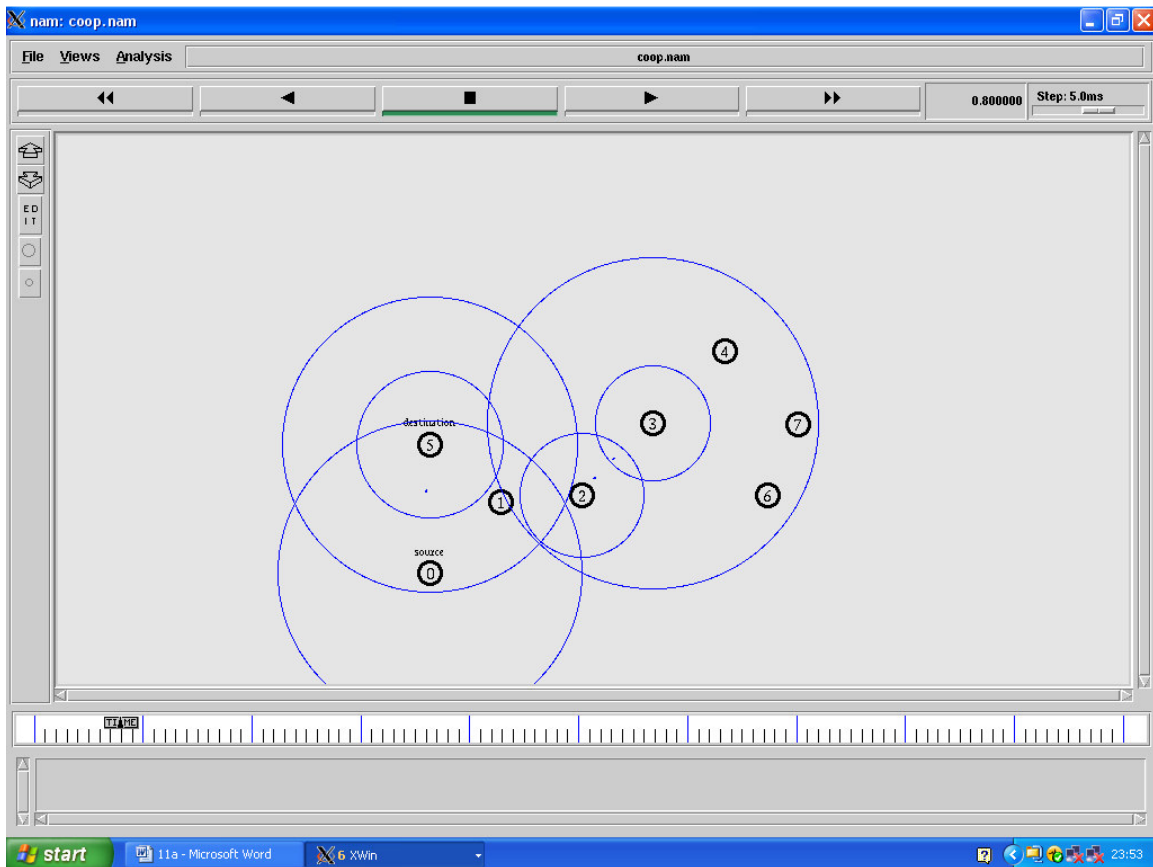


Figure 6.3 Screen shot 802.11a simulation

Figure 6.4 delay and throughput of 802_11a simulation shown that packet received are less than expected_packet, in this situation we used uniform error bit rate with 0.05 and random variable in class errormodel.c As we show in section 5.2, but receiving time and actual time is equal in all duration (between 0 to 50000) less than 10 ms

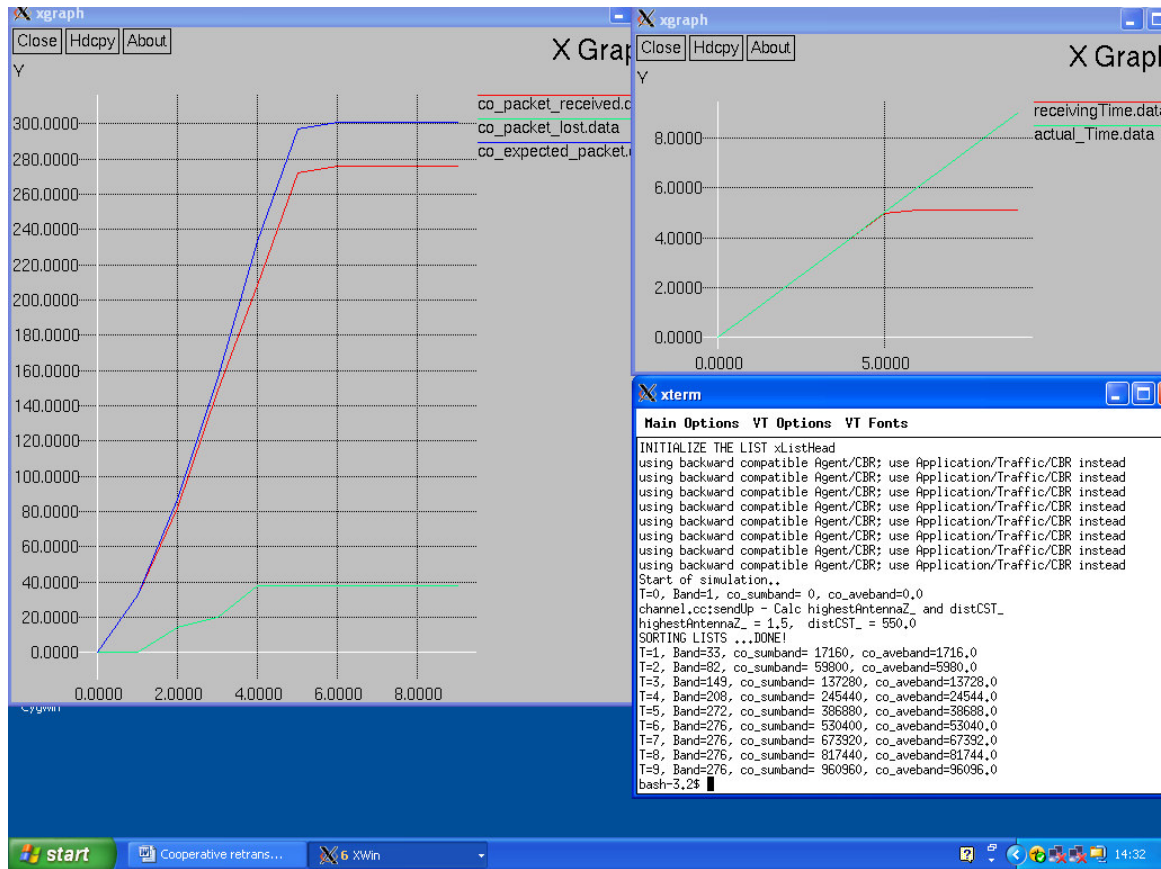


Figure 6.4 Delay and throughput analysis of 802_11a simulation

6.3 Simulation 802.11p

In simulation 802.11p the payload length is set to be 500 bytes. The sizes of RTS, and CTS, frames automatically are set to be 37, and 31 bytes respectively. All parameter in the physical layer and mac-802.11 shown in below (blue color): and configured according to the 802.11p standard and this specification doesn't work with (**Mac/802_11, Wireless-Phy**) in my case:

802_11p.tcl with 802.11p parameters

```
# =====
# Define options
# =====
set val(chan) Channel/WirelessChannel ;# channel type
```

```

set val(prop)      Propagation/TwoRayGround ;# radio-propagation model
set val(ant)       Antenna/OmniAntenna    ;# Antenna type
set val(ll)        LL                      ;# Link layer type
set val(ifq)       Queue/DropTail/PriQueue ;# Interface queue type
set val(ifqlen)    50                      ;# max packet in ifq
set val(netif)     Phy/WirelessPhy        ;# network interface type
set val(mac)       Mac/802_11             ;# MAC type
set val(nn)        8                      ;# number of mobilenodes
set val(rp)        AODV                   ;# routing protocol
set val(x)         650
set val(y)         650

```

```

#The Antenna height of transmitter and receiver is 1.5m.
#The propagation model is TwoRayGround model.
# set up the antennas to be centered in the node and 1.5 meters above it
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
Antenna/OmniAntenna set Gt_ 1.0 ;# Transmit antenna gain
Antenna/OmniAntenna set Gr_ 1.0 ;# Receive antenna gain

```

#802.11p parameters

```

Phy/WirelessPhyExt set CStresh_ 3.162e-12 ;#-85 dBm Wireless interface
#sensitivity
;#(sensitivity defined in the standard)
Phy/WirelessPhyExt set Pt_ 0.001
Phy/WirelessPhyExt set freq_ 5.9e+9
Phy/WirelessPhyExt set noise_floor_ 1.26e-13 ;#-99 dBm for 10MHz bandwidth
Phy/WirelessPhyExt set L_ 1.0 ;#default radio circuit gain/loss
Phy/WirelessPhyExt set PowerMonitorThresh_ 6.310e-14 ;#-102dBm power
monitor sensitivity
Phy/WirelessPhyExt set HeaderDuration_ 0.000040 ;#40 us
Phy/WirelessPhyExt set BasicModulationScheme_ 0
Phy/WirelessPhyExt set PreambleCaptureSwitch_ 1
Phy/WirelessPhyExt set DataCaptureSwitch_ 0
Phy/WirelessPhyExt set SINR_PreambleCapture_ 2.5118; ;# 4 dB
Phy/WirelessPhyExt set SINR_DataCapture_ 100.0; ;# 10 dB
Phy/WirelessPhyExt set trace_dist_ 1e6 ;# PHY trace until distance of 1 Mio. km
("infinty")
Phy/WirelessPhyExt set PHY_DBG_ 0

Mac/802_11Ext set CWMin_ 15
Mac/802_11Ext set CWMax_ 1023
Mac/802_11Ext set SlotTime_ 0.000013

```

```

Mac/802_11Ext set SIFS_ 0.000032
Mac/802_11Ext set ShortRetryLimit_ 7
Mac/802_11Ext set LongRetryLimit_ 4
Mac/802_11Ext set HeaderDuration_ 0.000040
Mac/802_11Ext set SymbolDuration_ 0.000008
Mac/802_11Ext set BasicModulationScheme_ 0
Mac/802_11Ext set use_802_11a_flag_ true
#Mac/802_11Ext set RTSThreshold_ 2346
Mac/802_11Ext set MAC_DBG_ 0

```

Here again we are used **Mac/802_11Ext** and **Phy/WirelessPhyExt**. NS-2, figure 6.5 delay and throughput of 802_11a simulation shown that packet received are less then expected_packet, which is same as in scenario figure 6.4, however we used different parameter in our tcl file for specification 802.11p and like common parameter such as uniform error bit rate with 0.05 and random variable in class errormodel.c, here receiving time and actual time is equal in all duration (between 0 to 50000) less then 10 ms

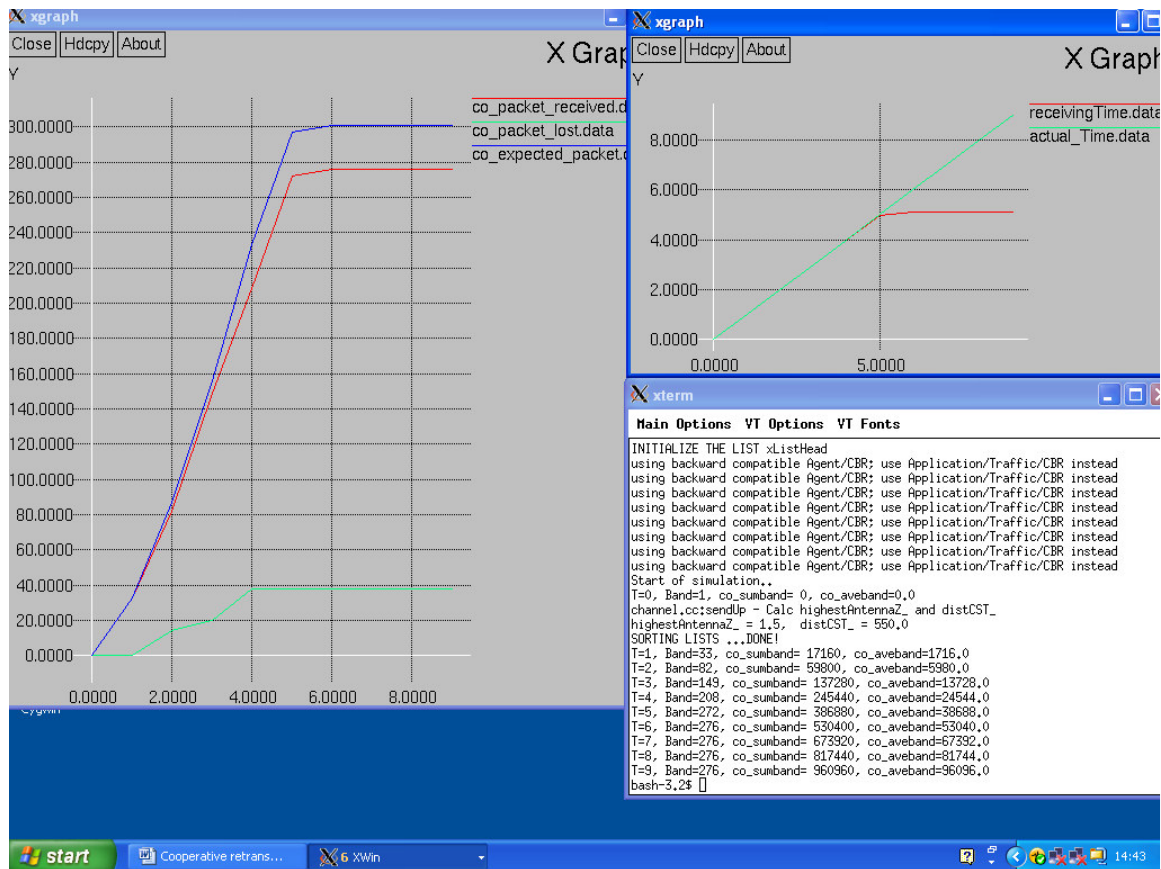


Figure 6.5 Delay and throughput analysis of 802_11p simulation

6.4 Simulation results part

The payload length is set to be 500 bytes. The sizes of RTS, and CTS, frames automatically are 37, and 31 bytes respectively. The physical layer data rate of 802.11g is set to be 54 Mbps and the basic data rate for control packets is 6 Mbps. The size of the ACK packet is 31 bytes and it is transmitted at the same rate as the data packet, i.e. at 54 Mbps.. All the other parameters are shown in table 6.1 or below (in blue color) and configured according to the 802.11 standard, we are used (**Mac/802_11, Wireless-Phy**) and some modification files (channel.c Mac-802.11.cc, WirelessPhy.cc and packet.h and can be find in appendix A) and changing parameter for cooperative such as SINR, power monitor, in order to provide an IEEE 802.11g model, we extended the simulation environment described in [27], to make a model and MAC layer parameters to 802.11g standard specification. These parameters include MAC and physical layer convergence procedure (PLCP) header formats, data rates and use of. Wireless-Phy and result can be shown in Figure 6.6 Screen shot cooperative simulation (as cooperative Retransmission phase or cooperative three-way handshaking) and in figure 6.7 Delay and throughput analysis cooperative simulation benefit of throughput and less delay

Parameters	Value
Nodes	8
SIFS	16us
DIFS	28us
Slot Time	9us
CWmin	15
CWmax	1023
Frame Type	Size in byte
RTS	37bytes
CTS	31bytes
ACK	31bytes

Table 6.1 Simulation parameters

Parameters	802.11b	802.11a	Mixed 802.11g	Pure 802.11g
Slot time	20us	9us	20us	9us
SIFS	10us	16us	10us	10us
DIFS	50us	34us	50us	28us
CWmin	16	32	32	32
CWmax	1024	1024	1024	1024

Table 6.2 Parameters for IEEE 802.11

```

# co.tcl with 802.11g parameters
# =====
# Define options
# =====
set val(chan) Channel/WirelessChannel ;# channel type
set val(prop) Propagation/TwoRayGround ;# radio-propagation model
set val(ant) Antenna/OmniAntenna ;# Antenna type
set val(ll) LL ;# Link layer type
set val(ifq) Queue/DropTail/PriQueue ;# Interface queue type
set val(ifqlen) 50 ;# max packet in ifq
set val(netif) Phy/WirelessPhy ;# network interface type
set val(mac) Mac/802_11 ;# MAC type
set val(nn) 8 ;# number of mobilenodes
set val(rp) AODV ;# routing protocol
set val(x) 650
set val(y) 650

#The Antenna height of transmitter and receiver is 1.5m.
#The propagation model is TwoRayGround model.
# set up the antennas to be centered in the node and 1.5 meters above it
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
Antenna/OmniAntenna set Gt_ 1.0 ;# Transmit antenna gain
Antenna/OmniAntenna set Gr_ 1.0 ;# Receive antenna gain

Phy/WirelessPhy set bandwidth_ 54e6 ;#Data Rate

# Initialize the SharedMedia interface with parameters to make
# it work like the 914MHz Lucent WaveLAN DSSS radio interface
#Collision Threshold
Phy/WirelessPhy set CPTthresh_ 10.0
#Receive Power Threshold;calculated under TwoRayGround model by tools from
NS2
Phy/WirelessPhy set RXThresh_ 3.652e-10
#Transmit Power
Phy/WirelessPhy set Pt_ 0.28183815
# Channel 2.4 GHz
Phy/WirelessPhy set freq_ 2.4e9
#System loss facor
Phy/WirelessPhy set L_ 1.0
#Carrier Sense Power
Phy/WirelessPhy set CSThresh_ 1.559e-11

```



```

Phy/WirelessPhy set PowerMonitorThresh_ 6.310e-14 ;#-102dBm power monitor
sensitivity
Phy/WirelessPhy set HeaderDuration_ 0.000040 ;#40 us
Phy/WirelessPhy set BasicModulationScheme_ 0
Phy/WirelessPhy set PreambleCaptureSwitch_ 1
Phy/WirelessPhy set DataCaptureSwitch_ 0
Phy/WirelessPhy set SINR_PreambleCapture_ 2.5118; ;# 4 dB
Phy/WirelessPhy set SINR_DataCapture_ 100.0; ;# 10 dB
Phy/WirelessPhy set trace_dist_ 1e6 ;# PHY trace until distance of 1
Mio.km("infinty")
Phy/WirelessPhy set PHY_DBG_ 0
Phy/WirelessPhy set noise_floor_ 1.26e-13 ;#-99 dBm for 10MHz bandwidth

```

```

#you can set dataRate for DATA here
Mac/802_11 set dataRate_ 54e6 ;# Rate for Data frame
#you can set basicRate for RTS/CTS, and ACK here
Mac/802_11 set basicRate_ 6e6 ;# Rate for Control Frame

```

```

#Mac/802_11 set RTSThreshold_ 3000 ;# Disable RTS/CTS
# 802.11g parameters

```

```

Mac/802_11 set CWMin_ 15
Mac/802_11 set CWMax_ 1023
Mac/802_11 set SlotTime_ 0.000009 ;# 9us
Mac/802_11 set CCATime_ 0.000003
Mac/802_11 set RxTxTurnaroundTime_ 0.000002
Mac/802_11 set SIFSTime_ 0.000016 ;# 16us
Mac/802_11 set DIFS_ 0.000028 ;# 50us
Mac/802_11 set PreambleLength_ 96 ;# 96 bit
Mac/802_11 set PLCPHeaderLength_ 40 ;# 40 bits
Mac/802_11 set PLCPDataRate_ 6.0e6 ;# 6Mbps
Mac/802_11 set MaxPropagationDelay_ 0.0000005 ;# 0.5us

```

```

Mac/802_11 set ShortRetryLimit_ 7
Mac/802_11 set LongRetryLimit_ 4
Mac/802_11 set HeaderDuration_ 0.000040
Mac/802_11 set SymbolDuration_ 0.000008
Mac/802_11 set BasicModulationScheme_ 0
Mac/802_11 set use_802_11a_flag_ true
#Mac/802_11 set RTSThreshold_ 2346
Mac/802_11 set MAC_DBG_ 0

```

Figure 6.6 Screen shot cooperative simulation shown phase two, where the three-way handshaking is introduced before data retransmission to protect message sequences against collision during the cooperative phase, the handshaking starts with the Relay Ready to Send, RRS (in our case in figure below node 1 as helper between source node 0 and destination node 5) frame sent by the active relay node, respond by the Destination Clear for relay to Send, DCS (here node 5 marked as destination) frame from the destination node and the Source Clear for relay to Send, SCS (that is node zero as labeled source) frame from the original source node.

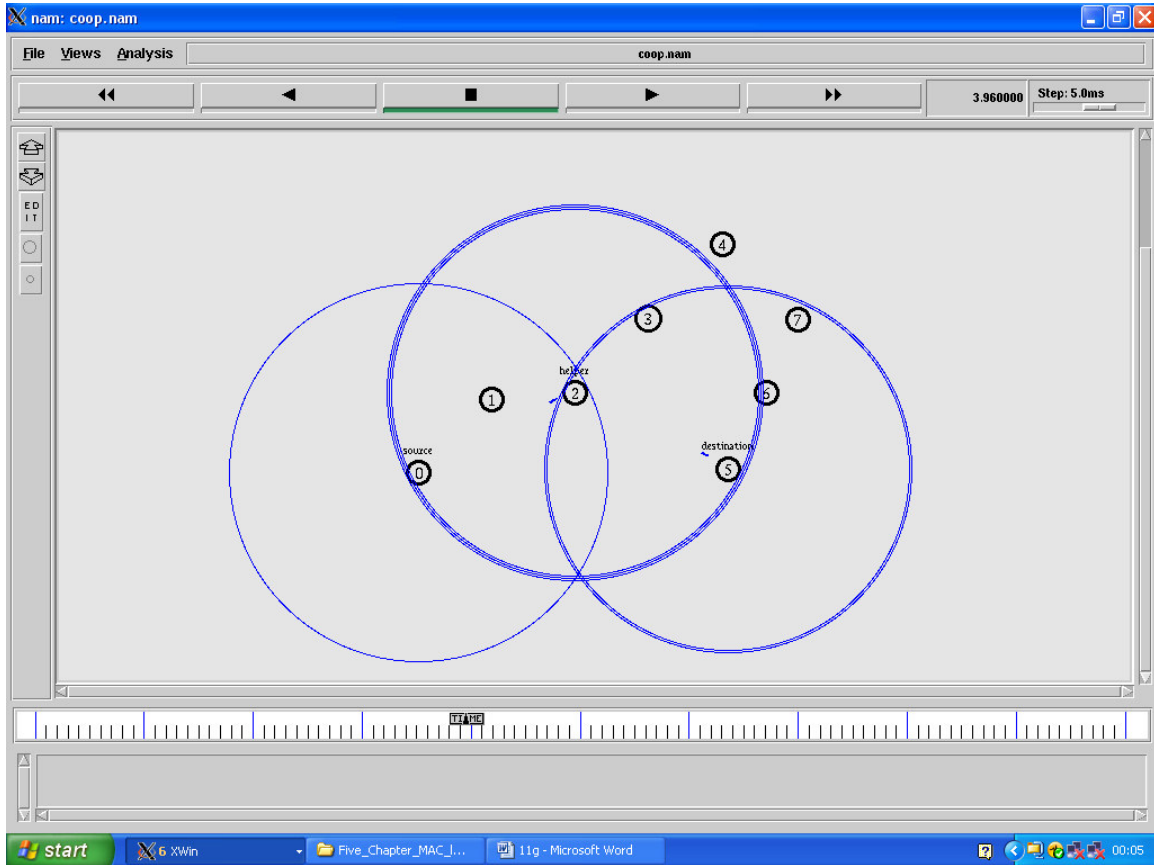


Figure 6.6 Screen shot cooperative simulation

Figure 6.7 Delay and throughput analysis cooperative simulation shown that co_packet_received.data and exected_packet.data match together and co_packet_lost is also zero., here receiving time and actual time perfectly matched, here we used same parameter in our co. tcl file such as uniform error bit rate with 0.05 and random variable in class errormodel.c, and used specification parameter for 802.11g

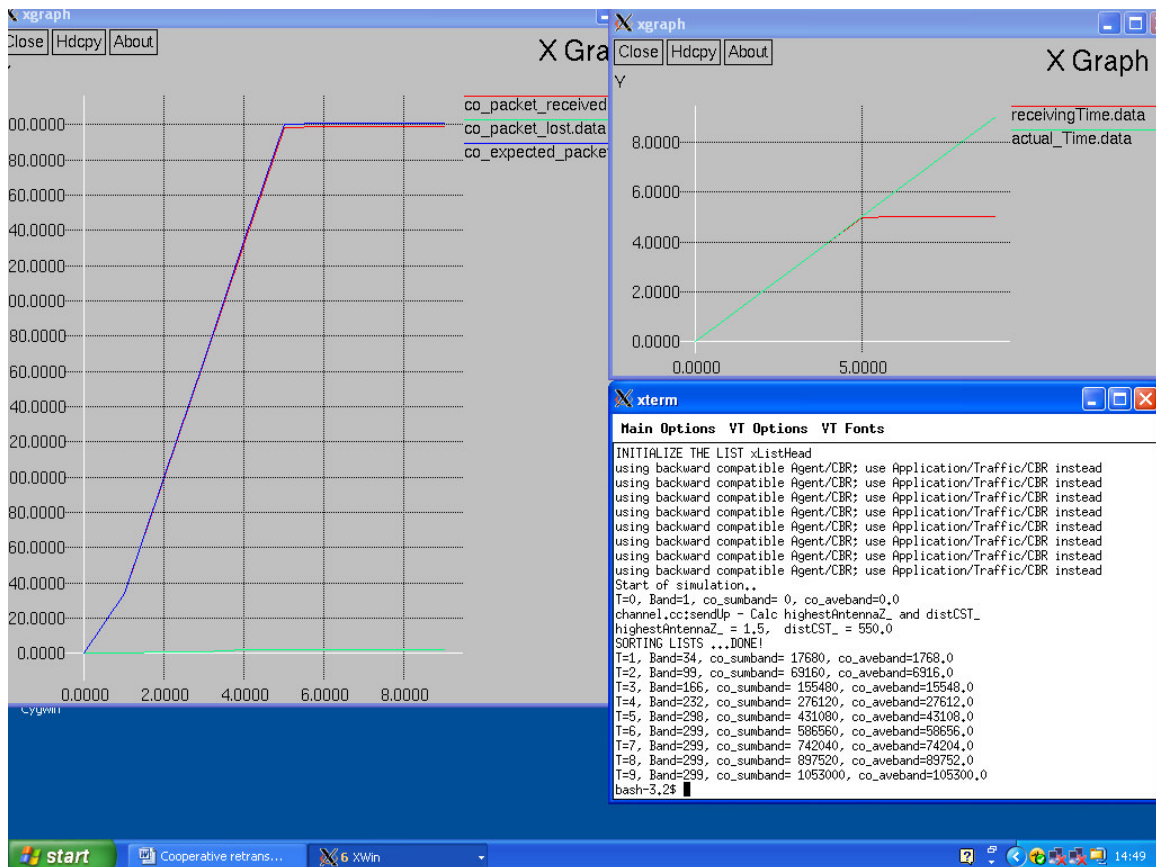


Figure 6.7 Delay and throughput analysis cooperative simulation

6.5 validation of results part

In this section, we are interested to show that our simulation in section 6.4 is correct for this reason; we are going to use **Mac/802_11Ext** and **Phy/WirelessPhyExt**, instead of **Mac/802_11** and **Phy/WirelessPhy**, therefore, using those same parameters which are used above. MAC802_11Ext and WirelessPhyExt are able to simulate MAC frame in different modulation schemes. A MAC frame modulated with an advanced modulation

scheme has shorter transmission duration but requires a higher SINR to receive and decode it.

coExT.tcl with 802_11Extg parameters

```
# =====  
# Define options  
# =====  
set val(chan) Channel/WirelessChannel ;# channel type  
set val(prop) Propagation/TwoRayGround ;# radio-propagation model  
set val(ant) Antenna/OmniAntenna ;# Antenna type  
set val(ll) LL ;# Link layer type  
set val(ifq) Queue/DropTail/PriQueue ;# Interface queue type  
set val(ifqlen) 50 ;# max packet in ifq  
set val(netif) Phy/WirelessPhy ;# network interface type  
set val(mac) Mac/802_11 ;# MAC type  
set val(nn) 8 ;# number of mobilenodes  
set val(rp) AODV ;# routing protocol  
set val(x) 650  
set val(y) 650
```

#The Antenna height of transmitter and receiver is 1.5m.

#The propagation model is TwoRayGround model.

set up the antennas to be centered in the node and 1.5 meters above it

Antenna/OmniAntenna set X_ 0

Antenna/OmniAntenna set Y_ 0

Antenna/OmniAntenna set Z_ 1.5

Antenna/OmniAntenna set Gt_ 1.0 ;# Transmit antenna gain

Antenna/OmniAntenna set Gr_ 1.0 ;# Receive antenna gain

Phy/WirelessPhyExt set bandwidth_ 54e6 ;#Data Rate

Initialize the SharedMedia interface with parameters to make

it work like the 914MHz Lucent WaveLAN DSSS radio interface

#Collision Threshold

Phy/WirelessPhyExt set CPTthresh_ 10.0

#Receive Power Threshold;calculated under TwoRayGround model by tools from NS2

Phy/WirelessPhyExt set RXThresh_ 3.652e-10

#Transmit Power

Phy/WirelessPhyExt set Pt_ 0.28183815

Channel 2.4 GHz

Phy/WirelessPhyExt set freq_ 2.4e9

#System loss facor

Phy/WirelessPhyExt set L_ 1.0

```

#Carrier Sense Power
Phy/WirelessPhyExt set CStresh_ 1.559e-11

Phy/WirelessPhyExt set PowerMonitorThresh_ 6.310e-14 ;#-102dBm power
monitor sensitivity
Phy/WirelessPhyExt set HeaderDuration_ 0.000040 ;#40 us
Phy/WirelessPhyExt set BasicModulationScheme_ 0
Phy/WirelessPhyExt set PreambleCaptureSwitch_ 1
Phy/WirelessPhyExt set DataCaptureSwitch_ 0
Phy/WirelessPhyExt set SINR_PreambleCapture_ 2.5118; ;# 4 dB
Phy/WirelessPhyExt set SINR_DataCapture_ 100.0; ;# 10 dB
Phy/WirelessPhyExt set trace_dist_ 1e6 ;# PHY trace until distance of 1
Mio.km("infinty")
Phy/WirelessPhyExt set PHY_DBG_ 0
Phy/WirelessPhyExt set noise_floor_ 1.26e-13 ;#-99 dBm for 10MHz bandwidth

#you can set dataRate for DATA here
Mac/802_11Ext set dataRate_ 54e6 ;# Rate for Data frame
#you can set basicRate for RTS/CTS, and ACK here
Mac/802_11Ext set basicRate_ 6e6 ;# Rate for Control Frame

#Mac/802_11 set RTSThreshold_ 3000 ;# Disable RTS/CTS
# 802.11g parameters

Mac/802_11Ext set CWMin_ 15
Mac/802_11Ext set CWMax_ 1023
Mac/802_11Ext set SlotTime_ 0.000009 ;# 9us
Mac/802_11Ext set CCATime_ 0.000003
Mac/802_11Ext set RxTxTurnaroundTime_ 0.000002
Mac/802_11Ext set SIFSTime_ 0.000016 ;# 16us
Mac/802_11Ext set DIFS_ 0.000028 ;# 50us
Mac/802_11Ext set PreambleLength_ 96 ;# 96 bit
Mac/802_11Ext set PLCPHeaderLength_ 40 ;# 40 bits
Mac/802_11Ext set PLCPDataRate_ 6.0e6 ;# 6Mbps
Mac/802_11Ext set MaxPropagationDelay_ 0.0000005 ;# 0.5us

Mac/802_11Ext set ShortRetryLimit_ 7
Mac/802_11Ext set LongRetryLimit_ 4
Mac/802_11Ext set HeaderDuration_ 0.000040
Mac/802_11Ext set SymbolDuration_ 0.000008
Mac/802_11Ext set BasicModulationScheme_ 0
Mac/802_11Ext set use_802_11a_flag_ true
#Mac/802_11Ext set RTSThreshold_ 2346
Mac/802_11Ext set MAC_DBG_ 0

```

Mac-802_11Ext and WirelessPhyExt simulations shown that ACK, CTS sizes are 38 bytes for each and 44 bytes for RTS, sine above in section 6.4 simulation respectively are 31 bytes, 31bytes, and 38 bytes. In this case more significant improvement scheme is observe in figure 6.8 (Enhanced) Delay and throughput analysis cooperative simulation which confirms that, we have got different values since, with same parameter in both cases, and **Mac/802_11** and **Phy/WirelessPhy** files have better simulation result, therefore can be used in our furthermore simulations

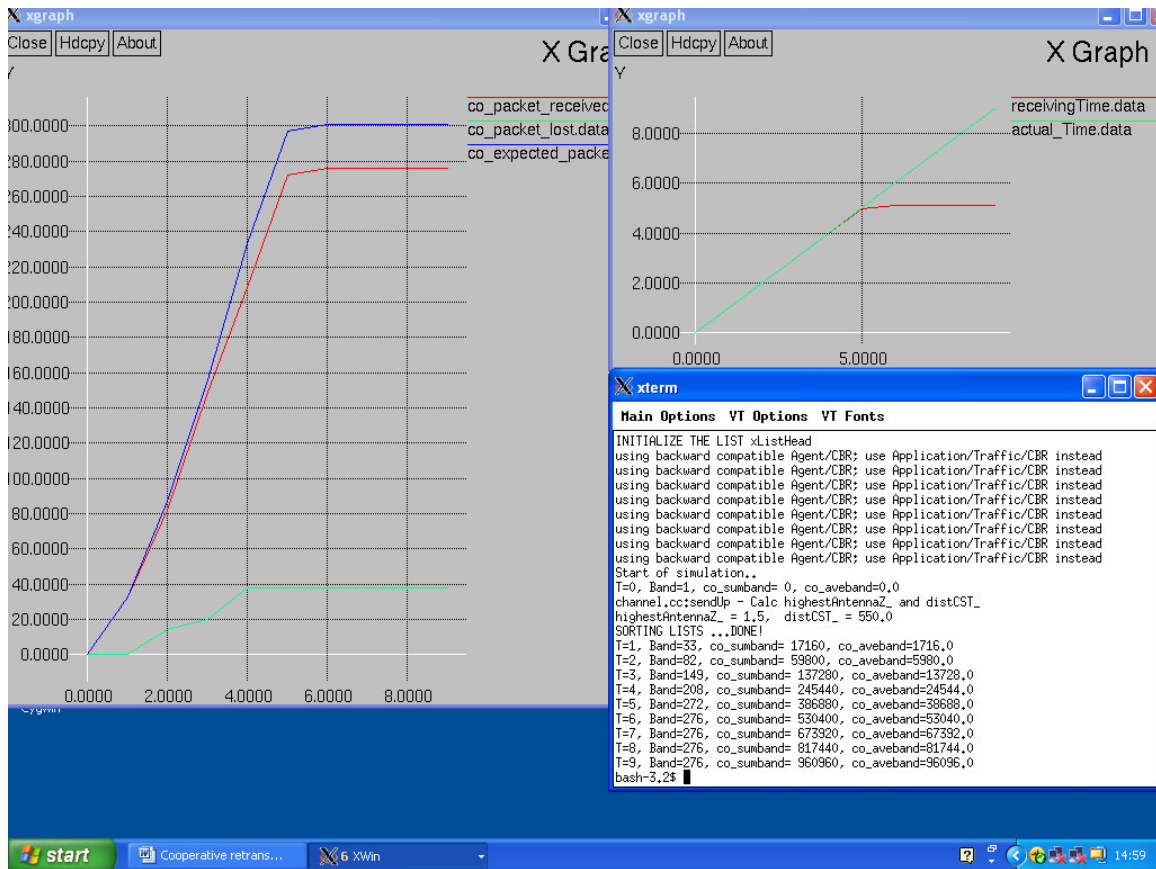


Figure 6.8 (Enhanced) Delay and throughput analysis cooperative simulation

Below you can see a simulation throughput with sixteen nodes are moving a extended topology and communicate among themselves through one another as shown some packet drop when the helper exchanges, the payload length is set to be 500 bytes in interval 0.015 the size of RTS, and CTS, frames automatically are set to be 37, and 31 bytes respectively in all duration, here packet received match exactly with expected packet and packet lost close to zero, for duration time sending and receiving are delay less then 10ms and the best of our all simulations.

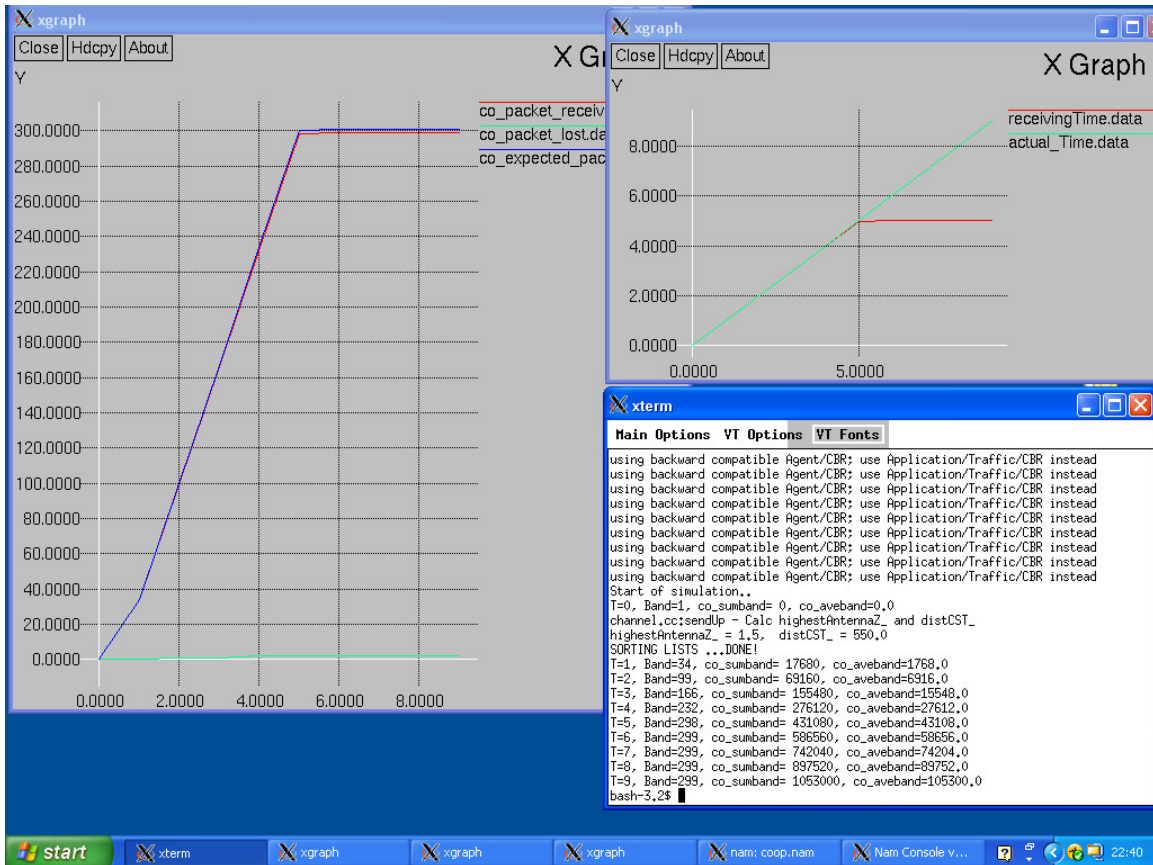


Figure 6.9 delay and throughput analysis cooperative simulation with teen nodes

It can be observed that the automatic cooperative retransmission mac protocol in wireless network into ns 2 obviously reduce the access delay significantly when cooperative retransmission take place, as shown packet lost is close to zero and bite error rate must be roughly reduced and expect cooperative packet match exactly in both case as shown in figure 6.10 and 6.11

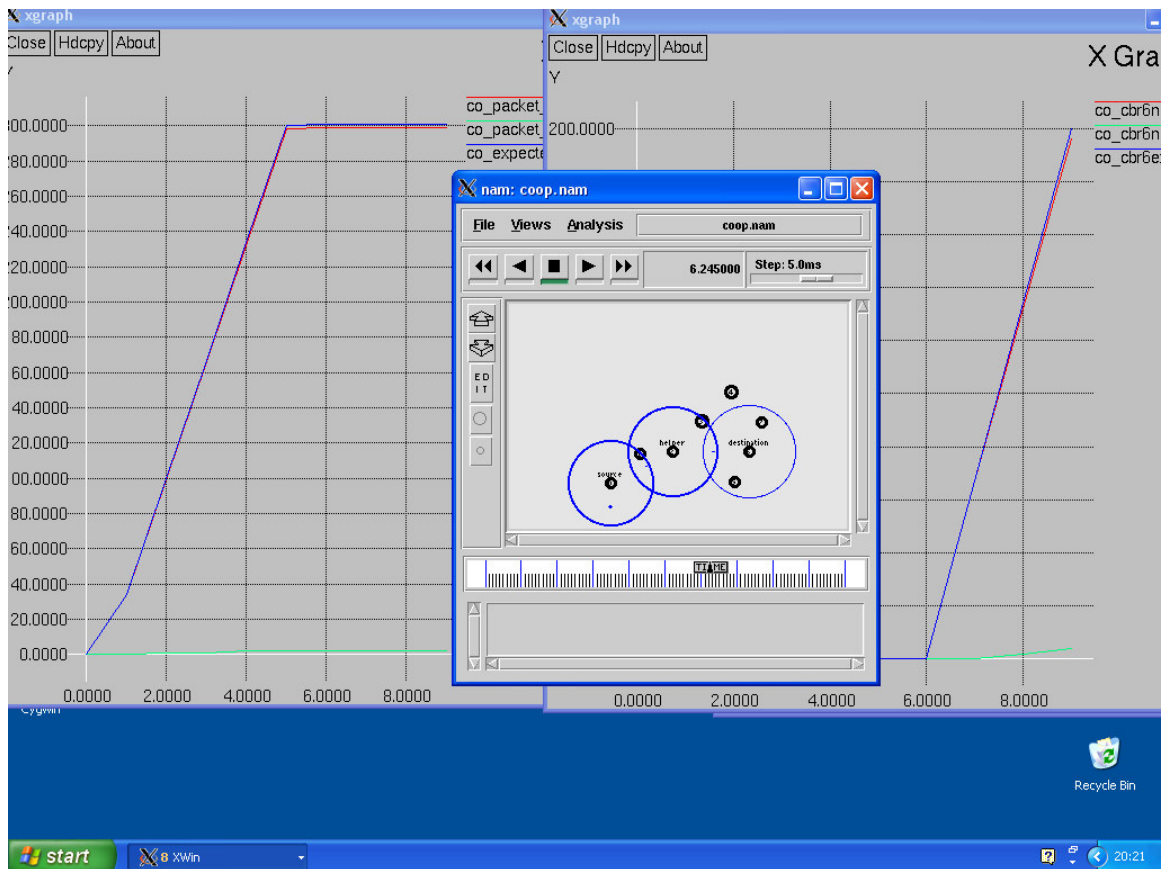


Figure 6.10 delay and throughput analysis cooperative simulation with teen nodes

In Figure 6.11 show the packet delivery different time and same rate performance and lost packet comparison of the proposed cooperative scheme. Both node using same threshold SNR for the relay selections in simulation

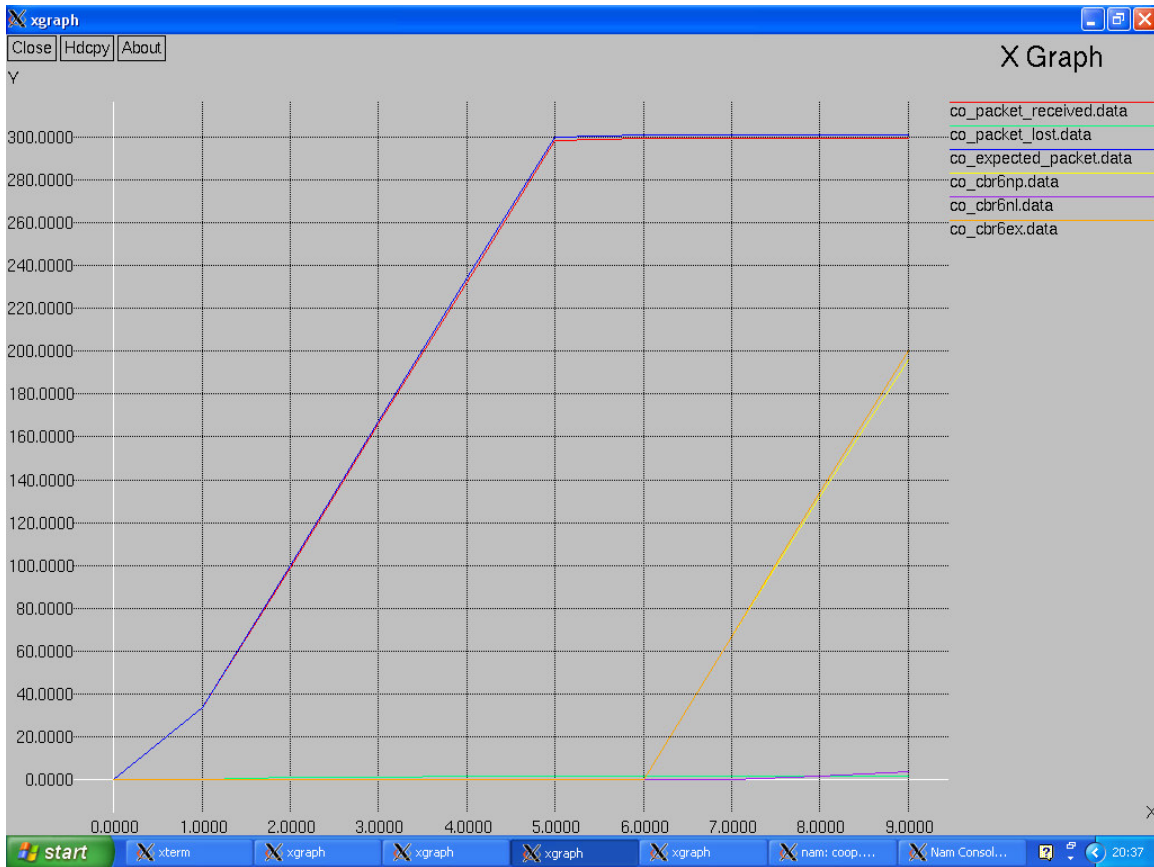


Figure 6.11 delay and throughput analysis cooperative simulation with teen nodes

7. Conclusions

We have simulation Cooperative Retransmission MAC Protocol in wireless network which Implementation in ns 2 with its IEEE 802.11 support is a widely utilized simulation tool for wireless communications researchers. We want to answer three key issue, namely, when to cooperate, whom to cooperate with and how to protect cooperative transmissions. In a simulation scenario, the wireless channel will be interconnecting all wireless nodes that interchange frames among them. The wireless channel task is to pass frames from each sender to all possible receivers, creating an exclusive copy for each of them, as we implement in first step to adding some basic simulator calculates the performance trend with greater precision that such as, if collision occur when the packet occupies the channel for the transmission time, by setting the error bit or discard them, and the corrupt will be dropping, in addition the channel contention which determines when the contention window is over, with schedule a callback to the MAC. Second step for improve of simulations of wireless networks, especially for several nodes, we would like modification in some files caused better simulation such as uniform error and random variable, for incoming nodes which all are attached to a common wireless channel, when a node transmits, the channel notifies all attached nodes of the transmission event with location, power, and each node communicates with a RF Model, which computes received signal strength for each incoming transmission using a selected RF propagation model and has a critical error in design or implementation., in the MAC module only operates at the logical level that depends on physical layer to handle actual transmission, receptions and physical channel sensing. When we come to third steps particularly in MAC protocol design for cooperative communication based on 802.11 CSMA/CA MAC protocol and three key issues which focus of the MAC design is to correctly the complexities in the IEEE 802.11 CSMA/CA mechanism that is also known as Distributed Coordination Function (DCF) , we answer this with little modification and different parameter in new design and new version is to make it easy for users to understand and extend the MAC for their study.

8. Appendix

Appendix A

Files in ns-2 simulation have been used

In Channel.c; all added

```
send():
/* The packet occupies the channel for the transmission time, txtime
If collision occur (>1 pkts overlap), corrupt all pkts involved
by setting the error bit or discard them
*/

int Channel::send(Packet* p, Phy *tifp)
{
//without collision, return 0
Scheduler& s = Scheduler::instance();
double now = s.clock();

// busy = time when the channel are still busy with earlier tx
double busy = max(txstop_, cwstop_);

// txstop = when the channel is no longer busy from this tx
txstop_ = max(busy, now + txtime);

/* now < busy => collision
mark the pkt error bit, EF_COLLISION
drop if there is a drop target, drop_
*/
if (now < busy) {
// if still transmit earlier packet, pkt_, then corrupt it
if (pkt_ && pkt_->time_ > now) {
hdr_cmn::access(pkt_->error() |= EF_COLLISION;
if (drop_) {
s.cancel(pkt_);
drop(pkt_);
pkt_ = 0;
}
}
}
```

```

// corrupts the current packet p, and drop if drop_ exists
hdr_cmn::access(p)->error() |= EF_COLLISION;
if (drop_) {
drop(p);
return 1;
}
}

// if p was not dropped, call recv() or hand it to trace_ if present
pkt_ = p;
trace_ ? trace_->recv(p, 0) : recv(p, 0);
return 0;
}

```

```

contention():
/*The MAC calls this Channel::contention() to enter contention period
It determines when the contention window is over, cwstop_,
and schedule a callback to the MAC for the actual send()
*/

```

```

void Channel::contention(Packet* p, Handler* h)
{
Scheduler& s = Scheduler::instance();
double now = s.clock();
if (now > cwstop_) {
cwstop_ = now + delay_;
numtx_ = 0;
}
numtx_++;
s.schedule(h, p, cwstop_ - now);
}

```

```

jam():
/* Jam the channel for a period txtime
Some MAC protocols use this to let other MAC detect collisions
*/

```

```

int Channel::jam(double txtime)
{
// without collision, return 0
double now = Scheduler::instance().clock();
if (txstop_ > now) {
txstop_ = max(txstop_, now + txtime);
}
}

```

```

return 1;
}
txstop_ = now + txtime;
return (now < cwstop_);
}

```

```

int DuplexChannel::send(Packet* p, double txtime)
{
double now = Scheduler::instance().clock();
txstop_ = now + txtime;
trace_ ? trace_->rcv(p, 0) : rcv(p, 0);
return 0;
}

```

```

void DuplexChannel::contention(Packet* p, Handler* h)
{
Scheduler::instance().schedule(h, p, delay_);
numtx_ = 1;
}

```

```

=====
=====

```

File: in ns-2 simulation

In WirelessPhy.cc; which has been changed. Here red means removed and blue added

```

if(propagation_) {
s.stamp((MobileNode*)node(), ant_, 0, lambda_);
Pr = propagation_->Pr(&p->txinfo_, &s, this);
if (Pr < CStresh_) {
pkt_rcvd = 0;
goto DONE;
}
if (Pr < RXThresh_) {
/*
* We can detect, but not successfully receive
* this packet.
*/
hdr_cmn *hdr = HDR_CMN(p);
//hdr->error() = 1;
hdr->busy() = 1;
#if DEBUG > 3
printf("SM %f.9 _%d_ drop pkt from %d low POWER %e/%e\n",
Scheduler::instance().clock(), node()->index(),
p->txinfo_.getNode()->index(),

```

```
Pr,RXThresh);
#endif
```

```
=====
```

File:in ns-2 simulation

In Packet.h: which has been changed . Here red means removed and blue added

```
struct hdr_cmn {
    enum dir_t { DOWN= -1, NONE= 0, UP= 1 };
    packet_t ptype_; // packet type (see above)
    int size_; // simulated packet size
    int uid_; // unique id
    //int error_; // error flag
    int busy_; // error flag
    int errbitcnt_; // # of corrupted bits jahn
    int fecsize_;
    double ts_; // timestamp: for q-delay measurement
    int iface_; // receiving interface (label)
    dir_t direction_; // direction: 0=none, 1=up, -1=down
    // source routing
    char src_rt_valid;
    double ts_arr_; // Required by Marker of JOBS

    //Monarch extn begins
    nsaddr_t prev_hop_; // IP addr of forwarding hop
    nsaddr_t next_hop_; // next hop for this packet
    int addr_type_; // type of next_hop_addr
    nsaddr_t last_hop_; // for tracing on multi-user channels

    // AOMDV patch
    int aomdv_salvage_count_;

    // called if pkt can't obtain media or isn't ack'd. not called if
    // dropped by a queue
    FailureCallback xmit_failure_;
    void *xmit_failure_data_;

    /*
     * MONARCH wants to know if the MAC layer is passing this back because
     * it could not get the RTS through or because it did not receive
     * an ACK.
     */
    int xmit_reason_;
#define XMIT_REASON_RTS 0x01
#define XMIT_REASON_ACK 0x02
```

```

    // filled in by GOD on first transmission, used for trace analysis
    int num_forwards_; // how many times this pkt was forwarded
    int opt_num_forwards_; // optimal #forwards
// Monarch extn ends;

// tx time for this packet in sec
double txtime_;
inline double& txtime() { return(txtime_); }

static int offset_; // offset for this header
inline static int& offset() { return offset_; }
inline static hdr_cmn* access(const Packet* p) {
    return (hdr_cmn*) p->access(offset_);
}

    /* per-field member functions */
inline packet_t& ptype() { return (ptype_); }
inline int& size() { return (size_); }
inline int& uid() { return (uid_); }
//inline int& error() { return error_; }
inline int& busy() { return busy_; }

inline int& errbitcnt() {return errbitcnt_; }
inline int& fecsize() {return fecsize_; }
inline double& timestamp() { return (ts_); }
inline int& iface() { return (iface_); }
inline dir_t& direction() { return (direction_); }
// monarch_begin
inline nsaddr_t& next_hop() { return (next_hop_); }
inline int& addr_type() { return (addr_type_); }
inline int& num_forwards() { return (num_forwards_); }
inline int& opt_num_forwards() { return (opt_num_forwards_); }
//monarch_end

ModulationScheme mod_scheme_;
inline ModulationScheme& mod_scheme() { return (mod_scheme_); }
};

```

File

Mac802_11.cc; which has been changed . Here red means removed and blue added

```

Mac802_11::recv_timer()
{

```

```

u_int32_t src;
hdr_cmn *ch = HDR_CMN(pktRx_);
hdr_mac802_11 *mh = HDR_MAC802_11(pktRx_);
u_int32_t dst = ETHER_ADDR(mh->dh_ra);
u_int32_t ap_dst = ETHER_ADDR(mh->dh_3a);
u_int8_t type = mh->dh_fc.fc_type;
u_int8_t subtype = mh->dh_fc.fc_subtype;

assert(pktRx_);
assert(rx_state_ == MAC_RECV || rx_state_ == MAC_COLL);

/*
 * If the interface is in TRANSMIT mode when this packet
 * "arrives", then I would never have seen it and should
 * do a silent discard without adjusting the NAV.
 */
if(tx_active_) {
    Packet::free(pktRx_);
    goto done;
}

/*
 * Handle collisions.
 */
if(rx_state_ == MAC_COLL) {
    discard(pktRx_, DROP_MAC_COLLISION);
    set_nav(usec(phymib_.getEIFS()));
    goto done;
}

/*
 * Check to see if this packet was received with enough
 * bit errors that the current level of FEC still could not
 * fix all of the problems - ie; after FEC, the checksum still
 * failed.
 */
//if( ch->error() ) {
if( ch->busy() ) {
    Packet::free(pktRx_);
    set_nav(usec(phymib_.getEIFS()));
    goto done;
}

```

co.tcl (802.11g)

```
# =====  
# Define options  
# =====  
set val(chan)    Channel/WirelessChannel ;# channel type  
set val(prop)    Propagation/TwoRayGround ;# radio-propagation model  
set val(ant)     Antenna/OmniAntenna    ;# Antenna type  
set val(ll)      LL                      ;# Link layer type  
set val(ifq)     Queue/DropTail/PriQueue ;# Interface queue type  
set val(ifqlen)  50                      ;# max packet in ifq  
set val(netif)   Phy/WirelessPhy        ;# network interface type  
set val(mac)     Mac/802_11             ;# MAC type  
set val(nn)      8                      ;# number of mobilenodes  
set val(rp)      AODV                   ;# routing protocol  
set val(x)       650  
set val(y)       650
```

#The Antenna height of transmitter and receiver is 1.5m.

#The propagation model is TwoRayGround model.

set up the antennas to be centered in the node and 1.5 meters above it

Antenna/OmniAntenna set X_ 0

Antenna/OmniAntenna set Y_ 0

Antenna/OmniAntenna set Z_ 1.5

Antenna/OmniAntenna set Gt_ 1.0 ;# Transmit antenna gain

Antenna/OmniAntenna set Gr_ 1.0 ;# Receive antenna gain

Phy/WirelessPhy set bandwidth_ 54e6 ;#Data Rate

Initialize the SharedMedia interface with parameters to make

it work like the 914MHz Lucent WaveLAN DSSS radio interface

#Collision Threshold

Phy/WirelessPhy set CPTresh_ 10.0

#Receive Power Threshold;calculated under TwoRayGround model by tools from NS2

Phy/WirelessPhy set RXThresh_ 3.652e-10

#Transmit Power

Phy/WirelessPhy set Pt_ 0.28183815

Channel 2.4 GHz

Phy/WirelessPhy set freq_ 2.4e9

#System loss facor

Phy/WirelessPhy set L_ 1.0

#Carrier Sense Power

Phy/WirelessPhy set CStresh_ 1.559e-11

```

Phy/WirelessPhy set PowerMonitorThresh_ 6.310e-14 ;#-102dBm power monitor
sensitivity
Phy/WirelessPhy set HeaderDuration_ 0.000040 ;#40 us
Phy/WirelessPhy set BasicModulationScheme_ 0
Phy/WirelessPhy set PreambleCaptureSwitch_ 1
Phy/WirelessPhy set DataCaptureSwitch_ 0
Phy/WirelessPhy set SINR_PreambleCapture_ 2.5118; ;# 4 dB
Phy/WirelessPhy set SINR_DataCapture_ 100.0; ;# 10 dB
Phy/WirelessPhy set trace_dist_ 1e6 ;# PHY trace until distance of 1 Mio.km("infinty")
Phy/WirelessPhy set PHY_DBG_ 0
Phy/WirelessPhy set noise_floor_ 1.26e-13 ;#-99 dBm for 10MHz bandwidth

```

```

#you can set dataRate for DATA here
Mac/802_11 set dataRate_ 54e6 ;# Rate for Data frame
#you can set basicRate for RTS/CTS, and ACK here
Mac/802_11 set basicRate_ 6e6 ;# Rate for Control Frame

```

```

#Mac/802_11 set RTSThreshold_ 3000 ;# Disable RTS/CTS
# 802.11g parameters

```

```

Mac/802_11 set CWMin_ 15
Mac/802_11 set CWMax_ 1023
Mac/802_11 set SlotTime_ 0.000009 ;# 9us
Mac/802_11 set CCATime_ 0.000003
Mac/802_11 set RxTxTurnaroundTime_ 0.000002
Mac/802_11 set SIFSTime_ 0.000016 ;# 16us
Mac/802_11 set DIFS_ 0.000028 ;# 50us
Mac/802_11 set PreambleLength_ 96 ;# 96 bit
Mac/802_11 set PLCPHeaderLength_ 40 ;# 40 bits
Mac/802_11 set PLCPDataRate_ 6.0e6 ;# 6Mbps
Mac/802_11 set MaxPropagationDelay_ 0.0000005 ;# 0.5us

```

```

Mac/802_11 set ShortRetryLimit_ 7
Mac/802_11 set LongRetryLimit_ 4
Mac/802_11 set HeaderDuration_ 0.000040
Mac/802_11 set SymbolDuration_ 0.000008
Mac/802_11 set BasicModulationScheme_ 0
Mac/802_11 set use_802_11a_flag_ true
#Mac/802_11 set RTSThreshold_ 2346
Mac/802_11 set MAC_DBG_ 0

```

```

# fix for when RTS/CTS not used
#Mac/802_11 set RTS_ 20.0
#Mac/802_11 set CTS_ 14.0

#To create a simulator object. This can be done with the command:
set ns [new Simulator]

set f [open coop.tr w]
$ns trace-all $f

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

set f0 [open co_packet_received.data w]
set f1 [open co_packet_lost.data w]
set f2 [open co_expected_packet.data w]
set f3 [open byte.data w]
set f4 [open receivingTime.data w]
set f5 [open actual_Time.data w]

set f8 [open co_cbr6np.data w]
set f9 [open co_cbr6nl.data w]
set f10 [open co_cbr6ex.data w]
set f11 [open cbr6by.data w]
set f12 [open cbr6re.data w]
set f13 [open co_sumband.data w]
set f14 [open co_avecbr6.data w]

set f6 [open co_sumband.data w]
set f7 [open co_aveband.data w]
#set f8 [open sequence_number.data w]

set topo [new Topography]
$topo load_flatgrid 650 650

create-god $val(nn)

set chan_1 [new $val(chan)]
set chan_2 [new $val(chan)]
set chan_3 [new $val(chan)]
set chan_4 [new $val(chan)]
set chan_5 [new $val(chan)]

```

```

set chan_6 [new $val(chan)]
set chan_7 [new $val(chan)]
set chan_8 [new $val(chan)]

# CONFIGURE AND CREATE NODES

proc UniformErr {} {
set err [new ErrorModel]
$err unit packet
$err set rate_ 0.05
$err ranvar [new RandomVariable/Uniform]
$err drop-target [new Agent/Null]
return $err}

$ns node-config -adhocRouting $val(rp) \
    -llType $val(ll) \
        -macType $val(mac) \
        -ifqType $val(ifq) \
        -ifqLen $val(ifqlen) \
        -antType $val(ant) \
        -propType $val(prop) \
        -phyType $val(netif) \
        #-channelType $val(chan) \
        -topoInstance $topo \
        -agentTrace OFF \
        -routerTrace ON \
        -macTrace ON \
        -movementTrace OFF \
        -channel $chan_1 \
        #-energyModel EnergyModel \
        -IncomingErrProc UniformErr

for {set i 0} {$i < $val(nn)} {incr i} {
    set node_($i) [$ns node]
    $node_($i) random-motion 0 ;# disable random motion
}

proc finish {} {
    global ns f f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 f10 f11 f12 f13 f14 namtrace
    $ns flush-trace
    close $namtrace
    close $f0
    close $f1
    close $f2
    close $f3
    close $f4
}

```

```
close $f5
close $f6
close $f7
close $f8
close $f9
close $f10
close $f11
close $f12
close $f13
```

```
exec xgraph co_packet_received.data co_packet_lost.data co_expected_packet.data &
exec xgraph receivingTime.data actual_Time.data -geometry 500x300 &
exec xgraph byte.data co_sumband.data -geometry 400x300 &
```

```
exec xgraph co_packet_received.data co_packet_lost.data co_expected_packet.data
co_cbr6np.data co_cbr6nl.data co_cbr6ex.data &
exec xgraph cbr6re.data -geometry 500x300 &
exec xgraph byte.data co_sumband.data cbr6by.data co_sumcbr6.data -geometry
400x300 &
```

```
    exec nam -r 5m coop.nam &
    exit 0
}
set co_sumband 0
set co_aveband 0
set co_sumcbr6 0
set co_avecbr6 0
```

```
proc record { } {
    global sink0 sink1 sink2 sink3 sink4 sink5 sink6 sink7 f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 f10
    f11 f12 f13 f14 co_sumband co_aveband co_sumcbr6 co_avecbr6
```

```
    #Get An Instance Of The Simulator
    set ns [Simulator instance]
```

```
    #Set The Time After Which The Procedure Should Be Called Again
    set time 1.0
```

```
    #How Many Bytes Have Been Received By The Traffic Sinks?
```

```
    set bw0 [$sink5 set npkts_]
    set bw1 [$sink5 set nlost_]
    set bw [$sink5 set expected_]
    set byte [$sink5 set bytes_]
    set receivingTime [$sink5 set lastPktTime_]
```

```
set cbr6np [$sink0 set npkts_]
set cbr6nl [$sink0 set nlost_]
set cbr6ex [$sink0 set expected_]
set cbr6by [$sink0 set bytes_]
set cbr6re [$sink0 set lastPktTime_]
```

```
#Get The Current Time
```

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

```
#Save Data To The Files
```

```
puts $f0 "$now [expr $bw0]"
puts $f1 "$now [expr $bw1]"
puts $f2 "$now [expr $bw]"
puts $f3 "$now [expr $byte]"
puts $f4 "$now [expr $receivingTime]"
```

```
puts $f5 "$now [expr $now]"
puts $f6 "$now [expr $co_sumband]"
puts $f7 "$now [expr $co_aveband]"
```

```
puts $f8 "$now [expr $cbr6np]"
puts $f9 "$now [expr $cbr6nl]"
puts $f10 "$now [expr $cbr6ex]"
puts $f11 "$now [expr $cbr6by]"
puts $f12 "$now [expr $cbr6re]"
puts $f13 "$now [expr $co_sumcbr6]"
puts $f14 "$now [expr $co_avecbr6]"
```

```
#calculate loss ratio, to avoid divided-by-zero error
```

```
if {$bw0==0} {set bw0 1}
```

```
set co_sumband [expr $co_sumband + $byte]
```

```
#average throughput in bps, 10 sec is session time
```

```
set co_aveband [expr double($co_sumband)/10]
```

```
puts "T=$now, Band=$bw0, co_sumband= $co_sumband, co_aveband=$co_aveband"
```

```
#puts " At time = $now, Loss ratio [expr double($bw1)/double($bw1+$bw0)]"
```

```

if {$cbr6np==0} {set cbr6np 1}

set co_sumcbr6 [expr $co_sumcbr6 + $cbr6by]
#average throughput in bps, 10 sec is session time
set co_avecbr6 [expr double($co_sumcbr6)/10]

#Reset the bytes_ values on the traffic sinks
#$sink5 set byte_ 0
#reset the nlost_ values to zero
#$sink0 set nlost_ 0
#reset the npkts_ values to zero
#$sink0 set npkts_ 0

#Re-Schedule The Procedure
$ns at [expr $now+$time] "record"
}

# define color index
$ns color 0 blue
$ns color 1 red
$ns color 2 chocolate
$ns color 3 red
$ns color 4 brown
$ns color 5 tan
$ns color 6 gold
$ns color 7 black

for {set i 0} {$i < $val(nn)} {incr i} {
    $ns initial_node_pos $node_($i) 30+i*100
}

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

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

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

```

\$node_(3) set X_ 0.0
\$node_(3) set Y_ 0.0
\$node_(3) set Z_ 0.0

\$node_(4) set X_ 0.0
\$node_(4) set Y_ 0.0
\$node_(4) set Z_ 0.0

\$node_(5) set X_ 0.0
\$node_(5) set Y_ 0.0
\$node_(5) set Z_ 0.0

\$node_(6) set X_ 0.0
\$node_(6) set Y_ 0.0
\$node_(6) set Z_ 0.0

\$node_(7) set X_ 0.0
\$node_(7) set Y_ 0.0
\$node_(7) set Z_ 0.0

\$ns at 0.0 "\$node_(0) setdest 100.0 100.0 3000.0"
\$ns at 0.0 "\$node_(1) setdest 200.0 200.0 3000.0"
\$ns at 0.0 "\$node_(2) setdest 300.0 200.0 3000.0"
\$ns at 0.0 "\$node_(3) setdest 400.0 300.0 3000.0"
\$ns at 0.0 "\$node_(4) setdest 500.0 400.0 3000.0"
\$ns at 0.0 "\$node_(5) setdest 100.0 280.0 3000.0"
\$ns at 0.0 "\$node_(6) setdest 550.0 200.0 3000.0"
\$ns at 0.0 "\$node_(7) setdest 600.0 300.0 3000.0"

\$ns at 1.5 "\$node_(5) setdest 100.0 400.0 500.0"
#\$ns at 1.0 "\$node_(3) setdest 400.0 400.0 500.0"
\$ns at 2.5 "\$node_(5) setdest 500.0 300.0 500.0"
\$ns at 3.5 "\$node_(5) setdest 500.0 100.0 500.0"
#\$ns at 4.0 "\$node_(3) setdest 600.0 400.0 500.0"
#\$ns at 3.5 "\$node_(5) setdest 300.0 100.0 500.0"
\$ns at 7.0 "\$node_(6) setdest 550.0 550.0 500.0"
\$ns at 8.0 "\$node_(6) setdest 100.0 400.0 500.0"

\$ns at 0.4 "\$node_(0) label \"source\""
\$ns at 0.4 "\$node_(5) label \"destination\""
#\$ns at 0.4 "\$node_(3) label \"destination\""


```

$ns at 1.9 "$node_(1) label \"helper\""
#$ns at 2.4 "$node_(3) label \" \"\"
$ns at 3.3 "$node_(1) label \" \"\"
$ns at 3.4 "$node_(2) label \"helper\""
$ns at 4.8 "$node_(5) label \" \"\"
#$ns at 4.5 "$node_(3) label \"destination\""
$ns at 6.2 "$node_(6) label \"destination\""
$ns at 6.8 "$node_(2) label \" \"\"
$ns at 7.0 "$node_(3) label \"helper\""
$ns at 7.9 "$node_(3) label \" \"\"
$ns at 8.0 "$node_(4) label \"helper\""
$ns at 9.0 "$node_(4) label \" \"\"
$ns at 9.1 "$node_(1) label \"helper\""

```

CONFIGURE AND SET UP A FLOW

```

set sink0 [new Agent/LossMonitor]
set sink1 [new Agent/LossMonitor]
set sink2 [new Agent/LossMonitor]
set sink3 [new Agent/LossMonitor]
set sink4 [new Agent/LossMonitor]
set sink5 [new Agent/LossMonitor]
set sink6 [new Agent/LossMonitor]
set sink7 [new Agent/LossMonitor]
$ns attach-agent $node_(0) $sink0
$ns attach-agent $node_(1) $sink1
$ns attach-agent $node_(2) $sink2
$ns attach-agent $node_(3) $sink3
$ns attach-agent $node_(4) $sink4
$ns attach-agent $node_(5) $sink5
$ns attach-agent $node_(6) $sink6
$ns attach-agent $node_(7) $sink7

```

```

#$ns attach-agent $sink2 $sink3
set tcp0 [new Agent/TCP]
$ns attach-agent $node_(0) $tcp0
set tcp1 [new Agent/TCP]
$ns attach-agent $node_(1) $tcp1
set tcp2 [new Agent/TCP]
$ns attach-agent $node_(2) $tcp2
set tcp3 [new Agent/TCP]
$ns attach-agent $node_(3) $tcp3

```

```

set tcp4 [new Agent/TCP]
$ns attach-agent $node_(4) $tcp4
set tcp5 [new Agent/TCP]
$ns attach-agent $node_(5) $tcp5
set tcp6 [new Agent/TCP]
$ns attach-agent $node_(6) $tcp6
set tcp7 [new Agent/TCP]
$ns attach-agent $node_(7) $tcp7

```

```

proc attach-CBR-traffic { node sink size interval } {
    #Get an instance of the simulator
    set ns [Simulator instance]
    #Create a CBR agent and attach it to the node
    set cbr [new Agent/CBR]
    $ns attach-agent $node $cbr
    $cbr set packetSize_ $size
    $cbr set interval_ $interval

    #Attach CBR source to sink;
    $ns connect $cbr $sink
    return $cbr
}

```

```

#src bitrate: 500*8/0.015=26.666 Kbps
set cbr0 [attach-CBR-traffic $node_(0) $sink5 500 .015]
set cbr1 [attach-CBR-traffic $node_(1) $sink2 500 .015]
set cbr2 [attach-CBR-traffic $node_(2) $sink3 500 .015]
set cbr3 [attach-CBR-traffic $node_(3) $sink0 500 .015]
set cbr4 [attach-CBR-traffic $node_(4) $sink3 500 .015]
set cbr5 [attach-CBR-traffic $node_(5) $sink0 500 .015]
set cbr6 [attach-CBR-traffic $node_(6) $sink0 500 .015]
set cbr7 [attach-CBR-traffic $node_(7) $sink0 500 .015]

```

```

$ns at 0.0 "record"
$ns at 0.5 "$cbr0 start"
$ns at 0.6 "$cbr2 start"
$ns at 2.0 "$cbr2 stop"
$ns at 4.6 "$cbr4 start"
$ns at 5.0 "$cbr0 stop"
$ns at 5.8 "$cbr4 stop"
$ns at 6.0 "$cbr0 stop"
$ns at 6.0 "$cbr6 start"
$ns at 9.0 "$cbr4 stop"

```

```
#$ns at 4.5 "$cbr2 start"  
#$ns at 6.0 "$cbr2 stop"  
#$ns at 5.4 "$cbr0 start"
```

```
$ns at 10.0 "finish"
```

```
puts "Start of simulation.."  
$ns run
```

802.11a.tcl and parameters

```
# =====  
# Define options  
# =====  
set val(chan) Channel/WirelessChannel ;# channel type  
set val(prop) Propagation/TwoRayGround ;# radio-propagation model  
set val(ant) Antenna/OmniAntenna ;# Antenna type  
set val(ll) LL ;# Link layer type  
set val(ifq) Queue/DropTail/PriQueue ;# Interface queue type  
set val(ifqlen) 50 ;# max packet in ifq  
set val(netif) Phy/WirelessPhy ;# network interface type  
set val(mac) Mac/802_11 ;# MAC type  
set val(nn) 8 ;# number of mobilenodes  
set val(rp) AODV ;# routing protocol  
set val(x) 650  
set val(y) 650  
  
#The Antenna height of transmitter and receiver is 1.5m.  
#The propagation model is TwoRayGround model.  
# set up the antennas to be centered in the node and 1.5 meters above it  
Antenna/OmniAntenna set X_ 0  
Antenna/OmniAntenna set Y_ 0  
Antenna/OmniAntenna set Z_ 1.5  
Antenna/OmniAntenna set Gt_ 1.0 ;# Transmit antenna gain  
Antenna/OmniAntenna set Gr_ 1.0 ;# Receive antenna gain  
  
#802.11a parameters  
  
Phy/WirelessPhyExt set CStresh_ 6.31e-12 ;#-82 dBm Wireless interface sensitivity  
;(sensitivity defined in the standard)  
Phy/WirelessPhyExt set Pt_ 0.001  
Phy/WirelessPhyExt set freq_ 5.18e+9
```

```

Phy/WirelessPhyExt set noise_floor_ 2.512e-13 ;#-96 dBm for 10MHz bandwidth
Phy/WirelessPhyExt set L_ 1.0 ;#default radio circuit gain/loss
Phy/WirelessPhyExt set PowerMonitorThresh_ 1.259e-13 ;#-99dBm power monitor
;#sensitivity
Phy/WirelessPhyExt set HeaderDuration_ 0.000020 ;#20 us
Phy/WirelessPhyExt set BasicModulationScheme_ 0
Phy/WirelessPhyExt set PreambleCaptureSwitch_ 1
Phy/WirelessPhyExt set DataCaptureSwitch_ 0
Phy/WirelessPhyExt set SINR_PreambleCapture_ 2.5118; ;# 4 dB
Phy/WirelessPhyExt set SINR_DataCapture_ 100.0; ;# 10 dB
Phy/WirelessPhyExt set trace_dist_ 1e6 ;# PHY trace until distance of 1 Mio. km
;#("infinty")
Phy/WirelessPhyExt set PHY_DBG_ 0

```

```

Mac/802_11Ext set CWMin_ 15
Mac/802_11Ext set CWMax_ 1023
Mac/802_11Ext set SlotTime_ 0.000009
Mac/802_11Ext set SIFS_ 0.000016
Mac/802_11Ext set ShortRetryLimit_ 7
Mac/802_11Ext set LongRetryLimit_ 4
Mac/802_11Ext set HeaderDuration_ 0.000020
Mac/802_11Ext set SymbolDuration_ 0.000004
Mac/802_11Ext set BasicModulationScheme_ 0
Mac/802_11Ext set use_802_11a_flag_ true
#Mac/802_11Ext set RTSThreshold_ 2346
Mac/802_11Ext set MAC_DBG_ 0

```

```

# fix for when RTS/CTS not used
#Mac/802_11 set RTS_ 20.0
#Mac/802_11 set CTS_ 14.0

```

```

#To create a simulator object. This can be done with the command:
set ns [new Simulator]

```

```

set f [open coop.tr w]
$ns trace-all $f

```

```

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

```

```

set f0 [open co_packet_received.data w]
set f1 [open co_packet_lost.data w]
set f2 [open co_expected_packet.data w]
set f3 [open byte.data w]
set f4 [open receivingTime.data w]

```

```

set f5 [open actual_Time.data w]

set f6 [open co_sumband.data w]
set f7 [open co_aveband.data w]
#set f8 [open sequence_number.data w]

set topo [new Topography]
$topo load_flatgrid 650 650

create-god $val(nn)

set chan_1 [new $val(chan)]
set chan_2 [new $val(chan)]
set chan_3 [new $val(chan)]
set chan_4 [new $val(chan)]
set chan_5 [new $val(chan)]
set chan_6 [new $val(chan)]
set chan_7 [new $val(chan)]
set chan_8 [new $val(chan)]

# CONFIGURE AND CREATE NODES

proc UniformErr {} {
set err [new ErrorModel]
$err unit packet
$err set rate_ 0.05
$err ranvar [new RandomVariable/Uniform]
$err drop-target [new Agent/Null]
return $err}

$ns node-config -adhocRouting $val(rp) \
    -llType $val(ll) \
        -macType $val(mac) \
        -ifqType $val(ifq) \
        -ifqLen $val(ifqlen) \
        -antType $val(ant) \
        -propType $val(prop) \
        -phyType $val(netif) \
    #-channelType $val(chan) \
    -topoInstance $topo \
    -agentTrace OFF \
    -routerTrace ON \
    -macTrace ON \
    -movementTrace OFF \
    -channel $chan_1 \

```

```

    #-energyModel EnergyModel \
    -IncomingErrProc UniformErr

for {set i 0} {$i < $val(nn)} {incr i} {
    set node_($i) [$ns node]
    $node_($i) random-motion 0 ;# disable random motion
}

proc finish {} {
    global ns f f0 f1 f2 f3 f4 f5 f6 f7 namtrace
    $ns flush-trace
    close $namtrace
    close $f0
    close $f1
    close $f2
    close $f3
    close $f4
    close $f5
    close $f6
    close $f7
    # close $f8

    exec xgraph co_packet_received.data co_packet_lost.data co_expected_packet.data &
    exec xgraph receivingTime.data actual_Time.data -geometry 500x300 &
    exec xgraph byte.data co_sumband.data -geometry 400x300 &
        exec nam -r 5m coop.nam &
    exit 0
}

set co_sumband 0
set co_aveband 0
proc record {} {
    global sink0 sink1 sink2 sink3 sink4 sink5 f0 f1 f2 f3 f4 f5 f6 f7 co_sumband
    co_aveband
    #Get An Instance Of The Simulator
    set ns [Simulator instance]

    #Set The Time After Which The Procedure Should Be Called Again
    set time 1.0
    #How Many Bytes Have Been Received By The Traffic Sinks?

    set bw0 [$sink5 set npkts_]
    set bw1 [$sink5 set nlost_]
    set bw [$sink5 set expected_]
    set byte [$sink5 set bytes_]
    set receivingTime [$sink5 set lastPktTime_]
}

```

```

#Get The Current Time
set now [$ns now]

#Save Data To The Files
puts $f0 "$now [expr $bw0]"
puts $f1 "$now [expr $bw1]"
puts $f2 "$now [expr $bw]"
puts $f3 "$now [expr $byte]"
puts $f4 "$now [expr $receivingTime]"

puts $f6 "$now [expr $co_sumband]"
puts $f7 "$now [expr $co_aveband]"
puts $f5 "$now [expr $now]"

#calculate loss ratio, to avoid divided-by-zero error
if {$bw0==0} {set bw0 1}
set co_sumband [expr $co_sumband + $byte]
#average throughput in bps, 10 sec is session time
set co_aveband [expr double($co_sumband)/10]
puts "T=$now, Band=$bw0, co_sumband= $co_sumband, co_aveband=$co_aveband"
#puts " At time =$now, Loss ratio [expr double($bw1)/double($bw1+$bw0)]"

#Reset the bytes_ values on the traffic sinks
  #$sink5 set byte_ 0
  #reset the nlost_ values to zero
  #$sink0 set nlost_ 0
  #reset the npkts_ values to zero
  #$sink0 set npkts_ 0

#Re-Schedule The Procedure
$ns at [expr $now+$time] "record"
}

# define color index
$ns color 0 blue
$ns color 1 red
$ns color 2 chocolate
$ns color 3 red
$ns color 4 brown
$ns color 5 tan

```

```
$ns color 6 gold
$ns color 7 black
```

```
for {set i 0} {$i < $val(mn)} {incr i} {
    $ns initial_node_pos $node_($i) 30+i*100
}
```

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

```
$node_(1) set X_ 0.0
$node_(1) set Y_ 0.0
$node_(1) set Z_ 0.0
```

```
$node_(2) set X_ 0.0
$node_(2) set Y_ 0.0
$node_(2) set Z_ 0.0
```

```
$node_(3) set X_ 0.0
$node_(3) set Y_ 0.0
$node_(3) set Z_ 0.0
```

```
$node_(4) set X_ 0.0
$node_(4) set Y_ 0.0
$node_(4) set Z_ 0.0
```

```
$node_(5) set X_ 0.0
$node_(5) set Y_ 0.0
$node_(5) set Z_ 0.0
```

```
$node_(6) set X_ 0.0
$node_(6) set Y_ 0.0
$node_(6) set Z_ 0.0
```

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

```
$ns at 0.0 "$node_(0) setdest 100.0 100.0 3000.0"
$ns at 0.0 "$node_(1) setdest 200.0 200.0 3000.0"
$ns at 0.0 "$node_(2) setdest 300.0 200.0 3000.0"
$ns at 0.0 "$node_(3) setdest 400.0 300.0 3000.0"
```



```
$ns at 0.0 "$node_(4) setdest 500.0 400.0 3000.0"
$ns at 0.0 "$node_(5) setdest 100.0 280.0 3000.0"
$ns at 0.0 "$node_(6) setdest 550.0 200.0 3000.0"
$ns at 0.0 "$node_(7) setdest 600.0 300.0 3000.0"
```

```
$ns at 1.5 "$node_(5) setdest 100.0 400.0 500.0"
#$ns at 1.0 "$node_(3) setdest 400.0 400.0 500.0"
$ns at 2.5 "$node_(5) setdest 500.0 300.0 500.0"
$ns at 3.5 "$node_(5) setdest 500.0 100.0 500.0"
#$ns at 4.0 "$node_(3) setdest 600.0 400.0 500.0"
#$ns at 3.5 "$node_(5) setdest 300.0 100.0 500.0"
$ns at 7.0 "$node_(6) setdest 550.0 550.0 500.0"
$ns at 8.0 "$node_(6) setdest 100.0 400.0 500.0"
```

```
$ns at 0.4 "$node_(0) label \"source\""
$ns at 0.4 "$node_(5) label \"destination\""
#$ns at 0.4 "$node_(3) label \"destination\""
$ns at 1.9 "$node_(1) label \"helper\""
#$ns at 2.4 "$node_(3) label \" \""
$ns at 3.3 "$node_(1) label \" \""
$ns at 3.4 "$node_(2) label \"helper\""
$ns at 4.8 "$node_(5) label \"\""
#$ns at 4.5 "$node_(3) label \"destination\""
$ns at 6.2 "$node_(6) label \"destination\""
$ns at 6.8 "$node_(2) label \"\""
$ns at 7.0 "$node_(3) label \"helper\""
$ns at 7.9 "$node_(3) label \"\""
$ns at 8.0 "$node_(4) label \"helper\""
$ns at 9.0 "$node_(4) label \"\""
$ns at 9.1 "$node_(1) label \"helper\""
```

CONFIGURE AND SET UP A FLOW

```
set sink0 [new Agent/LossMonitor]
set sink1 [new Agent/LossMonitor]
set sink2 [new Agent/LossMonitor]
set sink3 [new Agent/LossMonitor]
set sink4 [new Agent/LossMonitor]
set sink5 [new Agent/LossMonitor]
set sink6 [new Agent/LossMonitor]
set sink7 [new Agent/LossMonitor]
$ns attach-agent $node_(0) $sink0
$ns attach-agent $node_(1) $sink1
```

```
$ns attach-agent $node_(2) $sink2
$ns attach-agent $node_(3) $sink3
$ns attach-agent $node_(4) $sink4
$ns attach-agent $node_(5) $sink5
$ns attach-agent $node_(6) $sink6
$ns attach-agent $node_(7) $sink7
```

```
#$ns attach-agent $sink2 $sink3
set tcp0 [new Agent/TCP]
$ns attach-agent $node_(0) $tcp0
set tcp1 [new Agent/TCP]
$ns attach-agent $node_(1) $tcp1
set tcp2 [new Agent/TCP]
$ns attach-agent $node_(2) $tcp2
set tcp3 [new Agent/TCP]
$ns attach-agent $node_(3) $tcp3
set tcp4 [new Agent/TCP]
$ns attach-agent $node_(4) $tcp4
set tcp5 [new Agent/TCP]
$ns attach-agent $node_(5) $tcp5
set tcp6 [new Agent/TCP]
$ns attach-agent $node_(6) $tcp6
set tcp7 [new Agent/TCP]
$ns attach-agent $node_(7) $tcp7
```

```
proc attach-CBR-traffic { node sink size interval } {
    #Get an instance of the simulator
    set ns [Simulator instance]
    #Create a CBR agent and attach it to the node
    set cbr [new Agent/CBR]
    $ns attach-agent $node $cbr
    $cbr set packetSize_ $size
    $cbr set interval_ $interval

    #Attach CBR source to sink;
    $ns connect $cbr $sink
    return $cbr
}
```

```
#src bitrate: 500*8/0.015=26.666 Kbps
set cbr0 [attach-CBR-traffic $node_(0) $sink5 500 .015]
set cbr1 [attach-CBR-traffic $node_(1) $sink2 500 .015]
set cbr2 [attach-CBR-traffic $node_(2) $sink3 500 .015]
```

```

set cbr3 [attach-CBR-traffic $node_(3) $sink0 500 .015]
set cbr4 [attach-CBR-traffic $node_(4) $sink3 500 .015]
set cbr5 [attach-CBR-traffic $node_(5) $sink0 500 .015]
set cbr6 [attach-CBR-traffic $node_(6) $sink0 500 .015]
set cbr7 [attach-CBR-traffic $node_(7) $sink0 500 .015]

```

```

$ns at 0.0 "record"
$ns at 0.5 "$cbr0 start"
$ns at 0.6 "$cbr2 start"
$ns at 2.0 "$cbr2 stop"
$ns at 4.6 "$cbr4 start"
$ns at 5.0 "$cbr0 stop"
$ns at 5.8 "$cbr4 stop"
$ns at 6.0 "$cbr0 stop"
$ns at 6.0 "$cbr6 start"
$ns at 9.0 "$cbr4 stop"

```

```

#$ns at 4.5 "$cbr2 start"
#$ns at 6.0 "$cbr2 stop"
#$ns at 5.4 "$cbr0 start"

```

```

$ns at 10.0 "finish"

```

```

puts "Start of simulation.."
$ns run

```

802.11p.tcl and parameters

```

# =====
# Define options
# =====
set val(chan)      Channel/WirelessChannel ;# channel type
set val(prop)      Propagation/TwoRayGround ;# radio-propagation model
set val(ant)       Antenna/OmniAntenna    ;# Antenna type
set val(ll)        LL                      ;# Link layer type
set val(ifq)       Queue/DropTail/PriQueue ;# Interface queue type
set val(ifqlen)    50                      ;# max packet in ifq
set val(netif)     Phy/WirelessPhy        ;# network interface type
set val(mac)       Mac/802_11            ;# MAC type
set val(nn)        8                      ;# number of mobilenodes
set val(rp)        AODV                   ;# routing protocol
set val(x)         650
set val(y)         650

```

```

#The Antenna height of transmitter and receiver is 1.5m.
#The propagation model is TwoRayGround model.
# set up the antennas to be centered in the node and 1.5 meters above it
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
Antenna/OmniAntenna set Gt_ 1.0    ;# Transmit antenna gain
Antenna/OmniAntenna set Gr_ 1.0    ;# Receive antenna gain

#802.11p parameters

Phy/WirelessPhyExt set CStresh_ 3.162e-12    ;#-85 dBm Wireless interface
#sensitivity
                                ;#(sensitivity defined in the standard)
Phy/WirelessPhyExt set Pt_ 0.001
Phy/WirelessPhyExt set freq_ 5.9e+9
Phy/WirelessPhyExt set noise_floor_ 1.26e-13    ;#-99 dBm for 10MHz bandwidth
Phy/WirelessPhyExt set L_ 1.0    ;#default radio circuit gain/loss
Phy/WirelessPhyExt set PowerMonitorThresh_ 6.310e-14 ;#-102dBm power monitor
sensitivity
Phy/WirelessPhyExt set HeaderDuration_ 0.000040    ;#40 us
Phy/WirelessPhyExt set BasicModulationScheme_ 0
Phy/WirelessPhyExt set PreambleCaptureSwitch_ 1
Phy/WirelessPhyExt set DataCaptureSwitch_ 0
Phy/WirelessPhyExt set SINR_PreambleCapture_ 2.5118;    ;# 4 dB
Phy/WirelessPhyExt set SINR_DataCapture_ 100.0;    ;# 10 dB
Phy/WirelessPhyExt set trace_dist_ 1e6 ;# PHY trace until distance of 1 Mio. km
("infinty")
Phy/WirelessPhyExt set PHY_DBG_ 0

Mac/802_11Ext set CWMin_ 15
Mac/802_11Ext set CWMax_ 1023
Mac/802_11Ext set SlotTime_ 0.000013
Mac/802_11Ext set SIFS_ 0.000032
Mac/802_11Ext set ShortRetryLimit_ 7
Mac/802_11Ext set LongRetryLimit_ 4
Mac/802_11Ext set HeaderDuration_ 0.000040
Mac/802_11Ext set SymbolDuration_ 0.000008
Mac/802_11Ext set BasicModulationScheme_ 0
Mac/802_11Ext set use_802_11a_flag_ true
#Mac/802_11Ext set RTSThreshold_ 2346
Mac/802_11Ext set MAC_DBG_ 0

# fix for when RTS/CTS not used

```

```
#Mac/802_11 set RTS_ 20.0
#Mac/802_11 set CTS_ 14.0
```

```
#To create a simulator object. This can be done with the command:
set ns [new Simulator]
```

```
set f [open coop.tr w]
$ns trace-all $f
```

```
set namtrace [open coop.nam w]
$ns namtrace-all-wireless $namtrace $val(x) $val(y)
```

```
set f0 [open co_packet_received.data w]
set f1 [open co_packet_lost.data w]
set f2 [open co_expected_packet.data w]
set f3 [open byte.data w]
set f4 [open receivingTime.data w]
set f5 [open actual_Time.data w]
```

```
set f6 [open co_sumband.data w]
set f7 [open co_aveband.data w]
#set f8 [open sequence_number.data w]
```

```
set topo [new Topography]
$topo load_flatgrid 650 650
```

```
create-god $val(nn)
```

```
set chan_1 [new $val(chan)]
set chan_2 [new $val(chan)]
set chan_3 [new $val(chan)]
set chan_4 [new $val(chan)]
set chan_5 [new $val(chan)]
set chan_6 [new $val(chan)]
set chan_7 [new $val(chan)]
set chan_8 [new $val(chan)]
```

```
# CONFIGURE AND CREATE NODES
```

```
proc UniformErr {} {
set err [new ErrorModel]
$err unit packet
$err set rate_ 0.05
$err ranvar [new RandomVariable/Uniform]
$err drop-target [new Agent/Null]
```

```

return $err}

$ns node-config -adhocRouting $val(rp) \
  -llType $val(ll) \
    -macType $val(mac) \
    -ifqType $val(ifq) \
    -ifqLen $val(ifqlen) \
    -antType $val(ant) \
    -propType $val(prop) \
    -phyType $val(netif) \
    #-channelType $val(chan) \
    -topoInstance $topo \
    -agentTrace OFF \
    -routerTrace ON \
    -macTrace ON \
    -movementTrace OFF \
    -channel $chan_1 \
    #-energyModel EnergyModel \
    -IncomingErrProc UniformErr

for {set i 0} {$i < $val(nn)} {incr i} {
  set node_($i) [$ns node]
  $node_($i) random-motion 0 ;# disable random motion
}

proc finish {} {
  global ns f f0 f1 f2 f3 f4 f5 f6 f7 namtrace
  $ns flush-trace
  close $namtrace
  close $f0
  close $f1
  close $f2
  close $f3
  close $f4
  close $f5
  close $f6
  close $f7
  # close $f8

  exec xgraph co_packet_received.data co_packet_lost.data co_expected_packet.data &
  exec xgraph receivingTime.data actual_Time.data -geometry 500x300 &
  exec xgraph byte.data co_sumband.data -geometry 400x300 &
  exec nam -r 5m coop.nam &
  exit 0
}
set co_sumband 0

```

```

set co_aveband 0
proc record { } {
    global sink0 sink1 sink2 sink3 sink4 sink5 f0 f1 f2 f3 f4 f5 f6 f7 co_sumband
co_aveband
    #Get An Instance Of The Simulator
    set ns [Simulator instance]

    #Set The Time After Which The Procedure Should Be Called Again
    set time 1.0
    #How Many Bytes Have Been Received By The Traffic Sinks?

    set bw0 [$sink5 set npkts_]
    set bw1 [$sink5 set nlost_]
    set bw [$sink5 set expected_]
    set byte [$sink5 set bytes_]
    set receivingTime [$sink5 set lastPktTime_]

    #Get The Current Time
    set now [$ns now]

    #Save Data To The Files
    puts $f0 "$now [expr $bw0]"
    puts $f1 "$now [expr $bw1]"
    puts $f2 "$now [expr $bw]"
    puts $f3 "$now [expr $byte]"
    puts $f4 "$now [expr $receivingTime]"

    puts $f6 "$now [expr $co_sumband]"
    puts $f7 "$now [expr $co_aveband]"
    puts $f5 "$now [expr $now]"

    #calculate loss ratio, to avoid divided-by-zero error
    if {$bw0==0} {set bw0 1}
    set co_sumband [expr $co_sumband + $byte]
    #average throughput in bps, 10 sec is session time
    set co_aveband [expr double($co_sumband)/10]
    puts "T=$now, Band=$bw0, co_sumband= $co_sumband, co_aveband=$co_aveband"
    #puts " At time =$now, Loss ratio [expr double($bw1)/double($bw1+$bw0)]"

    #Reset the bytes_ values on the traffic sinks
    #sink5 set byte_ 0
    #reset the nlost_ values to zero

```

```
#$sink0 set nlost_ 0
#reset the npkts_ values to zero
#$sink0 set npkts_ 0
```

```
#Re-Schedule The Procedure
$ns at [expr $now+$time] "record"
}
```

```
# define color index
$ns color 0 blue
$ns color 1 red
$ns color 2 chocolate
$ns color 3 red
$ns color 4 brown
$ns color 5 tan
$ns color 6 gold
$ns color 7 black
```

```
for {set i 0} {$i < $val(nn)} {incr i} {
    $ns initial_node_pos $node_($i) 30+i*100
}
```

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

```
$node_(1) set X_ 0.0
$node_(1) set Y_ 0.0
$node_(1) set Z_ 0.0
```

```
$node_(2) set X_ 0.0
$node_(2) set Y_ 0.0
$node_(2) set Z_ 0.0
```

```
$node_(3) set X_ 0.0
$node_(3) set Y_ 0.0
$node_(3) set Z_ 0.0
```

```
$node_(4) set X_ 0.0
$node_(4) set Y_ 0.0
$node_(4) set Z_ 0.0
```


\$node_(5) set X_ 0.0
\$node_(5) set Y_ 0.0
\$node_(5) set Z_ 0.0

\$node_(6) set X_ 0.0
\$node_(6) set Y_ 0.0
\$node_(6) set Z_ 0.0

\$node_(7) set X_ 0.0
\$node_(7) set Y_ 0.0
\$node_(7) set Z_ 0.0

\$ns at 0.0 "\$node_(0) setdest 100.0 100.0 3000.0"
\$ns at 0.0 "\$node_(1) setdest 200.0 200.0 3000.0"
\$ns at 0.0 "\$node_(2) setdest 300.0 200.0 3000.0"
\$ns at 0.0 "\$node_(3) setdest 400.0 300.0 3000.0"
\$ns at 0.0 "\$node_(4) setdest 500.0 400.0 3000.0"
\$ns at 0.0 "\$node_(5) setdest 100.0 280.0 3000.0"
\$ns at 0.0 "\$node_(6) setdest 550.0 200.0 3000.0"
\$ns at 0.0 "\$node_(7) setdest 600.0 300.0 3000.0"

\$ns at 1.5 "\$node_(5) setdest 100.0 400.0 500.0"
#\$ns at 1.0 "\$node_(3) setdest 400.0 400.0 500.0"
\$ns at 2.5 "\$node_(5) setdest 500.0 300.0 500.0"
\$ns at 3.5 "\$node_(5) setdest 500.0 100.0 500.0"
#\$ns at 4.0 "\$node_(3) setdest 600.0 400.0 500.0"
#\$ns at 3.5 "\$node_(5) setdest 300.0 100.0 500.0"
\$ns at 7.0 "\$node_(6) setdest 550.0 550.0 500.0"
\$ns at 8.0 "\$node_(6) setdest 100.0 400.0 500.0"

\$ns at 0.4 "\$node_(0) label \"source\""
\$ns at 0.4 "\$node_(5) label \"destination\""
#\$ns at 0.4 "\$node_(3) label \"destination\""
\$ns at 1.9 "\$node_(1) label \"helper\""
#\$ns at 2.4 "\$node_(3) label \" \""
\$ns at 3.3 "\$node_(1) label \" \""
\$ns at 3.4 "\$node_(2) label \"helper\""
\$ns at 4.8 "\$node_(5) label \" \""
#\$ns at 4.5 "\$node_(3) label \"destination\""
\$ns at 6.2 "\$node_(6) label \"destination\""
\$ns at 6.8 "\$node_(2) label \" \""
\$ns at 7.0 "\$node_(3) label \"helper\""
\$ns at 7.9 "\$node_(3) label \" \""

```
$ns at 8.0 "$node_(4) label \"helper\""  
$ns at 9.0 "$node_(4) label \"\""  
$ns at 9.1 "$node_(1) label \"helper\""
```

CONFIGURE AND SET UP A FLOW

```
set sink0 [new Agent/LossMonitor]  
set sink1 [new Agent/LossMonitor]  
set sink2 [new Agent/LossMonitor]  
set sink3 [new Agent/LossMonitor]  
set sink4 [new Agent/LossMonitor]  
set sink5 [new Agent/LossMonitor]  
set sink6 [new Agent/LossMonitor]  
set sink7 [new Agent/LossMonitor]  
$ns attach-agent $node_(0) $sink0  
$ns attach-agent $node_(1) $sink1  
$ns attach-agent $node_(2) $sink2  
$ns attach-agent $node_(3) $sink3  
$ns attach-agent $node_(4) $sink4  
$ns attach-agent $node_(5) $sink5  
$ns attach-agent $node_(6) $sink6  
$ns attach-agent $node_(7) $sink7
```

```
#$ns attach-agent $sink2 $sink3  
set tcp0 [new Agent/TCP]  
$ns attach-agent $node_(0) $tcp0  
set tcp1 [new Agent/TCP]  
$ns attach-agent $node_(1) $tcp1  
set tcp2 [new Agent/TCP]  
$ns attach-agent $node_(2) $tcp2  
set tcp3 [new Agent/TCP]  
$ns attach-agent $node_(3) $tcp3  
set tcp4 [new Agent/TCP]  
$ns attach-agent $node_(4) $tcp4  
set tcp5 [new Agent/TCP]  
$ns attach-agent $node_(5) $tcp5  
set tcp6 [new Agent/TCP]  
$ns attach-agent $node_(6) $tcp6  
set tcp7 [new Agent/TCP]  
$ns attach-agent $node_(7) $tcp7
```

```

proc attach-CBR-traffic { node sink size interval } {
    #Get an instance of the simulator
    set ns [Simulator instance]
    #Create a CBR agent and attach it to the node
    set cbr [new Agent/CBR]
    $ns attach-agent $node $cbr
    $cbr set packetSize_ $size
    $cbr set interval_ $interval

    #Attach CBR source to sink;
    $ns connect $cbr $sink
    return $cbr
}

#src bitrate: 500*8/0.015=26.666 Kbps
set cbr0 [attach-CBR-traffic $node_(0) $sink5 500 .015]
set cbr1 [attach-CBR-traffic $node_(1) $sink2 500 .015]
set cbr2 [attach-CBR-traffic $node_(2) $sink3 500 .015]
set cbr3 [attach-CBR-traffic $node_(3) $sink0 500 .015]
set cbr4 [attach-CBR-traffic $node_(4) $sink3 500 .015]
set cbr5 [attach-CBR-traffic $node_(5) $sink0 500 .015]
set cbr6 [attach-CBR-traffic $node_(6) $sink0 500 .015]
set cbr7 [attach-CBR-traffic $node_(7) $sink0 500 .015]

$ns at 0.0 "record"
$ns at 0.5 "$cbr0 start"
$ns at 0.6 "$cbr2 start"
$ns at 2.0 "$cbr2 stop"
$ns at 4.6 "$cbr4 start"
$ns at 5.0 "$cbr0 stop"
$ns at 5.8 "$cbr4 stop"
$ns at 6.0 "$cbr0 stop"
$ns at 6.0 "$cbr6 start"
$ns at 9.0 "$cbr4 stop"

#$ns at 4.5 "$cbr2 start"
#$ns at 6.0 "$cbr2 stop"
#$ns at 5.4 "$cbr0 start"

$ns at 10.0 "finish"

puts "Start of simulation.."
$ns run

```

coExt.tcl and parameters

```
# =====
# Define options
# =====
set val(chan)    Channel/WirelessChannel ;# channel type
set val(prop)    Propagation/TwoRayGround ;# radio-propagation model
set val(ant)     Antenna/OmniAntenna    ;# Antenna type
set val(ll)      LL                      ;# Link layer type
set val(ifq)     Queue/DropTail/PriQueue ;# Interface queue type
set val(ifqlen)  50                      ;# max packet in ifq
set val(netif)   Phy/WirelessPhy        ;# network interface type
set val(mac)     Mac/802_11             ;# MAC type
set val(nn)      8                      ;# number of mobilenodes
set val(rp)      AODV                   ;# routing protocol
set val(x)       650
set val(y)       650

#The Antenna height of transmitter and receiver is 1.5m.
#The propagation model is TwoRayGround model.
# set up the antennas to be centered in the node and 1.5 meters above it
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
Antenna/OmniAntenna set Gt_ 1.0 ;# Transmit antenna gain
Antenna/OmniAntenna set Gr_ 1.0 ;# Receive antenna gain

Phy/WirelessPhyExt set bandwidth_ 54e6 ;#Data Rate

# Initialize the SharedMedia interface with parameters to make
# it work like the 914MHz Lucent WaveLAN DSSS radio interface
#Collision Threshold
Phy/WirelessPhyExt set CPTresh_ 10.0
#Receive Power Threshold;calculated under TwoRayGround model by tools from NS2
Phy/WirelessPhyExt set RXThresh_ 3.652e-10
#Transmit Power
Phy/WirelessPhyExt set Pt_ 0.28183815
# Channel 2.4 GHz
Phy/WirelessPhyExt set freq_ 2.4e9
```

```

#System loss facor
Phy/WirelessPhyExt set L_ 1.0
#Carrier Sense Power
Phy/WirelessPhyExt set CStresh_ 1.559e-11

Phy/WirelessPhyExt set PowerMonitorThresh_ 6.310e-14 ;#-102dBm power monitor
sensitivity
Phy/WirelessPhyExt set HeaderDuration_ 0.000040 ;#40 us
Phy/WirelessPhyExt set BasicModulationScheme_ 0
Phy/WirelessPhyExt set PreambleCaptureSwitch_ 1
Phy/WirelessPhyExt set DataCaptureSwitch_ 0
Phy/WirelessPhyExt set SINR_PreambleCapture_ 2.5118; ;# 4 dB
Phy/WirelessPhyExt set SINR_DataCapture_ 100.0; ;# 10 dB
Phy/WirelessPhyExt set trace_dist_ 1e6 ;# PHY trace until distance of 1
Mio.km("infinty")
Phy/WirelessPhyExt set PHY_DBG_ 0
Phy/WirelessPhyExt set noise_floor_ 1.26e-13 ;#-99 dBm for 10MHz bandwidth

#you can set dataRate for DATA here
Mac/802_11Ext set dataRate_ 54e6 ;# Rate for Data frame
#you can set basicRate for RTS/CTS, and ACK here
Mac/802_11Ext set basicRate_ 6e6 ;# Rate for Control Frame

#Mac/802_11 set RTSThreshold_ 3000 ;# Disable RTS/CTS
# 802.11g parameters

Mac/802_11Ext set CWMin_ 15
Mac/802_11Ext set CWMax_ 1023
Mac/802_11Ext set SlotTime_ 0.000009 ;# 9us
Mac/802_11Ext set CCATime_ 0.000003
Mac/802_11Ext set RxTxTurnaroundTime_ 0.000002
Mac/802_11Ext set SIFSTime_ 0.000016 ;# 16us
Mac/802_11Ext set DIFS_ 0.000028 ;# 50us
Mac/802_11Ext set PreambleLength_ 96 ;# 96 bit
Mac/802_11Ext set PLCPHeaderLength_ 40 ;# 40 bits
Mac/802_11Ext set PLCPDataRate_ 6.0e6 ;# 6Mbps
Mac/802_11Ext set MaxPropagationDelay_ 0.0000005 ;# 0.5us

Mac/802_11Ext set ShortRetryLimit_ 7
Mac/802_11Ext set LongRetryLimit_ 4
Mac/802_11Ext set HeaderDuration_ 0.000040
Mac/802_11Ext set SymbolDuration_ 0.000008

```

```
Mac/802_11Ext set BasicModulationScheme_ 0
Mac/802_11Ext set use_802_11a_flag_ true
#Mac/802_11Ext set RTSThreshold_ 2346
Mac/802_11Ext set MAC_DBG_ 0
```

```
# fix for when RTS/CTS not used
#Mac/802_11 set RTS_ 20.0
#Mac/802_11 set CTS_ 14.0
```

```
#To create a simulator object. This can be done with the command:
```

```
set ns [new Simulator]
```

```
set f [open coop.tr w]
$ns trace-all $f
```

```
set namtrace [open coop.nam w]
$ns namtrace-all-wireless $namtrace $val(x) $val(y)
```

```
set f0 [open co_packet_received.data w]
set f1 [open co_packet_lost.data w]
set f2 [open co_expected_packet.data w]
set f3 [open byte.data w]
set f4 [open receivingTime.data w]
set f5 [open actual_Time.data w]
```

```
set f6 [open co_sumband.data w]
set f7 [open co_aveband.data w]
#set f8 [open sequence_number.data w]
```

```
set topo [new Topography]
$topo load_flatgrid 650 650
```

```
create-god $val(nn)
```

```
set chan_1 [new $val(chan)]
set chan_2 [new $val(chan)]
set chan_3 [new $val(chan)]
set chan_4 [new $val(chan)]
set chan_5 [new $val(chan)]
set chan_6 [new $val(chan)]
set chan_7 [new $val(chan)]
set chan_8 [new $val(chan)]
```

```
# CONFIGURE AND CREATE NODES
```

```

proc UniformErr {} {
set err [new ErrorModel]
Serr unit packet
Serr set rate_ 0.05
Serr ranvar [new RandomVariable/Uniform]
Serr drop-target [new Agent/Null]
return $err}

$ns node-config -adhocRouting $val(rp) \
  -llType $val(ll) \
    -macType $val(mac) \
    -ifqType $val(ifq) \
    -ifqLen $val(ifqlen) \
    -antType $val(ant) \
    -propType $val(prop) \
    -phyType $val(netif) \
    #-channelType $val(chan) \
    -topoInstance $topo \
    -agentTrace OFF \
    -routerTrace ON \
    -macTrace ON \
    -movementTrace OFF \
    -channel $chan_1 \
    #-energyModel EnergyModel \
    -IncomingErrProc UniformErr

for {set i 0} {$i < $val(nn)} {incr i} {
  set node_($i) [$ns node]
  $node_($i) random-motion 0 ;# disable random motion
}

proc finish {} {
  global ns f f0 f1 f2 f3 f4 f5 f6 f7 namtrace
  $ns flush-trace
  close $namtrace
  close $f0
  close $f1
  close $f2
  close $f3
  close $f4
  close $f5
  close $f6
  close $f7
  # close $f8

```

```

exec xgraph co_packet_received.data co_packet_lost.data co_expected_packet.data &
exec xgraph receivingTime.data actual_Time.data -geometry 500x300 &
exec xgraph byte.data co_sumband.data -geometry 400x300 &
    exec nam -r 5m coop.nam &
    exit 0
}
set co_sumband 0
set co_aveband 0
proc record { } {
    global sink0 sink1 sink2 sink3 sink4 sink5 f0 f1 f2 f3 f4 f5 f6 f7 co_sumband
co_aveband
    #Get An Instance Of The Simulator
    set ns [Simulator instance]

    #Set The Time After Which The Procedure Should Be Called Again
    set time 1.0
    #How Many Bytes Have Been Received By The Traffic Sinks?

    set bw0 [$sink5 set npkts_]
    set bw1 [$sink5 set nlost_]
    set bw [$sink5 set expected_]
    set byte [$sink5 set bytes_]
    set receivingTime [$sink5 set lastPktTime_]

    #Get The Current Time
    set now [$ns now]

    #Save Data To The Files
    puts $f0 "$now [expr $bw0]"
    puts $f1 "$now [expr $bw1]"
    puts $f2 "$now [expr $bw]"
    puts $f3 "$now [expr $byte]"
    puts $f4 "$now [expr $receivingTime]"

    puts $f6 "$now [expr $co_sumband]"
    puts $f7 "$now [expr $co_aveband]"
    puts $f5 "$now [expr $now]"

    #calculate loss ratio, to avoid divided-by-zero error
    if {$bw0==0} {set bw0 1}
    set co_sumband [expr $co_sumband + $byte]
    #average throughput in bps, 10 sec is session time
    set co_aveband [expr double($co_sumband)/10]

```



```
puts "T=$now, Band=$bw0, co_sumband= $co_sumband, co_aveband=$co_aveband"  
#puts " At time =$now, Loss ratio [expr double($bw1)/double($bw1+$bw0)]"
```

```
#Reset the bytes_ values on the traffic sinks
```

```
  #sink5 set byte_ 0
```

```
  #reset the nlost_ values to zero
```

```
  #sink0 set nlost_ 0
```

```
  #reset the npkts_ values to zero
```

```
  #sink0 set npkts_ 0
```

```
#Re-Schedule The Procedure
```

```
$ns at [expr $now+$time] "record"
```

```
}
```

```
# define color index
```

```
$ns color 0 blue
```

```
$ns color 1 red
```

```
$ns color 2 chocolate
```

```
$ns color 3 red
```

```
$ns color 4 brown
```

```
$ns color 5 tan
```

```
$ns color 6 gold
```

```
$ns color 7 black
```

```
for {set i 0} {$i < $val(nn)} {incr i} {
```

```
  $ns initial_node_pos $node_($i) 30+i*100
```

```
}
```

```
$node_(0) set X_ 0.0
```

```
$node_(0) set Y_ 0.0
```

```
$node_(0) set Z_ 0.0
```

```
$node_(1) set X_ 0.0
```

```
$node_(1) set Y_ 0.0
```

```
$node_(1) set Z_ 0.0
```

```
$node_(2) set X_ 0.0
```

```
$node_(2) set Y_ 0.0
```

```
$node_(2) set Z_ 0.0
```

```
$node_(3) set X_ 0.0
```

```
$node_(3) set Y_ 0.0
```

\$node_(3) set Z_ 0.0

\$node_(4) set X_ 0.0

\$node_(4) set Y_ 0.0

\$node_(4) set Z_ 0.0

\$node_(5) set X_ 0.0

\$node_(5) set Y_ 0.0

\$node_(5) set Z_ 0.0

\$node_(6) set X_ 0.0

\$node_(6) set Y_ 0.0

\$node_(6) set Z_ 0.0

\$node_(7) set X_ 0.0

\$node_(7) set Y_ 0.0

\$node_(7) set Z_ 0.0

\$ns at 0.0 "\$node_(0) setdest 100.0 100.0 3000.0"

\$ns at 0.0 "\$node_(1) setdest 200.0 200.0 3000.0"

\$ns at 0.0 "\$node_(2) setdest 300.0 200.0 3000.0"

\$ns at 0.0 "\$node_(3) setdest 400.0 300.0 3000.0"

\$ns at 0.0 "\$node_(4) setdest 500.0 400.0 3000.0"

\$ns at 0.0 "\$node_(5) setdest 100.0 280.0 3000.0"

\$ns at 0.0 "\$node_(6) setdest 550.0 200.0 3000.0"

\$ns at 0.0 "\$node_(7) setdest 600.0 300.0 3000.0"

\$ns at 1.5 "\$node_(5) setdest 100.0 400.0 500.0"

#\$ns at 1.0 "\$node_(3) setdest 400.0 400.0 500.0"

\$ns at 2.5 "\$node_(5) setdest 500.0 300.0 500.0"

\$ns at 3.5 "\$node_(5) setdest 500.0 100.0 500.0"

#\$ns at 4.0 "\$node_(3) setdest 600.0 400.0 500.0"

#\$ns at 3.5 "\$node_(5) setdest 300.0 100.0 500.0"

\$ns at 7.0 "\$node_(6) setdest 550.0 550.0 500.0"

\$ns at 8.0 "\$node_(6) setdest 100.0 400.0 500.0"

\$ns at 0.4 "\$node_(0) label \"source\""

\$ns at 0.4 "\$node_(5) label \"destination\""

#\$ns at 0.4 "\$node_(3) label \"destination\""

\$ns at 1.9 "\$node_(1) label \"helper\""

#\$ns at 2.4 "\$node_(3) label \" \""

\$ns at 3.3 "\$node_(1) label \" \""

```

$ns at 3.4 "$node_(2) label \"helper\""
$ns at 4.8 "$node_(5) label \"\""
#$ns at 4.5 "$node_(3) label \"destination\""
$ns at 6.2 "$node_(6) label \"destination\""
$ns at 6.8 "$node_(2) label \"\""
$ns at 7.0 "$node_(3) label \"helper\""
$ns at 7.9 "$node_(3) label \"\""
$ns at 8.0 "$node_(4) label \"helper\""
$ns at 9.0 "$node_(4) label \"\""
$ns at 9.1 "$node_(1) label \"helper\""

```

CONFIGURE AND SET UP A FLOW

```

set sink0 [new Agent/LossMonitor]
set sink1 [new Agent/LossMonitor]
set sink2 [new Agent/LossMonitor]
set sink3 [new Agent/LossMonitor]
set sink4 [new Agent/LossMonitor]
set sink5 [new Agent/LossMonitor]
set sink6 [new Agent/LossMonitor]
set sink7 [new Agent/LossMonitor]
$ns attach-agent $node_(0) $sink0
$ns attach-agent $node_(1) $sink1
$ns attach-agent $node_(2) $sink2
$ns attach-agent $node_(3) $sink3
$ns attach-agent $node_(4) $sink4
$ns attach-agent $node_(5) $sink5
$ns attach-agent $node_(6) $sink6
$ns attach-agent $node_(7) $sink7

```

```

#$ns attach-agent $sink2 $sink3
set tcp0 [new Agent/TCP]
$ns attach-agent $node_(0) $tcp0
set tcp1 [new Agent/TCP]
$ns attach-agent $node_(1) $tcp1
set tcp2 [new Agent/TCP]
$ns attach-agent $node_(2) $tcp2
set tcp3 [new Agent/TCP]
$ns attach-agent $node_(3) $tcp3
set tcp4 [new Agent/TCP]
$ns attach-agent $node_(4) $tcp4
set tcp5 [new Agent/TCP]

```

```

$ns attach-agent $node_(5) $tcp5
set tcp6 [new Agent/TCP]
$ns attach-agent $node_(6) $tcp6
set tcp7 [new Agent/TCP]
$ns attach-agent $node_(7) $tcp7

```

```

proc attach-CBR-traffic { node sink size interval } {
    #Get an instance of the simulator
    set ns [Simulator instance]
    #Create a CBR agent and attach it to the node
    set cbr [new Agent/CBR]
    $ns attach-agent $node $cbr
    $cbr set packetSize_ $size
    $cbr set interval_ $interval

    #Attach CBR source to sink;
    $ns connect $cbr $sink
    return $cbr
}

```

```

#src bitrate: 500*8/0.015=26.666 Kbps
set cbr0 [attach-CBR-traffic $node_(0) $sink5 500 .015]
set cbr1 [attach-CBR-traffic $node_(1) $sink2 500 .015]
set cbr2 [attach-CBR-traffic $node_(2) $sink3 500 .015]
set cbr3 [attach-CBR-traffic $node_(3) $sink0 500 .015]
set cbr4 [attach-CBR-traffic $node_(4) $sink3 500 .015]
set cbr5 [attach-CBR-traffic $node_(5) $sink0 500 .015]
set cbr6 [attach-CBR-traffic $node_(6) $sink0 500 .015]
set cbr7 [attach-CBR-traffic $node_(7) $sink0 500 .015]

```

```

$ns at 0.0 "record"
$ns at 0.5 "$cbr0 start"
$ns at 0.6 "$cbr2 start"
$ns at 2.0 "$cbr2 stop"
$ns at 4.6 "$cbr4 start"
$ns at 5.0 "$cbr0 stop"
$ns at 5.8 "$cbr4 stop"
$ns at 6.0 "$cbr0 stop"
$ns at 6.0 "$cbr6 start"
$ns at 9.0 "$cbr4 stop"

```

```

#$ns at 4.5 "$cbr2 start"
#$ns at 6.0 "$cbr2 stop"
#$ns at 5.4 "$cbr0 start"

```

\$ns at 10.0 "finish"

puts "Start of simulation.."

\$ns run

Appendix B

IEEE 802.11 Standard

The IEEE 802.11 working group has issued and expanding list of standards, as shown in Table 1.1.

Standard	scope
IEEE 802.11	Medium access control (MAC): One common MAC for WLAN applications
	Physical layer: Infrared at 1 and 2 Mbps
	Physical layer: 2.4-GHz FHSS at 1 and 2 Mbps
	Physical layer: 2.4-GHz DSSS at 1 and 2Mbps
IEEE 802.11a	Physical layer: 5-GHz OFDM at rates from 6 to 54 Mbps
IEEE 802.11b	Physical layer: 2.4-GHz DSSS at 5.5 and 11 Mbps
IEEE 802.11c	Bridge operation at 802.11 MAC layer
IEEE 802.11d	Physical layer: Extend operation of 802.11 WLANs to new regulatory do mains (countries)
IEEE 802.11e	MAC: Enhance to improve quality of service and enhance security mechanisms
IEEE 802.11f	Recommended practices for multivendor access point interoperability
IEEE 802.11g	Physical layer : Extend 802.11b to data rates > 20 Mbps
IEEE 802.11h	Physical/MAC: Enhance IEEE 802.11a to add indoor and outdoor channel selection and to improve spectrum and transmit power management
IEEE 802.11i	MAC: Enhance security and authentication mechanisms
IEEE 802.11j	Physical : Enhance IEEE 802.11a to conform to Japanese requirements
IEEE 802.11 k	Radio resource measurement enhancements to provide interface to higher layers for radio and network measurements
IEEE 802.11 m	Maintenance of IEEE 802.11-1999 standard with technical and editorial corrections
IEEE 802.11 n	Physical/MAC: Enhancements to enable higher throughput
IEEE 802.11 p	Physical/MAC: Wireless access in vehicular environments
IEEE 802.11 r	Physical / MAC : Fast roaming (fast BSS transition)
IEEE 802.11 s	Physical / MAC : ESS mesh networking
IEEE 802.11 , 2	Recommended practice for the Evaluation of 802.11 wireless performance
IEEE 802.11 u	Physical / MAC : Interworking with external networks

Table 1.1 IEEE 802.11 Standards

A briefly defines key terms used in the IEEE 802.11 standard

Access point (AP)	Any entity that has station functionality and provides access to the distribution system via the wireless medium for associated stations
Basic service set (BSS)	A set of stations controlled by a single coordination function
Coordination function	The logical function that determines when a station operating within a BSS is permitted to transmit and may be able to receive PDUs
Distribution system (DS)	A system used to interconnect a set of BSSs and integrated LANs to create an ESS
Extended service set (ESS)	A set of one or more interconnected BSSs and integrated LANs that appear as a single BSS to the LLC layer at any station associated with one of these BSSs
MAC protocol data unit (MPDU)	The unit of data exchanged between two peer MAC entities using the services of the physical layer
MAC service data unit (MSDU)	Information that is delivered as a unit between MAC users
Station	Any device that contains an IEEE 802.11 conformant MAC and physical layer

Table 1.2 A briefly defines key terms used in the IEEE 802.11 standard

IEEE 802 Services

IEEE 802.11 defines nine services that need to be provided by the wireless LAN

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

Table 1.3 IEEE 802.11 define nine services

9.References

[1] The TCP / IP guide: Table of contents:

http://www.tcpipguide.com/free/t_toc.htm

[1] The TCP / IP guide: TCP/IP Overview and history:

http://www.tcpipguide.com/free/t_TCPIPOverviewandHistory.htm

[1] The TCP / IP: Architecture and the TCP/IP model : (TCP /IP architecture)

http://www.tcpipguide.com/free/t_TCPIPArchitectureandtheTCPIPModel.htm

[1] TCP/IP guide: Data link layer: (data link)

http://www.tcpipguide.com/free/t_DataLinkLayerLayer2.htm

[1] The TCP /IP protocol

http://www.tcpipguide.com/free/t_TCPIPProtocols.htm

[1] TCP/IP Architecture and the TCP/ IP Model

http://www.tcpipguide.com/free/t_TCPIPArchitectureandtheTCPIPModel-2.htm

[2] Medium Access Control (MAC)

<http://www.erg.abdn.ac.uk/users/gorry/eg3561/lan-pages/mac.html>

[2] Multiple Access Protocol: Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

<http://www.erg.abdn.ac.uk/users/gorry/eg3561/lan-pages/csma-cd.html>

[3] Networks and distributed systems: medium Access control Protocol in cognitive radio networks

<http://simula.no/research/nd/publications/Simula.ND.383>

<http://simula.no/research/nd>

[4] About.com: Wireless/ networking

<http://compnetworking.about.com/od/networkprotocolsip/l/aa062202a.htm>

[5]Wi-Fi: Planet: Tutorials: 802.11 Medium Access Methods: Distributed coordination function: PDF Protocol Issues: Point Coordination function

<http://www.wi-fiplanet.com/tutorials/article.php/1548381>

[5]Wi-Fi: Planet: Tutorials: Improving WLAN performance with RTS/CTS

<http://www.wi-fiplanet.com/tutorials/article.php/1445641>

[5]Wi-Fi: Planet: Tutorials: Antenna: The key to Maximizing RF coverage:

<http://www.wi-fiplanet.com/tutorials/article.php/1144391>

[5]Wi-Fi: Planet:Infrared WLAN: (Infrared LAN)

<http://www.wi-fiplanet.com/tutorials/article.php/2110301>

[6] WIMAX MADE SIMPLE

<http://wimax-made-simple.blogspot.com/2009/10/medium-access-control-protocol-data.html>

[7] Wireless communications and networks / William Stallings

Årstill: C 2005: ISBN: 0-13-191835-4, h. Utgave: 2nd ed

<http://www.pearsonhighered.com/stallings/details5.html>

[8] Wireless Center: All Wireless Articles

<http://www.wireless-center.net/WLANs-WPANs/1436.html>

[8] Wireless Center: All wireless articles: 802.11 family physical layer (physical layer)

<http://www.wireless-center.net/Wireless-Internet-Technologies-and-Applications/1871.html>

[8] How does a wireless LAN work

<http://www.wireless-center.net/Cisco-Wireless-Networking/725.html>

[8] Wireless LAN technologies: (technology)

<http://www.wireless-center.net/WLANs-WPANs/2435.html>

[8] Wireless Center: Infrared Light-Based Wireless LANs

<http://www.wireless-center.net/WLANs-WPANs/1385.html>

and: IEEE 802.11 Infrared Specifications

<http://www.wireless-center.net/Next-Generation-Wireless/IEEE-80211-Infrared-Specifications.html>

[8] Wireless Center All Wireless Articles: IEEE 802.11b Supplement to 802.11 standards: (IEEE 802.11 standards)

<http://www.wireless-center.net/Next-Generation-Wireless/IEEE-80211b-Supplement-to-802.11-Standards.html>

[8] Wireless Center All Wireless Articles: Medium Access Control

<http://www.wireless-center.net/Wireless-Internet-Technologies-and-Applications/1872.html>

[9] Intelligraphics device drivers: Introduction to IEEE 802.11: (Physical layer)

<http://www.intelligraphics.com/introduction-ieee-80211>

[10] Tutorial Reports.com: IEEE 802.11 Architecture: (IEEE 802.11 Architecture)

http://www.tutorial-reports.com/wireless/wlanwifi/wifi_architecture.php

[10] Tutorial-Report.com: Physical layer:

http://www.tutorial-reports.com/wireless/wlanwifi/wifi_physical_layer.php

- [11] IEEE 802.11 Technical Tutorial
<http://www.qsl.net/n9zia/wireless/pdf/802.11.pdf>
- [12] Answers.com : Media Access Control
<http://www.answers.com/topic/media-access-control>
- [13] Wi-Fi Planet: Tutorials: 802.11b Physical layer revealed
<http://www.wi-fiplanet.com/tutorials/article.php/2107261>
- [13] Wi-Fi Planet: Tutorials: 802.11a Physical layer Revealed
<http://www.wi-fiplanet.com/tutorials/article.php/2109881>
- [14]] Xin He and Frank Y, Li
<http://www.vde-verlag.de/proceedings-en/453167027.html>
- [14] Xin He and Frank Y, Li
http://www.uia.no/en/portals/about_the_university/engineering_and_science/-_ict/mobile_communications_group/aml/aml_publications
Cooperattive RTS/CTS MAC with Relay Section in Distributed wireless networks
Xin He and Frank Y, Li
Dept, of information and communication technology, University of Agder (UiA)
- [14] Pei Liu, Zhifeng Tao, Sathya Narayanan, Thanasis Korakis , and Shivendra S.Panwar: CoopMAC: A Cooperative MAC for Wireless LANs
- [15] The Network simulator ns-2
<http://www.isi.edu/nsnam/ns/>
- [16] Xin He and Frank Y, Li
An Automatic Cooperative Retransmission MAC Portocol in Wireless local Area Network
Xin He and Frank Y, Li
Dept, of information and communication technology, University of Agder (UiA)
- [17] CMU Monarch project, Computer Science Department, Canergie Mellon University, Pittsburgh. "The CMU Monarch project's wireless and mobility extentions to ns", 1999
- [18] Kevin Fall, Kannan Varadham;UC Berkeley, LBL, USC/ISI, and Xerox PARC
The ns Manual (formerly ns Notes and Documentation
- [19] ATutorial of wireless Simulation in ns-2
Yue Wang
- [20] Eitan Altman and Tania Jimenez
NS Simulator for beginners : Lecture notes. 2003-2004
- [21] The Network Simulator - ns-2
<http://www.isi.edu/nsnam/ns/>

[22] IEEE 802.11g, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, Amendment 4: Further Higher Data Rate Extension in the 2.4 GHz Band,

<http://grouper.ieee.org/groups/802/11/>

[23] Hossein, M., Turletty, T. 2003. Simulation-Based Performance Analysis of 802.11a Proceedings of the Int'l Symposium on Telecommunications (Isfahan, Iran, August 16-18, 2003), 758–762

[24] Multiband Atheros Driver for WiFi

<http://www.madwifi.org>

[25] Yang, L. , Hanzo, L. 2000. A Recursive Algorithm for the Error Probability Evaluation of M-QAM, IEEE Communications Letters, Vol. 4, No. 10, (Oct. 2000)

[26] Goldsmith, A. 2005. Wireless Communications. Cambridge University Press.

[27] The ns Manual (formerly ns Notes Documentation)

The VINT Project A collaboration between research at UC Berkeley, LBL, USC/ISI, and Xerox PARC Kevin Fall, and Kannan Varadha, January 6, 2009

<http://www.slideshare.net/code453/ns2-manual>

[28] Sniffing out correct error frame model for the ns-2 simulator:
Jeng-Fam Lee, Meng Chang Chen, Cheng-Lin Tsao

10.Abbreviations

Abbreviations	
AP	Access point
BER	Bit Error Rate
BPSK	Binary Phase-Shift Keying
CS	Carrier Sence
CSMA/CA	Carrire Sense Multiple Access/Collision Avoidance
CTS	Clear-to-Send
CW	Contention Window
DCF	Distributed Coordination Function
DIFS	DCF Interframe Space
FEC	Forward Error Correction
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
ISO	Internalional Standards Organization
LA	Link Adaptation
LOS	Line-of-Sight
MAC	Medium Access Control
MANET	Mobile Ad Hoc Network
modARF	Modified Auto Rate Fallback
MTU	Maximum Transmission Unit
NAV	Network Allocation Vector
NORT	No Routing Protocol

Abbreviations	
OAP	Opportunistic Auto Rate
OFDM	Orthogonal Frequency Division Modulation
OSI	Open System Interconnection Reference Model
PAM	Pulse-Amplitude Modulation
PCF	Point Coordination Function
PDF	Probability Density Function
PER	Packet Error Rate
PHY	Physical Layer
PLCP	Physical Layer Convergence Procedure
PSR	Packet Success Rate
QPSK	Quadrature Phase-Shift Keying
RBAR	Receiver Based Auto Rate
RTS	Request-to-Send
SER	Symbol Error Rate
SIFS	Short Interframe Space
SINR	Signal-to-Interference-and-Noise Ratio
SNIR	Signal-to-Noise-and-Interference Ratio
SNR	Signal-to-Noise Ratio
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VANET	Vehicular Ad Hoc Network
WLAN	Wireless Local Area Network