A RECEIVER-INITIATED MAC PROTOCOL WITH ENHANCEMENTS FOR MULTI-HOP WIRELESS NETWORKS

By

BALVINDER KAUR THIND

A thesis submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering and Computer Science

MAY 2004

To the Faculty of Washington State University:

The members of the Committee appointed to examine the thesis of BALVINDER KAUR THIND find it satisfactory and recommend that it be accepted.

Chair

ACKNOWLEDGEMENT

First and foremost, I would like to express my gratitude to my advisor Prof. Murali Medidi for being my advisor and for his persistent motivation, support, and farsighted guidance from the beginning of my research work. He taught all of us to have fun while doing research and that, those timely breaks are as essential as is timely work. This research project that was started under his supervision would have never worked out without his advice and guidance. He has been the best possible advisor, both in terms of research and in terms of professional advice.

Also, my sincere thanks to Prof. Sirisha Medidi and Prof. Dave Bakken for serving on my thesis committee, for sharing their expertise during my research work and for their constructive comments on the thesis. I thank Ruby Young for her friendly attitude and approachability and for guiding me through the many administrative paper work and deadlines.

I am also grateful to all who have been a part of the Wireless Security Research Group (WiSeR). I have enjoyed working with skillful colleagues, and friends, in an inspiring atmosphere since the days at Washington State University. Special Thanks to Aniruddha Daptardar for his friendship and valuable insights and for continually remembering my topic. I am grateful to our “Desi Cougs” gang who made my work easier by all that parental support that I needed when I was seven seas across from my parents.

This thesis is dedicated to my mother, Paramjeet Thind and my father, Zail Thind, for without their love and motivation over the years I would not be at this point today. I am forever indebted to my parents for everything that they have given me. Last and by no means least, a special mention for my brother, Ravi, my sister-in-law Sweety and my cute nephew, Ranveer who have provided all that support and love required for work as this to be completed successfully. Having Ravi, Sweety and Ranveer nearby kept the characters of this saga alive in my mind. Ranveer’s smile helped me learn that this thesis of mine on collisions and interference can be completed with my wits intact. Finally, I would like to mention that I dedicate this work to all the people who love me and whom I love.

A RECEIVER-INITIATED MAC PROTOCOL WITH ENHANCEMENTS FOR MULTI-HOP WIRELESS NETWORKS

Abstract

Balvinder Kaur Thind
Washington State University
May 2004

Chair: Muralidhar Medidi

Multiple channel access interference is a major cause of throughput degradation in wireless networks because of the shared channel. IEEE 802.11 MAC is a standard for medium access in wireless LANs, but suffers from channel contention and co-channel interference and thus, performs poorly. The focus of this thesis is to design and study a medium access control protocol that mitigates the effect of multiple channel interference.

We propose to use a receiver-initiated MAC protocol, instead of the sender-oriented 802.11 MAC. Our protocol is based on carrier sensing and resolves collisions among senders based on a deterministic tree splitting algorithm. By doing collision resolutions, we aim to use time efficiently otherwise wasted in 802.11 MAC for random backoffs, particularly in cases when the channel contention is high. Receiver-initiated based collision resolutions guarantee that, no data packets collide at a receiver because of interference from its own senders: a major cause of hidden node problem.

Further, the protocol is enhanced to have multiple subchannels by dividing the common communication channel. All the subchannels serve the purpose for both control as well as data packets. A subchannel assignment scheme is proposed to exploit the parallel transmissions that are possible in multi-channel networks. This should help in reducing the co-channel interference and thereby, improve the throughput.

In order to handle the unfairness issues associated with receiver-initiated protocols, we propose a third enhancement: an adaptive approach of deliberate transitions between receiving and sending modes. These mode transitions force nodes to spend fixed time in each mode, thereby giving a fair chance to become a sender as well as a receiver.

We also present simulation results using ns2 simulator, varying several system parameters to evaluate our approach and compare it with standard IEEE 802.11. The simulation results indicate that collision resolution with multiple subchannel access provides throughput enhancement and better packet delays than 802.11 MAC. We also observe that the maximum throughput is achieved when the channel is divided into three or four subchannels irrespective of the size, shape of the network, and traffic load. The results also indicate that we further improve on throughput by doing deliberate transitions and, that the protocol has better fairness for QoS assurance as compared to standard 802.11 MAC.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

1.1 Motivation and Problem Statement

Wireless communications in various forms has been the subject of much attention and research in recent years. With the emergence of Wireless Local Area Networks (WLANs) emerges the fact that ideally users of wireless networks will want the same services and capabilities that they have commonly come to expect with wired networks. However, to meet these objectives the wireless community faces certain challenges and constraints that are not imposed present in wired counterparts.

In wireless networks, the nodes must use a common transmission medium instead of multiple point-to-point interface. The common transmission medium is a very scarce resource and so, its efficient usage is imperative. In WLANs, the sharing of the common transmission medium, or channel, by several nodes is determined by the *Medium Access Control (MAC)* protocol. The MAC protocol is the main element that determines the efficiency of sharing the limited communication bandwidth of the wireless channel. Thus, the MAC scheme that governs the transmission of data packet between different nodes should attempt to maximize the number of packets exchanged per second which is calculated as the throughput and to minimize the time required for a packet transmission or the delay of the network. The fraction of channel bandwidth used by successfully transmitted messages gives a good indication of the protocol efficiency. However, designing MAC is a serious challenge in wireless networks because interference is inherent in all wireless systems and is one of the most important issues to be addressed in the design, operation and maintenance of wireless communication systems.

Consider an audio conference where:

1. If one person speaks all can hear.

2. If more than one person speaks at the same time, all the voices will be garbled.

We should try to formulate, how should the participants coordinate actions so that, the number of messages exchanged per second is maximized and time spent waiting for a chance to speak is minimized. This summarizes problem of *multiple access* which is present in wireless networks.

Uncontrolled transmission in a shared medium may lead to the time overlap of two or more packet receptions, called collision or interference resulting in corrupted packets at the destination. The topology dynamics of the wireless networks with the use of a common transmission medium brings up the problem that, in some cases, a node may receive concurrent transmissions from multiple neighbors that cannot hear one another. We call these nodes as *hidden* from each other and this problem is defined as the *hidden node problem* discussed by Tobagi and Kleinrock [38]. This can also be defined as the *contention interference* because two or more source nodes target the same receiver. Due the existence of the *hidden node problem*, unnecessary collisions of data packets occur which are followed by message retransmissions. Such message retransmissions causes a wastage of the efficient bandwidth which could have been otherwise used for successful message transmission.

There is also another serious issue with the design of efficient MAC schemes. Co-channel interference occurs when two or more pairs of nodes that are in communication range of each other try to transmit simultaneously in the same channel. The communication between one pair of nodes affects the packet transmission between other pairs. Such nodes are called as *exposed nodes* and the problem is defined as the *exposed node problem*. Presence of exposed nodes also results in less efficient usage of the medium. The throughput degrades as multiple parallel transmissions are not possible even when they are feasible.

Designing an efficient MAC protocol that works well even with the existence of hidden and exposed nodes is the fundamental problem of wireless networks. The IEEE 802.11 was developed as a standard for MAC in wireless local area networks. It is based on the Carrier Sense Multiple

Access Scheme with Collision Avoidance (CSMA/CA) and implemented a sender-initiated four-way handshake mechanism of RTS-CTS-DATA-ACK for data transfer. The idea here is that the channel is always sensed before any transmission. If the channel is sensed busy, the sender goes through a random backoff period before retrying. The Request-To-Send/Clear-To-Send (RTS/CTS) exchange before actual data transfer was to reserve the channel before data transmission. This technique was introduced to minimize the number of data packets being dropped due to collisions in presence of hidden nodes.

However, the standard was not able to completely eliminate the hidden node problem. The RTS and CTS sent using carrier sensing at the sender can still collide at the receiver. Although the size of these control packets, RTS and CTS, is very small as compared to data packets, severe degradation in throughput occurs at high loads. And in the worst case, if any of these control packets which serve to indicate the neighboring nodes of an ongoing transmission got corrupted, then the data packets would collide at the receiver. IEEE 802.11 MAC is prone to inefficiencies at heavy loads, since with increasing traffic there is a higher wastage of bandwidth from collisions and backoff. And since backoff delays are unsynchronized, medium can be idle if all contenting nodes are in backoff. Also, the standard does not deal with the exposed node problem.

Several MAC schemes have been proposed in the past that try to alleviate the problems mentioned above. The goal of our research is to introduce, study and compare a new channel access method which reduces the number of collisions among data packets thereby providing better throughput performance and channel utilization than the standard 802.11 MAC.

IEEE 802.11 MAC suffers throughput degradation because of the inefficient random backoff's when RTSs collide in presence of hidden nodes. At high loads, these RTSs collisions increase and significant amount of delay in introduced by successive retry attempts each after a longer backoff. Instead of following a probabilistic approach of random backoff's after collisions, we can follow a deterministic approach by resolving these collisions and hence, define an order in which the sender nodes will send the data packet. If the collisions of RTSs can be resolved efficiently in time less than data packet duration, then more efficient usage of bandwidth is possible. Also, a receiver node is more aware of the contention around itself. The hidden node problem exists because of some source nodes, unaware of each others existence starts transmitting to a common node at the same time. The common node which is the receiver node can listen to all its 1-hop neighbors who are one of the potential causes for this hidden node problem. So, if the common node can control the transmissions from these hidden nodes, we can minimize the data packets dropped due to hidden node problem. This concept led us to think about a medium access protocol which is receiver- initiated. The receiver invites its 1-hop neighbors to send RTSs to it. Due to propagation delays, the RTSs are prone to collide and the collisions among such RTSs from multiple senders are resolved using the deterministic tree splitting algorithm.

The use of multiple channels may provide some performance improvement by reducing the collisions due to co-channel interference, and thereby enabling concurrent transmissions. This will increase the channel utilization and hence, throughput. Multichannel protocols allow multiple nodes in the same neighborhood to transmit concurrently, without interference. We further enhance the receiver-initiated MAC protocol with collision resolutions to work in a multi-channel environment, where the common transmission channel is divided into subchannels, whose number is much smaller than the number of nodes. Every node now receives the data packets on a subchannel assigned to it. Since the subchannels share the fixed channel capacity allocated to the network, their number must not exceed a given bound. We aim at minimizing the exposed node problem and at the same time achieving high throughput. Hence, we propose a new subchannel assignment scheme that is based on maximum degree coloring scheme. The subchannel assignment attempts to assign a subchannel with minimum occurrence in 1-hop and 2-hop neighborhood and it also takes into account the density of 1-hop and 2-hop neighborhood.

Wireless channel is a shared scarce resource. The MAC protocols used over wireless networks are distributed protocols which try to avoid collisions and provide the nodes in a network with an access to the channel is a fair manner. A channel access protocol (MAC) is termed to be unfair if it fails to provide the channel access to individual nodes without giving preference to one node over others when there is no explicit differentiation. Though the wired Ethernet protocol based on CSMA/CD is known to be fair, its wireless counterpart 802.11 based on CSMA/CA is proven to be unfair. Fairness problems occur when some nodes tend to grab the shared channel thereby making other nodes suffer. The binary exponential backoff mechanism followed by the nodes in 802.11 for collision avoidance, increase the chances of a node which recently succeeded in winning the channel everytime it contends because of the lower backoff window than the failed nodes. So the binary exponential backoff becomes a cause of unfairness in wireless networks.

If we follow a different approach than exponential backoff, receiver-initiated protocol with collision resolution, for collision avoidance, then we may be able to overcome this fairness problem related to channel access. However, there is certain unfairness related to the receiver initiated approach. In receiver-oriented MAC protocols, the fairness problems might occur as a result of some node spending all of its time in its sending mode and very little time as a receiver. Such a node will behave unfairly to all its sender nodes by not initiating the CRI. This is common at high loads when the transmission queue of sender nodes are full. In order to control this fairness problem, we might have to forcibly make such a node behave as a receiver at some point, even if it has packets in its transmission queue. We further propose an enhancement to handle fairness issues among nodes by deliberately making a node to switch between its roles as sender and receiver.

This thesis advances the state of the art in several ways, ranging from the explanation of the current MAC protocols to the development of a new MAC protocol and some enhancements which when combined together are far more efficient than the standard IEEE 802.11 MAC.

1.2 Organization of Thesis

The thesis is organized as follows. Chapter 2 gives an insight about the standard IEEE 802.11 MAC scheme and serves to provide a background on the issues related with MAC schemes. It also describes some of the previous MAC schemes proposed to address the issues of the MAC. We point out what the previous MAC based protocols lack and present our motivation towards the proposed MAC scheme. Chapter 3 describes the proposed receiver-initiated MAC protocol. We have divided Chapter 3 into sections starting with the working of a receiver-initiated protocol in presence of collision resolutions following by the explanation of the deterministic tree splitting algorithm used for the resolution of collisions. Further we present the next enhancement to the proposal: subchannel assignment, where we identify the need for subchannels and propose a sub-channel assignment algorithm based on the heuristic of *largest degree first*. Finally, we talk about the fairness problems faced by the receiver-initiated scheme and propose a scheme of deliberate mode transitions to handle them. Chapter 4 presents the simulation results and the performance evaluation of the proposed protocol as compared to the IEEE 802.11 MAC. Lastly, Chapter 5 presents the conclusions and some future research directions.

CHAPTER TWO

BACKGROUND AND RELATED MATERIAL

2.1 IEEE 802.11 Wireless Local Area Networks

The IEEE 802.11 specifications are wireless standards that specify an “over-the-air” interface between a wireless client and a base station or access point, as well as among wireless clients. This project was initiated in 1990, and several draft standards have been published for review. The IEEE 802.11 specifications address both the Physical (PHY) and Media Access Control (MAC) layers to provide wireless connectivity for fixed, portable and moving stations within a local area.

Under the IEEE 802.11 standard, stations can operate in two configurations. According to the first configuration, stations can directly communicate with each other. The IEEE standard refers to this as *independent configuration* and can be considered as being similar to a point to point method of communication. As no infra-structure needs to be installed, this configuration represents the ad-hoc networks. The second configuration method supported by the IEEE standard is referred to as an *infrastructure configuration*. Under the infrastructure communication method, stations communicate with one or more access points, which are connected to a Wired LAN.

Mapped to the seven layer Open Systems Interconnection (OSI) model, 802.11 defines operations at the physical layer and data link layer. The standard is designed to provide wireless connectivity for portable, fixed and roaming stations within a local area with a Medium Access Control (MAC) and physical layer (PHY) specifications. The layers of the OSI model are included in the standard much like the current 802 LAN, and are designed to appear the same as a wired LAN to higher level layers. The standard defines the link between the actual radio channel and the higher network layer protocols.

2.1.1 PHY Layer

The PHY provides the link between the MAC sublayer and the actual radio channel. It encapsulates data from the MAC and transmits the physical frame to the receiving stations PHY. The IEEE 802.11 draft specification calls for three different physical-layer implementations: frequency hopping spread spectrum (FHSS), direct sequence spread spectrum (DSSS), and IR.

Frequency Hopping Spread Spectrum

FHSS defines a set of channels spaced across the whole radio bandwidth. The IEEE 802.11 frequency-hopping physical layer uses 79 non-overlapping frequency channels, with each channel having a 1-MHz channel spacing. This enables upto 26 co-located networks to operate, which can provide a reasonably high aggregate throughput.

Direct Sequence Spread Spectrum

DSSS defines a set of channels spaced across the whole radio bandwidth. There are 14 of these channels, but channel 14 is reserved for Japan. DSSS modulates the data with a spreading code (chipping) and transmits the result on only one of these channels. There has to be 30MHz between the carrier frequencies for multiple access points to operate within the same area without interference. Since the entire bandwidth is 83.5MHz, only a maximum of 3 DSSS access points can operate within the same area. The limited available total bandwidth is also the methods vulnerability. If narrow band interference occurs in the used channel, one can only wait until it disappears before communications can be resumed. In return DSSS gives a longer range.

Infrared

In concluding this examination of the IEEE 802.11 physical layers, lets turn to the third physical layer supported by the IEEE 802.11 specifications. This layer is the infrared physical layer. The IR specification identifies a wavelength range from 850 to 950 nm. IR reception is based on diffuse IR transmission, which means that a clear line-of-sight path between transmitter and receiver is not required. However, the allowable range between stations is limited to approximately 10m, and the use of this layer is restricted to in-building applications.

2.1.2 Medium Access Control sublayer (MAC)

The MAC layer represents a uniform scheme that supports multiple physical layers. Although the primary function of the MAC layer is to control access to the wireless environment, this layer is also responsible for fragmentation, encryption, power management, and synchronization. In addition, the MAC Layer is also responsible for providing roaming support where there are multiple access points.

Basic Access Method

The IEEE 802.11 standard uses a variation of the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) to provide a wireless access capability. The access method used by the IEEE 802.11 standard is referred to as the Distributed Coordinated Function (DCF) which can be considered to represent the CSMA/CA model. The MAC architecture is depicted in Fig. 2.1, where it is shown that the DCF is positioned directly on top of the physical layer and supports contention services. Contention services imply that each station with a data packet queued for transmission must contend for access to the channel and, once the data packet is transmitted, must re-contend for access to the channel for all subsequent frames. Contention services promote fair access to the channel for all stations.

The variation of the CSMA/CA protocol used requires a station that has information to transmit to first *listen* to the medium. If the channel is found to be busy; then the station waits until the channel becomes idle for a DIFS period (Distributed Interframe Space) and defers its transmission. If the medium is available, the station can transmit. Because it is possible that another station could transmit at approximately the same time, the receiver will check the CRC of received packets and transmit an acknowledgment that serves as an indicator to the originator that no collision occurred.

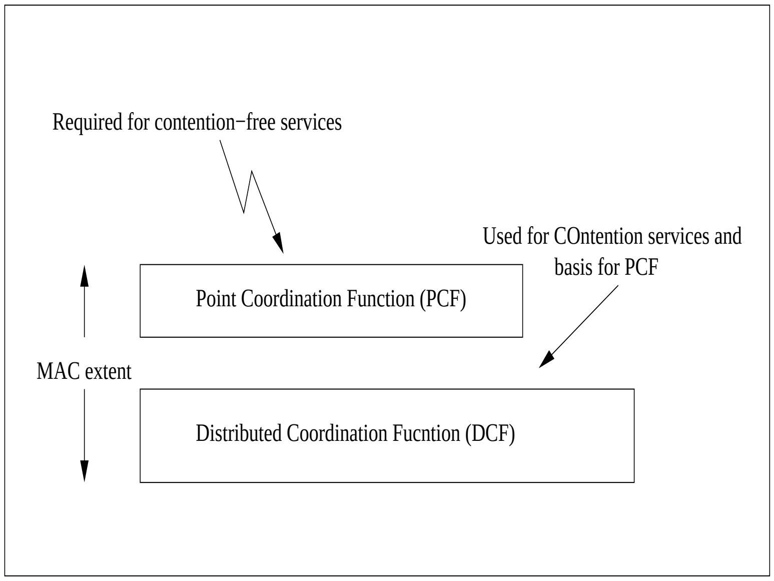

Required for contention−free services
Used for COntention services and basis for PCF
Point Coordination Function (PCF)
MAC extent
Distributed Coordination Fucntion (DCF)

Figure 2.1: MAC architecture

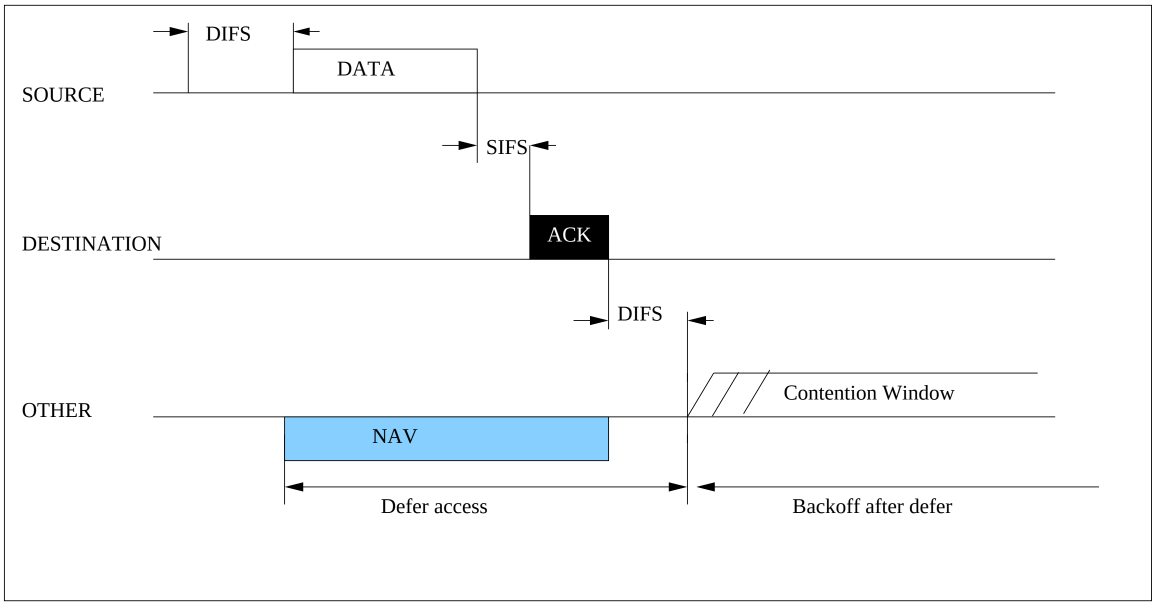


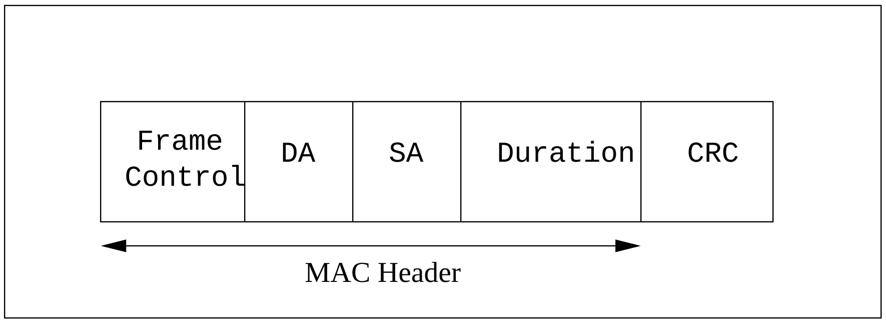
Figure 2.2: Transmission of data packet by 802.11 MAC

Otherwise, if the sender does not receive an acknowledgment, it will retransmit until it receives an acknowledgment or a predefined number of retransmissions occur. Fig. 2.2. shows this data transfer.

Use of Control Frames for Data Transfer

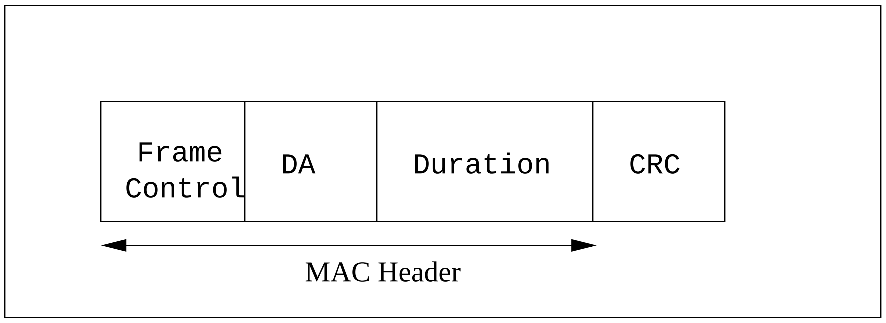
Because it is possible that two stations can both listen and hearing no channel activity, transmit or two stations do not hear one another and both transmit, collisions can occur. Detailed explanation of the reasons for such collisions are provided in the later sections. To reduce the probability of collisions, a source station performs Virtual Carrier Sensing (VCS). Under VCS, a station that needs to transmit data will first transmit a Request-To-Send(RTS) packet. The RTS packet is a relatively short control packet that contains the source and destination addresses and the duration of the following transmission. The format of the RTS packet is as shown in Fig. 2.3. The duration is specified in terms of the data packet and the acknowledgment of the packet by the receiver. The receiver will respond to the RTS packet with a Clear-To-Send (CTS) control packet. The format of CTS packet is as shown in Fig. 2.4. The CTS packet will indicate the same duration information as contained in the RTS control packet. Each station that receives either the RTS or CTS control packet or both will set its Virtual Carrier Sense indicator: Network Allocation Vector (NAV) for the duration of the transmission. This NAV serves as a mechanism to alert all other stations on the medium to back off or defer their transmissions.

The timing diagram for a data transfer using control packets is shown in Fig. 2.5 where a data transfer takes place between a source and destination with the help of control packets RTS/CTS and the other nodes set their NAV vector depending on the duration information present in the RTS/CTS. If the CTS is not received within a predefined period of time, the source station will assume that a collision occurred and will back off. After completing the back off, the source node will initiate the procedure again, issuing another RTS packet. Once the CTS frame is received and a data packet is sent, the receiver will return an acknowledgment (ACK) packet to acknowledge a successful data transmission.



DA: Destination Node Address
SA: Source Node Address
Duration: Time in Microseconds required to transmit the next data.

Figure 2.3: RTS Frame Format



DA: Destination Node Address

Duration: Time difference between the duration field of RTS and time required to transmit CTS frame

Figure 2.4: CTS Frame Format

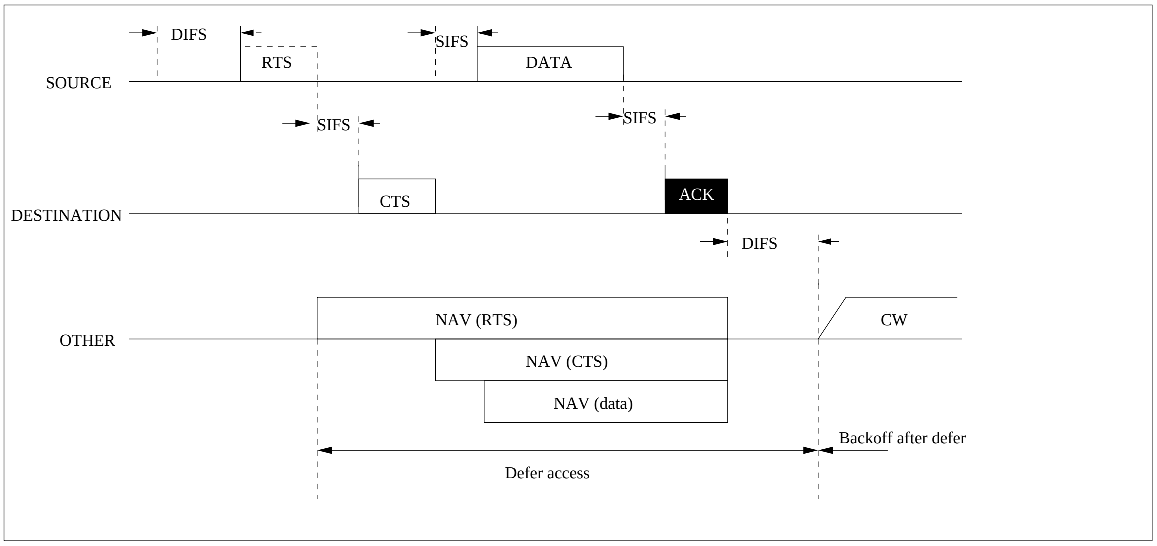


Figure 2.5: Transmission of a data packet by 802.11 MAC with the use of control packets

The use of RTS and CTS control packets reduces the probability of a collision occurring at the receiver from a station that the source node cannot listen. Collisions of RTS from multiple source nodes can still occur, but the size of a RTS (around 20 bytes) is very less as compared to the size of a data packet usually in the size of Kbytes.

2.2 MAC Limitations

In wireless network nodes transmit packets in an unsynchronized fashion. The Medium Access Control protocol [21] is responsible for co-ordinating multiple access to the shared channel minimizing conflicts. This multiple access schemes can be classified as fixed-assignment and demand-assignment multi access schemes. Fixed-assignment multi access divides the available space of the channel into subchannels with one subchannel assigned per individual user. Demand-assignment multi access allows a device to transmit immediately when data is available, but this protocol must account for the possibility of contention on the line when two or more devices simultaneously transmit. Some of the Demand-assignment multi access schemes are Pure ALOHA, Slotted

ALOHA and Carrier Sense Multiple Access (CSMA).

The pure ALOHA protocol is a random access protocol used for data transmission. A node accesses a channel as soon as a message is ready to be transmitted. After a transmission, the node waits for an acknowledgement (ACK) on either the same channel or a separate feedback channel. In case of collisions, i.e. when a negative ACK is received, the node waits for a random period of time and retransmits the message. As the number of nodes increase, a large delay occurs because the probability of collision increases. In slotted ALOHA, time is divided into equal slots of length greater than the packet duration. Each node have synchronized clocks and transmit a message only at the beginning of a new time slot thus preventing partial collisions, where one packet collides with a portion of another. As the number of nodes increase, a greater delay will occur due to complete collisions and the resulting repeated transmissions of those packets originally lost. ALOHA protocols do not listen to the channel before transmission, and therefore do not exploit information about the current state of the network. By listening to the channel before engaging in transmission, greater efficiencies may be achieved.

CSMA protocol [27] is based on the fact that each node on the network is able to monitor the status of the channel before transmitting. If the channel is idle, then the node is allowed to transmit a packet based on a particular algorithm which is common to all the nodes on the network. As the transmission time of the packet is usually larger than the propagation delay of the channel, packet vulnerability period is less than in ALOHA. Accordingly, CSMA performs better than ALOHA in a fully connected network. However, CSMA protocols do not work very well as the channel state might be different at the receiver from what is estimated at the receiver. The issues related to packet transmissions on a shared channel, as faced by the MAC protocol can be classified as:

1. Hidden Node Problem: Interference in wireless communications can be caused by simultaneous transmissions (i.e. collisions), two or more sources sharing the same channel become idle and begin transmission at the same time. The topology dynamics of the WLAN with the

use of the shared transmission medium brings up the problem that, in some cases a node may receive concurrent transmissions from two or more neighboring nodes that cannot hear one another. We call these nodes as **hidden** from each other. This is defined as the **hidden node problem**. It is also called as the **secondary channel interference**. Collisions are caused by when a node, sensing the channel idle, begins transmission without successfully detecting the presence of a transmission already in progress. Much like a broadcast storm on a wired LAN segment can bring traffic to a standstill, hidden nodes interfering with one another will have a very detrimental effect on the performance of every wireless node in the network. This interference can cause overall performance of the entire wireless network to drop by as much as 50%. When using video streaming this number can easily increase to 70% due to the continuous nature of the transmission.

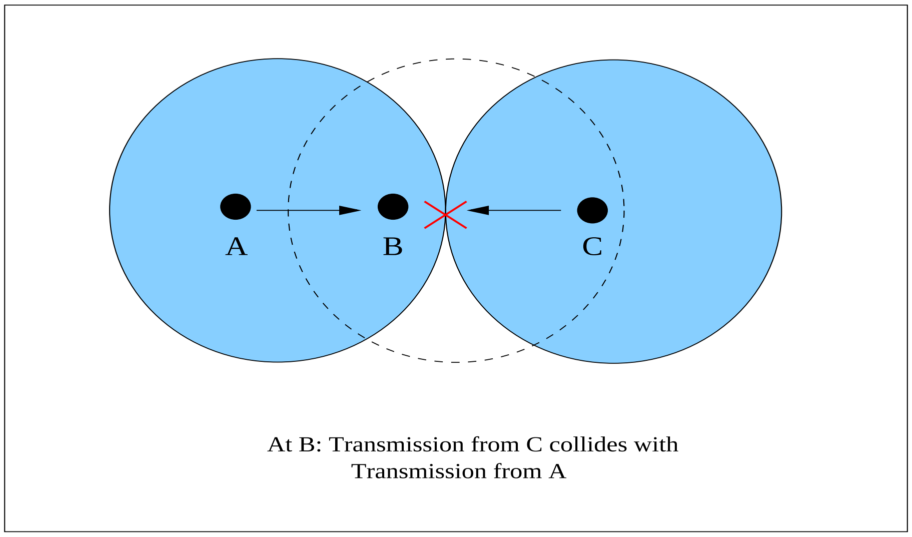


Figure 2.6: Hidden Node Problem

Fig. 2.6 illustrates a scenario of three nodes, wherein node-pairs, (A,B) and (B,C) can communicate with each other; however the transmission ranges of node C and A do not overlap and hence, node C cannot listen to the transmissions from node A. Following the standard

IEEE 802.11 MAC, without RTS/CTS handshake, nodes A and C both find the channel to be idle and hence, send data packet to B, thus, causing collisions of data packets. To overcome this problem, channel reservation techniques by exchanging small control packets RTS/CTS before data packet is sent was introduced in IEEE 802.11 MAC as explained in Sect. 2.1.2. This effectively performs a “virtual carrier sensing” at the receiver by letting the sender know whether the channel state at the receiver is conducive for packet reception. In addition, the channel is temporarily reserved for the data transmission since neighbors of the receiver who receive the CTS defer transmission at least for the duration of data transfer. RTSs sent within the propagation time delay would still collide which is not as bad as the collisions of the data packets, because the length of the RTSs and the CTSs are very small in comparison to the length of the data packets. However, if A and C did not send the RTS at the same time, this RTS/CTS handshake would be helpful, as node C after listening to the CTS from B will update its NAV and thus, not interfere with the ongoing transmission from A to B, solving the hidden node problem. In some worst cases, however, node C might receive a corrupted CTS from B and hence will loose the current state of the network. In that case it is a possibility that C might send its RTS to C when A was sending its data packet to B, thus, resulting in collision of data packet at B. Thus, we see that, by appending the RTS/CTS handshake with a normal Data/ACK handshake alone cannot solve the hidden terminal problem. The hidden terminal problem still persists in IEEE 802.11 MAC.

The side effects are that the throughput decreases and the average packet transmission delay increases. Thus, with the existence of this problem in the WLANs, sensing the channel activity at the sender node do not always offer valuable information about the state of the channel at the intended receiver. At high loads, there is a higher wastage of bandwidth from collisions and backoffs and also since, the backoff delays are unsynchronized, medium can be idle if all contending nodes are in backoff.

2. Exposed Node Problem: Carrier sensing medium access techniques face problems not only when nodes cannot listen to each other due to the nature of the topologies of the wireless networks, but also when the nodes can very well listen each other and so, are barred from having simultaneous transmissions in the neighborhood. A channel that is occupied by a pair of hosts can be reused by another pair of hosts, only if their communication range do not overlap. Such nodes are called as *exposed nodes* and the problem is defined as the *exposed node problem*. An exposed node can be one of two cases: a node which is in the neighborhood of the sender but not the receiver or a node which is in the neighborhood of the receiver but not the sender. Thus any node hearing RTS or CTS must defer at least until the end of the entire exchange (i.e until the end of ACK).

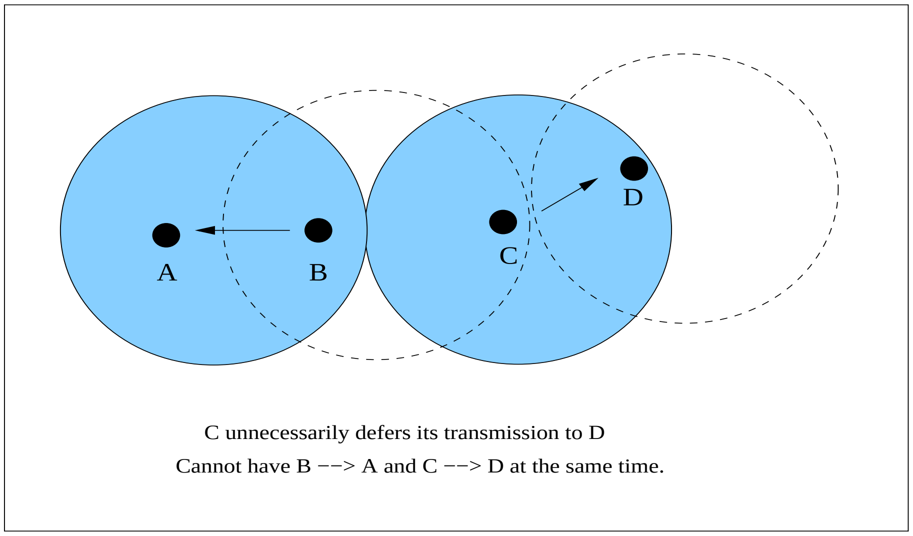

A
B
C
D
C unnecessarily defers its transmission to D
Cannot have B --> A and C --> D at the same time.

Figure 2.7: Exposed Terminal Problem

In Fig. 2.7, we have 4 nodes, of which the node-pairs (A,B), (C,D) and (C,B) are in communication range of each other. B is a potential sender to A and C has a data packet to send to D. Now we can have only one transmission either from B to A or from C to D taking place at one time. This is because the transmissions (RTS or DATA) from B will interfere with the transmissions from C though (A and C) or (B and D) cannot listen each other. Concurrent transmissions cannot take place, since each sender node or receiver node needs to be able to receive packets (CTS and ACK) and (RTS and Data) respectively, correctly, which may see interference from similar packets from the other pair of nodes. When B sends out the control packet RTS, node C picks up the duration information from that RTS and sets its NAV, further deferring its transmission to D. Thus, we can see that concurrent transmissions from B to A and C to D cannot take place even though they were feasible. Side effects are that the channel utilization decreases. Bandwidth does not get efficiently utilized and hence we see less throughput.

2.3 Related Work

Several Medium Access Control protocols have been proposed in the past that try to alleviate the problems mentioned above and to reduce the number of collisions among data packets, thereby providing better performance than the CSMA protocols. Fundamentally, all these protocols are based on the fact that the nodes contend for the channel on a packet-by-packet basis. These protocols try to concentrate on solving the interference issues and trying to handle the exposed problem issues which could possibly lead to throughput enhancement and better channel utilization. Further, they can be broadly classified as the sender-initiated MAC protocols, the receiver-initiated ones, and MAC protocols with multiple channels.

2.3.1 Sender Initiated Multiple Access protocols

The first category of such MAC protocols includes the sender-initiated methods where each sender node competes to acquire the floor by sending the control packet. The basic idea is for a sender to transmit a request-to-send(RTS) that the receiver acknowledges with a clear-to-send (CTS); now if they both have a successful RTS-CTS handshake, then the sender is allowed to transmit one or more packets. The protocols differ in the methods they use to resolve the collisions among RTSs.

Kleinrock and Tobagi [27] identified the hidden-terminal problem of carrier sensing, which makes Carrier Sensing Multiple Access (CSMA) perform as poorly as the pure ALOHA protocol when the senders of packets cannot hear one another and the vulnerability period of packets becomes twice the packet length.

Busy Tone Multiple Access

The BTMA protocol [38] was the first attempt to solve hidden terminal problem by introducing a separate busy tone channel. DBTMA protocol [12], further proposed the use of multiple busy tones to indicate when a receiver is busy. It divides a common channel into two subchannels, a data channel and a control channel. Data packets are transmitted on data channel. Control packets (RTS/CTS) are sent on control channel. Two busy tones are assigned on the control channel: BT_t(the transmit busy tone), which is an indication that a node is transmitting on the data channel and the BT_r (the receive busy tone) which shows that a node is receiving on the data channel. A node when has data packet senses the channel for BT_r before it acquires the channel. If no BT_r is heard, then there is no one in the node's neighborhood that is receiving data and so the node can send the RTS now. When some node receives a RTS, it senses the channel for BT_t, absence of which indicates that there are no other senders and so it can send CTS and at that time it turns on its BT_r signal. The sender of RTS on hearing CTS, turns on its BT_t signal. This scheme still cannot support parallel transmissions as it has only one data channel and so is not intended for increasing the channel utilization. The presence of one extra control channel cause wastage of bandwidth.

Floor Acquisition Multiple Access

The concept of floor acquisition was introduced in FAMA protocol [13, 15]. FAMA protocol is based on the concept that for a node that has data to send should acquire the floor before sending any data packet, and to ensure that no data packet collides with any other data packet at the receiver.

Although each station transmits an RTS only when it senses the channel to be free, a collision with other RTSs transmission may still occur due to propagation delays. RTSs are required to last a minimum amount of time that is the function of the channel propagation time and CTSs are required to last longer than an RTS time plus the maximum round trip time. The efficiency of FAMA protocols using carrier sensing to eliminate hidden terminal problem has been analyzed and verified.

However, the throughput of FAMA protocol still degrades rapidly once we have collisions with RTSs and unsuccessful retransmissions of RTSs. FAMA protocols solve collisions by backing off and rescheduling transmissions. This procedure produces good results only if the RTS traffic is low or in other words under light load. The probability of RTSs collision increases with RTS transmission rate with a corresponding decrease in system throughput. FAMA-NCS and FAMA-NPS [14], were introduced for wireless LANs and ad-hoc networks based on a single channel and asynchronous transmissions (i.e. no time slotting) using non-persistent carrier and packet sensing.

Group Allocation Multiple Access

Along similar lines is Group Allocation Multiple Access with Packet Sensing (GAMA-PS) [31]. It uses a form of dynamic reservations to improve efficiency and to ensure that no collisions involving data packets occur. To make a reservation, a node with data to transmit requests to join the “transmission group” by establishing a two-way handshake with its intended receiver using a packet-sensing strategy (i.e. a node backsoff only if it understands entire packet, not after detecting carrier). Once a station has been added to the transmission group, it is able to transmit a packet during each cycle. A position in the transmission group is assigned to an individual node, and the node can continue to transmit in this position while it has data ot send. Every node in the transmission group is required to listen to the channel and maintain its position in the transmission group.

There is no guarantee that a node will be able to hear all of the transmission periods, and hence, the nodes would no longer be synchronized and collisions of data packet could still occur. The solution lacks the ability for collision resolutions resulting in unsuccessful retransmissions of RTSs as in FAMA protocols. The scheme mentions about handling the hidden terminal problem using base-stations making it a centralized protocol causing a lot of overhead on some central authority like a base-station. The scheme lacks exploitation of feasible parallel transmissions.

Collision Avoidance and Resolution Multiple Access

Several collision resolution schemes were proposed establishing a three-way handshake between sender and receiver to attempt to avoid collisions, and resolve the RTS collisions using a tree-splitting algorithm [35]. Based on the simple principle of FAMA but trying to improve the performance of FAMA under high load conditions by doing collision resolution using the tree-splitting algorithm. Since the control packets never collide with data packets in FAMA protocol and also the propagation delays and the duration of RTSs and CTSs are less than the duration of data packet, so if resolving the collisions among RTSs can be done in duration less the data packet duration, we can improvise on throughput. Thats the basis for some of the MAC protocols using Collision resolution.

Garces and Garcia-Luna-Aceves introduced CARMA-NTG [18], a sender initiated protocol which divides the channel into cycles of variable length; each cycle consisting of a contention period and a group transmission period. During the contention period, a node with one or more data packets to send competes for the right to be added to the group of nodes allowed to send data without collisions. Advantages of Group Transmission include that once a station has reserved a position in the group-transmission period, it will be able to transmit at or better than a guaranteed rate. Protocols with transmission group seems to be more stable under heavy loads than CSMA, because it permits stations in the transmission group to send packets independently of new requests for additions to the transmission group. This scheme however, requires that each station must keep track of its position in the Transmission Group and also know how many total nodes are present in the Transmission Group. Each station must know how many data packets does each of those nodes in the Transmission Group have to send and delete those node from Transmission Group when the required number of data packets has been sent. This is required since each node has to align its data transfer after it listens to the data packet from the previous transmitting station is received. Its hard to assure that each node can listen to the transmissions from every other node in the Transmission Group and still suffers some of the disadvantages as mentioned in [31].

A slight modification to the collision avoidance and resolution strategy was CARMA-FS [16], the Collision Avoidance and Resolution Multiple Access First Success. In this protocol any sender that wishes to send data packet sends an RTS and waits for a CTS. If CTS is received, it acquires the floor. However if CTS is not received, the sender as well as the other stations in the system know that a collision has occurred and so execute a common collision resolution algorithm to resolve collisions and then the sender tries again. However it calls off the end of the collision resolution step at the first successful RTS-CTS handshake. It is advantageous to do so, as it will provide faster collision resolution. This scheme assumes a good mutual coordination between all the nodes who want to be potential senders to a particular receiver. This is hard to assure as not necessarily all the nodes (who want to be senders to a common node) can listen to each other. This scheme lacks the feature to successfully handle the hidden node problem as still the CTSs sent by the receiver can get corrupted and will cause troubles in proper execution of the collision resolution algorithm which has to be individually executed by all the nodes in the network as opposed to being controlled by some entity (like receiver) that can listen to all its potential senders. Also the exposed terminal problem still exists.

2.3.2 *Multiple Access with parallel transmissions*

The concept of parallel transmissions in single channel networks was proposed [4, 34]. where parallel transmissions can be initiated by simply aligning DATA transmissions with ongoing transmission without invoking the RTS/CTS exchange.

MACA-P [4] is a set of enhancements to the 802.11 MAC that allows parallel transmissions in many situations when two neighboring nodes are either both receivers or transmitters, but a receiver and a transmitter are not neighbors. Like 802.11, MACA-P contains a contention-based reservation phase prior to data transmission. However unlike many other MAC protocols, the data transmission is delayed by a control phase interval which provides the synchronization between multiple sender-receiver pairs. It requires the Control packets to carry additional information of the start time of the data packet and also to indicate the start time of the ACK packets. So when a node overhears a RTS from some sender not intended for it, if it has data packet to send it can initiate a RTS so as to align the start time of the data with start time of the node's transmission who's RTR was overheard. Both RTS and CTS carry the additional information so that nodes that are either neighbors of the sender or the receiver learn about the schedule DATA and ACK transmissions. The first transmission with which all the other parallel transmissions are synchronized is said to be the Master Transmission.

An important implication of a transmission being a master is that all overlapping transmissions must transfer data packets whose size is less than that of the master transmission. Otherwise, the DATA of an overlapping transfer will interfere with the ACK of the master. This places a large restriction on the amount of synchronization that can be achieved particularly with TCP type of traffic. Achieving such type of synchronization is a bit difficult task. Use of control gap to schedule parallel transmission imposes overhead on the transmissions. Further the length of the control gap limits the number of parallel transmissions significantly.

2.3.3 Receiver Initiated Multiple Access Protocols

The second category includes the collision-avoidance protocols in which the receiver initiates the collision-avoidance handshake.

Multiple Access with the help of polling

In MACA-BI, MACA by invitation [36], the receiver polls one of its neighbors asking if it has a data packet to send. A receiver-initiated collision avoidance strategy is attractive as it can reduce the number of control packets needed to avoid collisions. Though MACA-BI can reduce the number of control packets needed to avoid collisions, it can be easily shown that MACA-BI does not prevent data packets sent to a given receiver from colliding with data packets sent concurrently in the neighborhood of the receiver. Tzamaloukas and Garcia-Luna-Aceves showed that MACA-BI cannot ensure that data packets never collide with other packets in networks with hidden terminals [39]. So with hidden terminals, the protocol fails. They proposed a new scheme [39] where the polling node sends an RTR (Ready To Receive), a small packet to inform other listening nodes that the sender of RTR is ready to receive packets and so they can respond with RTS and also sends an additional packet NTR (No-Transmission Request) telling the polled node not to send any data if it senses the neighborhood busy, in order to handle the hidden terminal problem. Thus, with the help of RTR and NTR, the protocol tries to ensure that there are no collisions of data packets. However the solutions does not take care of the collisions among RTRs from different neighboring polling nodes. The protocol does not takes care of the collision resolution among the RTS's which would give a fair chance to all the nodes to transmit rather than mere Backoff.

Receiver Initiated Collision Resolution with Multiple Channels

Garcia-Luna-Aceves and Garces proposed a receiver-initiated multiple access protocol with collision resolution strategy [20]. Each node with no data packet behaves as a receiver in its assigned channel and starts a Collision Resolution Interval by sending the first RTR. Collisions among RTSs that follow up after the RTR are resolved using the tree splitting algorithm. The protocol operates in multichannel network in which hidden terminals may exist but no co-channel interference as every node is assigned a unique channel atleast within the 2-hop neighborhood. However, with dense 1-hop and 2-hop neighborhood, we might have to use large number of unique channels to prevent co-channel interference and hence, we might see throughput degradation as the packet transmission time increases with the number of channels and the node is found to be busy more often.

2.3.4 MAC protocols with multiple channels

Most of the MAC protocols for wireless networks require carrier sensing. However there is a another class of protocols that take advantage of spreading codes for multiple access to ensure that the intended receivers hear data packets without interference from the hidden nodes. Such protocols are usually referred to as Channel Assignment protocols. These protocols rely on multiple codes assigned to senders or to receivers or are based on dynamic channel selection from available pool of multiple channels. One caveat is that multiple channel networks can be inefficient as not all the channels are used at all times.

Hop Reservation Multiple Access (HRMA) protocol [37] takes advantage of the time-slotting properties of slow Frequency Hopping Spread Spectrum. This protocol uses a common hopping sequence and permits a pair of nodes to reserve frequency hops over which they can communicate without interference. A frequency hop is reserved by contention through a RTS/CTS exchange between sender and receiver. A successful exchange leads to a reservation, and each reserved hop starts with a reservation packet from sender and receiver that prevents other nodes from attempting to use the hop. A common frequency hop (like the Broadcast medium) is used to allow nodes to synchronize with one another, to agree on the current hop of the sequence and the beginning time of a frequency hop. After a hop is reserved, a sender is able to transmit data. This kind of channel hopping guarantees that no data packets from sender will collide with any other data packet from some hidden sender. When a node becomes operational, it has to listen to the synchronization channel to gather information about the current hopping pattern and timing information. Every HRMA slot begins with a synchronization period and this places a overhead for data transfer. Moreover it applies only to slow frequency hopping networks, and cannot be used in systems using other mechanisms such as Direct sequence spread spectrum.

Another Multichannel MAC protocol [32] was proposed which deals with “soft” channel reservation. The idea is similar to FDMA used in cellular systems, the difference being that there is no central infrastructure and thus, the channel assignment is done in a distributed fashion via carrier sensing as in CSMA protocols. If there are N channels, then the protocol assumes that each node can listen to all N channels concurrently. A node wishing to transmit listens to all the channels and searches for an idle channel and transmits on that channel. Among the Idle channel, the one that was last used for last successful transmission is preferred. For a node to be able to listen to all the N channels requires a more capable transreceiver which might increase the cost of the system.

Multi-channel MAC protocol with an on-demand channel assignment for multi-hop mobile ad hoc networks [40] was proposed that assigns channels dynamically, in an on-demand style. It maintains one dedicated channel for control messages and multiple channels for data. Each station has two transceivers, so that it can listen on the control channel and the data channel simultaneously. RTS/CTS packets are exchanged on the control channel, and data packets are transmitted on the data channel. In RTS packet, the sender includes suggested data channel information according to the channel condition around itself. The receiver, on receiving RTS, decides on which channel to communicate and includes the selected channel information in its CTS packet. Then DATA and ACK packets are exchanged on the agreed data channel. This protocol does not requires synchronization and can utilize multiple channels with little control message overhead. When the number of channels is relatively small, one channel dedicated for control messages can be costly. On the other hand, if the number of channels is large, the control channel can become a bottleneck and prevent data channels from being fully utilized.

Along similar lines is multichannel CSMA MAC protocol with receiver-based channel selection [23] for multihop wireless networks, that uses multiple channels and a dynamic channel selection method. It is similar to [40] in the sense that it uses one control channel and N data channels, where N is independent of the number of nodes in the network. However, the difference being that, the channel selection is based on maximizing the signal-to-interference plus noise ratio at the receiver. The protocol suffer the same disadvantages as mentioned [40].

A distributed algorithm for code assignment in multihop networks was proposed [7], based on saturation-degree coloring heuristic that the first nodes to be colored are those that have more colors already assigned to nodes in the neighborhood. The motivation is that these nodes have a more constrained choice and therefore a higher risk that at a certain moment, having all colors been assigned to neighbors, a new color needs to be introduced, and a higher overall number of different colors will be necessary in future steps.

Another dynamic channel assignment scheme [10] was proposed that tries to exploit the channel re-use properties by doing channel reassignment in order to prevent the communicating nodes from suffering from co-channel interference. For channel assignment and reassignment, the nodes have to maintain data structures and have to perform some kind of cost calculations. The nodes maintain a data structure, Neighboring Communication Table (NCT) in its cache which contains the ID of the neighboring hosts, the channel occupied by the 1-hop and the 2-hop neighbors and the cost of those channels. The protocol refers to the cost as, the cost of reassigning a new channel to the neighbor. That is the cost is equal to the number of 1-hop neighbors that will have to change their code after re-assignment of a code takes place. During co-channel interference, one of the two nodes who's transmissions are overlapping decides to change the code based on the cost of changing the code. Both the nodes will evaluate the cost of assigning a new code to itself, exchanges the cost evaluation information and then determine which node will change the code to obtain minimal cost. This selection of the code by changing the packets to identify the minimal cost places a large overhead before the data packets are actually send.

Fairness is a major issue in most of the CSMA strategies. There exists a collection of protocols that look at the fairness aspect of a MAC. Some of these protocols try to address the fairness issues by following a different backoff scheme than binary exponential backoff, which is one of the reason for unfairness in 802.11 MAC.

Ozugur et. al [33] proposed a p_{ij}-persistent CSMA based backoff algorithm. They proposed that each station calculates a link access probability p_{ij} for each of its links based on the number of connections from itself and its neighbors (connection based), or based on the average contention period of its and other stations individual links (time based). Whenever its backoff period ends, station i will send RTS packet to j with probability p_{ij} or backoff again with probability (1 - p_{ij}). This scheme relies on periodic broadcast packets in the time-based approach or on aperiodic broadcast packets in the contention-based approach whenever the network topology changes. Broadcast packets are unreliable to distribute information to neighbors. No one can ensure if broadcast packets can be delivered to all the sending station's neighbors, which makes the performance of this method tightly coupled to the successful distribution of the information in the network.

Another protocol for wireless LANs that attempts to handle the fairness problem in MACAW [9]. In MACAW, additional control packets and a different backoff algorithm named Multiplicative Increase and Linear Decrease with a backoff copy scheme are used to increase the throughput and alleviate fairness problem. The proposal talks about the "per stream" fairness which means that each stream originating from either the same station or different stations should be treated equally and given equal share of the channel capacity. For multiple streams that originate from a station, MACAW keeps seperate queues for each stream and runs backoff algorithms independently for each stream. Maintaining seperate queues for each stream places an overhead on the protocol.

Further an estimation based fair medium access was proposed [5] in which each station will estimate its share and other stations share of the channel and then adjust the contention window accordingly. So, if a station estimates that is has got more share than it should get, it will double its contention window until it reaches a maximum value so that its neighbors can have more chances to recover earlier from backoff procedure and win access to the channel and vice versa. However, if a station estimates that it has only got only its fair share, it will hold onto its current contention window size. Their simulation results show that better fairness is achieved but at the price of throughput. With concurrent transmission between two pairs of edge stations, some of the inner nodes that can listen to both such concurrent transmissions may suffer degradation in throughput as estimating the share of each node becomes inaccurate.

Previous works show that the receiver initiated collision-avoidance MAC protocols can be more efficient than sender-initiated ones. Some of the receiver-initiated current work lacks multiple sub-channels required to allow concurrent transmissions. Some of the multiple channel MAC protocols waste bandwidth because of the strict requirement of a control channel and also require well equipped trans-receivers to listen to all of the subchannels increasing the cost of the system. We present a new medium access control protocol for wireless networks with receiver-initiated collision resolution that minimizes data packets addressed to a given receiver from colliding with any other packets at the receiver and channel division multiple access (a channel sub-division) for allowing multiple nodes within range of the same receivers to transmit concurrently without interference. Compared to some previous works on (sub)channel assignment, our protocol does not requires any separate control channel, but considers all the channels as identical. It requires only one transreceiver per node. We also present enhancements to handle fairness issues that occur because of the strict receiver-initiated behavior of the protocol.

CHAPTER THREE

RECEIVER-INITIATED MAC PROTOCOL ENHANCEMENTS

In this chapter we first discuss the receiver-initiated MAC protocol and identify some of the issues inherently present in a receiver-initiated approach due to the collision avoidance strategy that it follows. We next present the first enhancement of collisions resolutions to the receiver-initiated protocol, wherein the collision resolutions takes place with the help of a deterministic tree-splitting algorithm. For better understanding of the tree-splitting algorithm we give a detailed explanation of the algorithm with the help of an example in the followup section. Next, we identify the need for the second enhancement of subchannel assignment and give a description about the same. Followup is the discussion of the need for a third enhancement, its significance and detailed description of the third enhancement: deliberate transitions of modes between sending and receiving.

3.1 Receiver-Initiated approach

We propose a receiver-initiated protocol where a node in its receiving mode initiates the process of allowing some node to start a data transfer by sending a *Ready To Receive (RTR)*. The control packet RTR is a small packet indicating to the other nodes listening, that the node transmitting the RTR is ready to receive a request for the channel. The sender nodes on the other hand with a data packet to send starts waiting for an RTR from their receiver. On receipt of an RTR, the sender nodes send its RTS to request the access to send the packet. With a densely populated network, there might be multiple senders interested in sending data to a particular receiver which might result in collisions of RTSs at the receiver side. So, when the receiver node detects collisions of RTSs, it does not respond back with a CTS. Here in order to prevent further collisions of RTSs from same sender nodes, the sender nodes in absence of a CTS do a random binary exponential backoff similar to IEEE 802.11 MAC. With random backoffs one node whose backoff time expires first will send the RTS and thus, the chances of collision with other RTSs are reduced. After this a

CTS would follow up from the receiver node and then the normal data and ACK handshake takes place.

Doing a random backoff after every collision would result in wastage of bandwidth particularly at high loads. Every sender node after detecting collision will do a random backoff and after expiration of every such backoff will have to wait again for an RTR from its receiver node. Their intended receiver would not wait for the sender nodes to complete their backoffs and hence, can choose to become a sender at that time if it has a packet to send or will try sending an RTR again. So, it is hard to guarantee that after the expiration of the backoff time the sender nodes would still find their receiver in receiving mode. It is quite possible that the backoff time for all such sender nodes expire while they wait for an RTR: results in further collisions of RTSs from the same senders. This causes unnecessary wastage of time which otherwise could have been used for successful transmissions. So, we can see that with a receiver-initiated approach, the random backoffs don't seem to solve the collision avoidance problem efficiently.

Instead of doing a probablistic approach of depending on random backoff's for resolving collisions, a deterministic approach could serve the same purpose and hence, make efficient use of bandwidth. Also, since we are following a receiver initiated approach, a receiver can very well execute such a collision resolution algorithm, where it decides which node will send data and when, in case of multiple senders. We know that, the propagation delays and the duration of RTSs and CTSs are less than the duration of data packet. Thus, if resolving the collisions among RTSs can be done in duration less than the data packet duration, we can make efficient use of time and improve throughput. Thus, our first enhancement is to add collision resolution of RTSs in the receiver-initiated approach. Since, the receiver is a common entity between all its 1-hop neighbors which are the potential candidates to become senders to this node, doing collision resolution of RTSs at the receiver side would make collision resolution relatively simple and more effective.

3.2 Collision Resolutions by deterministic tree-splitting algorithm

The enhanced protocol uses carrier sensing for initiating the transmissions and we resolve among collisions based on tree-splitting algorithm [35] when collisions among RTSs occur. A node in its receiving mode initiates the process of allowing a node to start a data transfer by sending an RTR. This RTR carries the range of the nodes (1-hop neighbors), that are allowed to send the RTS. Any node that receives this RTR with a data packet to send checks whether it can send an RTS by looking at the range in the RTR and then responds back with RTS in the hope to complete a four way handshake of RTS-CTS-DATA-ACK. If multiple RTSs follow up after sending the RTR, the receiver executes a collision resolution strategy. Inorder to understand the working of the scheme let's begin by defining some of the terms related to this approach.

3.2.1 Definitions and Assumptions

Each node in the network is assumed to have:

1. *Unique Identifier (ID)*: Every node is assigned a Unique Identifier, ID, such that, $0 \leq ID < N$, where N represents the total number of nodes in the system. This node ID is known to all its 1-hop neighbors.

2. *Backoff Stack*: Each node maintains a *Backoff Stack*; it pushes backoff intervals onto the stack and later pops out those intervals as required.

3. *Allowed Interval (LoID, HiID)*: Each node is assigned an *Allowed Interval (AI)* defined by *(LoID,HiID)*, where LoID and HiID is the lowest and highest ID nodes that are allowed to send RTS. Thus *Allowed Interval (AI)* specifies the range of nodes that are allowed to send RTS and it covers a node's all the 1-hop neighbors.

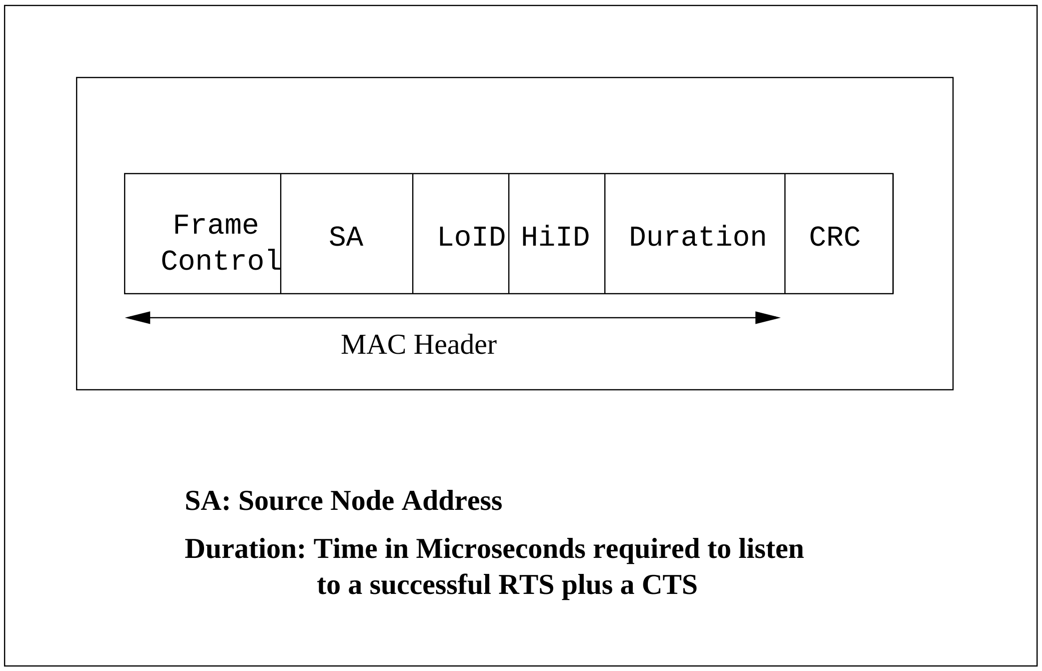

Frame Control
SA
LoID
HiID
Duration
CRC
MAC Header
SA: Source Node Address
Duration: Time in Microseconds required to listen to a successful RTS plus a CTS

Figure 3.1: RTR Frame Format

3.2.2 Basic Operation

Any node that does not have any data packet to send is assumed to be in its receiving mode. It can initiate the receiving period by transmitting an RTR on the channel.

RTR Frame Format

The format of the RTR frame is as shown in Fig. 3.1. The RTR carries the information about:

1. *SA*: Node address sending the RTR, which is the node in receiving mode.

2. *LoID,HiID*: This composes the AI, the range of all 1-hop neighbors those are allowed to send RTS in response to this RTR.

3. *Duration*: This field is set to ($T_{RTS} + \tau + T_{CTS} + \tau$), which is the total time for which the other nodes listening to this RTR will set their NAV. This is the time required to hear

a CTS from the receiver, if it received a successful RTS for a data transfer. So, if such a CTS arrived, the other nodes can update their NAV picking up the duration field information from the CTS packet (The format of CTS packet is as shown in Fig. 2.4). So this duration field specifies the maximum time required by other neighboring nodes to be aware of any transmission following this RTR. However, if there are no senders in which case there will be no RTS and hence the receiver node will not send a CTS. Absence of such a CTS, after time equal to the duration field in the RTR packet, the NAV of the neighboring nodes will expire and they will essentially sense the channel to be idle and can try to acquire the floor.

When the node enters the receiving mode and thus, becomes a receiver for its 1-hop neighbors, it begins by sensing the channel for *Sense Time* given by,

$$SenseTime = T_{RTR} + 2\tau + T_{RTS} \tag{3.1}$$

where T_{RTR} and T_{RTS} are the transmission times for RTR and RTS respectively and τ is the maximum channel propagation time. In order to prevent this node interfering with any ongoing transmission, it should sense the channel for time sufficient enough to tell whether the channel is busy. The *Sense Time* includes essentially the time required either to hear a RTR packet from its neighbor which is in receiving mode given by ($T_{RTR} + \tau$) or to hear an RTS packet from its neighbor which is in a sending mode given by ($T_{RTS} + \tau$). The *Sense Time* helps the other neighboring nodes to avoid some of the collisions with the ongoing transmission that might occur because of the staleness in the Network Allocation Vector (NAV) vector. This staleness occurs due to corrupted RTR or RTS packets. Thus, a receiver node who was sensing the channel, after hearing a RTR (possibly corrupted, from some other neighborhood receiver node) would again start sensing the channel and after listening to a CTS (in case of a single sender) or another RTR (in case of collisions) from this node, would know about some ongoing transmission. Thus, the

Sense Time provides some extra time to a node before it senses the channel idle and helps to update its NAV in case of collisions.

However, complete avoidance of interference cannot be assured if the reception of CTS packets gets corrupted. After receiving a corrupted RTR, the only way of updating the NAV is to listen to a CTS packet which would contain information about the duration of data transfer. If a corrupted CTS packet is received, then the node sensing the channel would find the channel to be idle and hence will send out an RTR interfering with an ongoing transmission.

After *Sense Time*, the channel may be either:

1. **BUSY**: If the channel is found to be busy the node becomes a sender if it has data packet to send or starts sensing the channel again.

2. **IDLE**: The timing diagram for the transmission of a data packet with the proposed protocol when the channel is found to be idle is shown in Fig. 3.2. The receiving or destination node, senses the channel for *Sense Time* and finds it to be IDLE. It sends an RTR and waits for maximum round-trip time, plus the time needed for the destination's RTS to arrive ($T_{RTS} + \tau$). Multiple RTSs from 2 or more source nodes arrive at the destination and hence, it detects collision and carries out a collision resolution scheme based on the tree splitting algorithm divides the allowed interval into 2 halves *Backoff Interval*, (LoID, (HiID+LoID)/2 - 1) and the *Allowed Interval*, ((HiID+LoID)/2, HiID). The *Backoff Interval* is pushed onto the *backoff stack* and the updated *Allowed Interval* is sent in the next RTR. Thus the window of granting access to senders is reduced by half.

   Let's say, that after dividing the interval into two halves, only source 2's ID falls within the *Allowed Interval* as shown in Fig. 3.2. The destination node again sends out the next RTR with the newly calculated *Allowed Interval*. As, only source 2 finds its ID in this new reduced AI, it responds back with an RTS. Source 1 on the other hand, waits for another RTR from the destination node. Since a single RTS is heard, a normal four way handshake of data

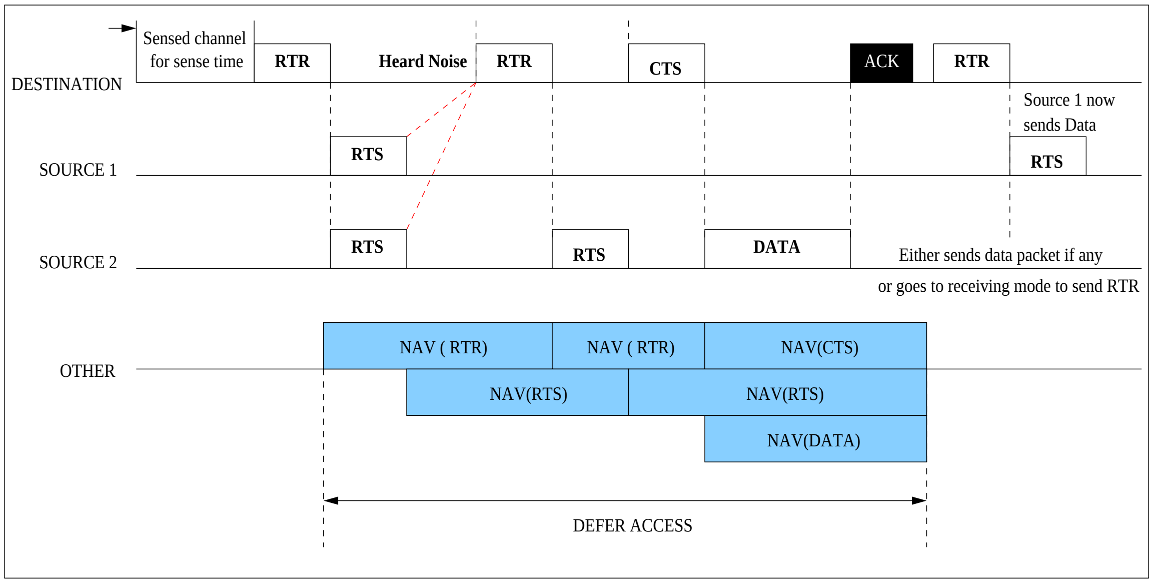


Figure 3.2: Transmission of an MPDU with receiver-initiated protocol enhancement

transfer now takes place. When the data transfer with source 2 is completed, the destination node pops out the backoff interval from the stack and sends out a new RTR with this interval as the new allowed interval. At this point, source 1 will consider itself as a candidate for data transfer as its ID will fall in the range of the AI sent in this RTR and hence, will send out an RTS. Now the data transfer with source 1 can take place as shown out in Fig. 3.2.

If the first collision resolution step again results in collisions, then the above mentioned step of dividing the allowed interval is repeated and so the stack keeps on growing. This sequence of collision resolution steps are defined as a **Collision Resolution Interval (CRI)**. CRI is said to be complete when the backoff stack becomes empty. However, if after sending the RTR, a timeout occurred and nothing is heard and also, if the backoff stack is empty, the node essentially backsoff and seeks again to sense the channel. This timeout after sending an RTR implies two things:

1. **No Senders** : This is the simple case wherein there are no senders to the node. At this point the node backs off and then, later again tries to send RTR and looks for any senders intended to send data to this node.

2. **Collision of RTRs** : Though every receiver node senses the channel before sending an RTR, RTRs are prone to collision due to propagation delays. Two or more node can sense the channel to be idle and thus have an illusion that they are the only ones using the channel, which may not necessarily be true. Any 2 RTRs send within the time difference of propagation delays are prone to collide. In this case, any sender node that can listen to both such receiver nodes sending RTRs, will detect noise, and hence will not respond back with any RTS. So, even if senders were present, a timeout will occur at the receiver end after sending a RTR.

   It is difficult for a receiver node to distinguish between two such cases where a timeout occurred after sending an RTR, essentially because it cannot detect the channel for some other ongoing transmission while it is transmitting. Random backoff's after such a timeout helps resolving such collisions among RTRs. With random backoff, only one of these potential receiver node will gain access to the floor first (whoever finishes its backoff early) and the others would sense the channel busy due to RTS or data packet from some sender node to the former receiver.

Since CRI is defined as a sequence of collision-resolution steps, each initiated by an RTR, the duration of a CRI varies according to the type of the collision resolution step. The size of the CRI is a function of the number of senders. It will be large if there are too many senders wishing to send at the same time because the number of steps required to resolve collisions will be more in that case. However on the other hand with a handful of senders wishing to communicate, the CRI might just converge a little faster. Thus, we can see that collision resolution is a deterministic approach wherein we know when all the sending nodes are going to get their chance and when will the entire collision resolution finish (this could be measured by the depth of the tree - logarithmic in the number of 1-hop neighbors, at most).

A CRI is said to be complete when the receiver has resolved all the collisions and also that its stack is empty. When this happens, the node in the receiver state can make a transition to the sender state, if it has any local data packet pending. However if at this point the node does not have any data packet pending, then it can send another RTR initiating a new CRI. Thus, once a node initiates a new CRI, it is bonded to remain as a receiver until the end of the current CRI. If at the end of the CRI the node has a data packet to send, then it starts monitoring the channel for an RTR from its intended receiver and upon receiving one, becomes a sender participating in the CRI of its intended receiver. On the other hand, if at the end of the CRI the node does not have a packet to send, then it remains as a receiver initiating a new CRI.

A node is always in its receiver state until it has a data packet pending in which case it makes a transition to the sender state. Such a sender node scans the channel for a maximum of *Wait Time* for an RTR from its intended receiver. The *Wait Time* is an indication of the maximum time a node might have to wait to listen to next RTR if it had just missed out an RTR. This is the total time taken for a receiving node to send a CTS plus the propagation delay of the channel ($T_{CTS} + \tau$), the total transmission time for a data packet plus the propagation delay ($T_{DATA} + \tau$), time for an ACK to follow plus the propagation delay ($T_{ACK} + \tau$) and the time for the next RTR to be received ($T_{RTR} + \tau$). Thus, the *Wait Time* is given by,

$$WaitTime = T_{CTS} + T_{DATA} + T_{ACK} + T_{RTR} + 4\tau + (ID + 1) * 0.001 \tag{3.2}$$

where T_{CTS} and T_{DATA} are the transmission times of CTS and DATA packets respectively and ID is the Unique ID of the node. If the node's intended receiver is doing a data transfer with some other node, then after this *Wait Time*, another RTR will follow up to which this node can respond or if the intended receiver is doing collision resolutions, then this node is sure to get another RTR within this *Wait Time*. The term (ID + 1)*0.001 gives the random nature to the *Wait Time* dependent on the node's ID, so that all the nodes would have slightly different Wait Times. Also, the term (ID

+ 1)*0.001 provides some extra waiting time for a sender node rather than just specifically waiting for a maximum time of data transfer.

If an RTR is heard during the waiting period, the node competes to acquire the floor extending its duration as a sender until it is successful in acquiring the floor. This floor acquisition is marked, when the sender node receives a successful CTS destined for it in response to its RTS. The sender node waits for one maximum round-trip time plus the time needed for the destinations CTS to arrive ($T_{RTS} + \tau$). If a successful CTS is received within this time, then the node acquires the floor and transmits its data packet. However, if the CTS is not received, the node detects a collision and thus, starts waiting for another RTR from its intended receiver. If within the wait time the node does not hear a valid RTR, the node transitions back to the receiver state and remains in this state until the end of its own CRI.

This deliberate transition from sending to receiving state is very essential as we don't want a node to keep on waiting for an RTR indefinitely which might occur if the sending node's intended receiver is itself in its sending mode waiting for an RTR packet. This might cause a lockout as both the nodes will not be able to fulfill each others requirements and might cause unfairness at some of the other nodes, while they are busy waiting. We give slightly different *Wait Time's* to different nodes by using the offset term (ID + 1)*0.001 so that we can resolve the conflict between two nodes who started waiting for each other at the same time. So, one of these node's *Wait Time* will expire first and hence, will send out an RTR giving the other node a chance to compete for the floor by sending RTS.

3.2.3 Advantages and Issues

By making the medium access control receiver-initiated, we controlled some of the interferences caused to ongoing data packet transfer due to the presence of hidden nodes. No sender node can now interrupt an ongoing data transfer from some hidden node to its intended receiver by sending an RTS, one of the main features of hidden node problem. This is guaranteed as every sender node waits for an RTR from its intended receiver and thus, will send an RTS only if its receiver node is in a mode of initiating a new floor acquisition request to its senders (by sending an RTR). By doing tree splitting for resolving collisions, we make efficient use of bandwidth by not wasting significant amount of time in backoffs and using that time for successful data transfers by following a deterministic approach.

However, the hidden node problem is not completely eliminated. By introducing an extra control packet RTR, we have introduced inefficiencies in the network as now every node with no packet to send will send out an RTR and hence, more collisions of RTR packets will occur. Because of these collisions of RTRs, chances of the NAV becoming stale increases. Though, a sender node will not interrupt an ongoing transmission to its intended receiver, it can still interfere with some ongoing transmission to its 1-hop neighbor receiver node. Also, due to redundant NAV any node initiating an RTR can interfere with an ongoing transmission. As mentioned above, CTS from some neighboring receiver node (not a node's intended receiver) is essential to update the NAV for an ongoing data transmission. However, if the CTS got collided with an RTR from some other neighbor, then the NAV would be incorrect. In this case, this node with a data packet to send will find the channel to be idle and hence, will respond back with an RTS on hearing an RTR from its intended receiver. Thus, corrupting the data packet being received by this node's some neighboring receiver node.

3.3 Deterministic Tree Splitting Algorithm

This section describes in detail the tree splitting algorithm used for collision resolutions as explained in Sect. 3.2.2

Time-division multiple-access (TDMA), where the channel is divided into time slots and every node is assigned an unique slot, is an example of a collision-free protocol that work well under high loads but under-utilize the channel bandwidth under low loads resulting in larger delays. ALOHA, Carrier Sense Multiple Access (CSMA), and Carrier Sense Multiple Access with Collision Detection (CSMA/CD) are examples of collision-based protocols that utilizes the channel efficiently under low loads. But the throughput reduces drastically at high loads due to an increase in the number of collisions. Some of these approaches use random backoff procedures in case of collisions, thereby resulting in poor channel utilization. An efficient way to achieve a good utilization both at low and high loads is to dynamically allocate the channel bandwidth to the contending nodes by resolving collisions.

Tree splitting was one of the first techniques proposed for collision resolution. When a collision occurs, the colliding nodes are split into two groups, Group 1 and Group 2. Nodes present in Group 1 are allowed to transmit, followed by nodes in Group 2. If a collision again occurs in the first level group, then further second-level groups are created. The above procedure is used recursively until all the collisions are resolved. The tree-splitting technique could be based on probabilistic or deterministic methods. In probabilistic methods, the nodes choose to be part of a group at random, while in deterministic approach, the nodes are assigned unique identifiers and the collisions are resolved using these identifiers. As described in Sect. 3.2.2, we are going to use a *deterministic tree splitting approach*, where the control packet RTR is used to resolve the collisions.

3.3.1 Algorithm Description

As pointed out earlier, every node is assigned an ID and an *Allowed Interval (AI)*, defined by *(LoID, HiID)*. The *AI* is sent as a part of the control packet RTR, so that any node that listens to this RTR can decodes this information. The *AI* grants an access to all those nodes who's ID falls within this range, to send an RTS.

Collision resolution steps (or Collision Resolution Interval (CRI)) can be classified as *Success*, *Idle* and *Collision* based on the number of RTSs generated. If multiple RTSs are received, this indicates a step of *Collision*, where the receiver node divides the *AI* into two halves. The first interval called as the *Backoff Interval*, (LoID, (HiID+LoID)/2 - 1) and the *Allowed Interval*, ((HiID+LoID)/2, HiID). The *Backoff Interval* is pushed onto the *backoff stack* and the newly calculated *AI* is sent in the next RTR. Depending on which of the two subintervals is pushed on the backoff stack, the priority is decided. In our algorithm, we assume that the key being pushed is the *Backoff Interval*, i.e the nodes with higher node ID's are given priority over nodes with lower node ID's. Therefore, the lower subinterval is pushed first on the stack followed by the higher interval. This process is repeated until all the collisions are solved. If the first collision resolution step results again in collisions, then the above mentioned step is repeated and so the stack keeps on growing.

Collision resolution step achieves *Success*, if after sending an RTR, if one successful RTS was received. In that case after the data transfer is complete, the receiver node pops out the top of the stack and sends out a new RTR with this interval as the new Allowed Interval. Since the previous RTR did not result into collisions and also just one RTS was received implies that there were no more senders in that *AI*, so we need not send the same RTR again nor do we have to divide the *AI*. Therefore, popping out the top of the stack ensures that we are not skipping any senders. An *Idle* step in collision resolution occurs, when after sending the RTR, none of the nodes in the *AI* had a packet to send and so have not responded back with a RTS. In this case the receiver node continues popping the stack.

The algorithm repeats these three collision resolution steps, until all the collisions have been resolved. As soon as the *backoff stack* becomes empty the receiver node is finished with this Collision Resolution Interval (CRI).

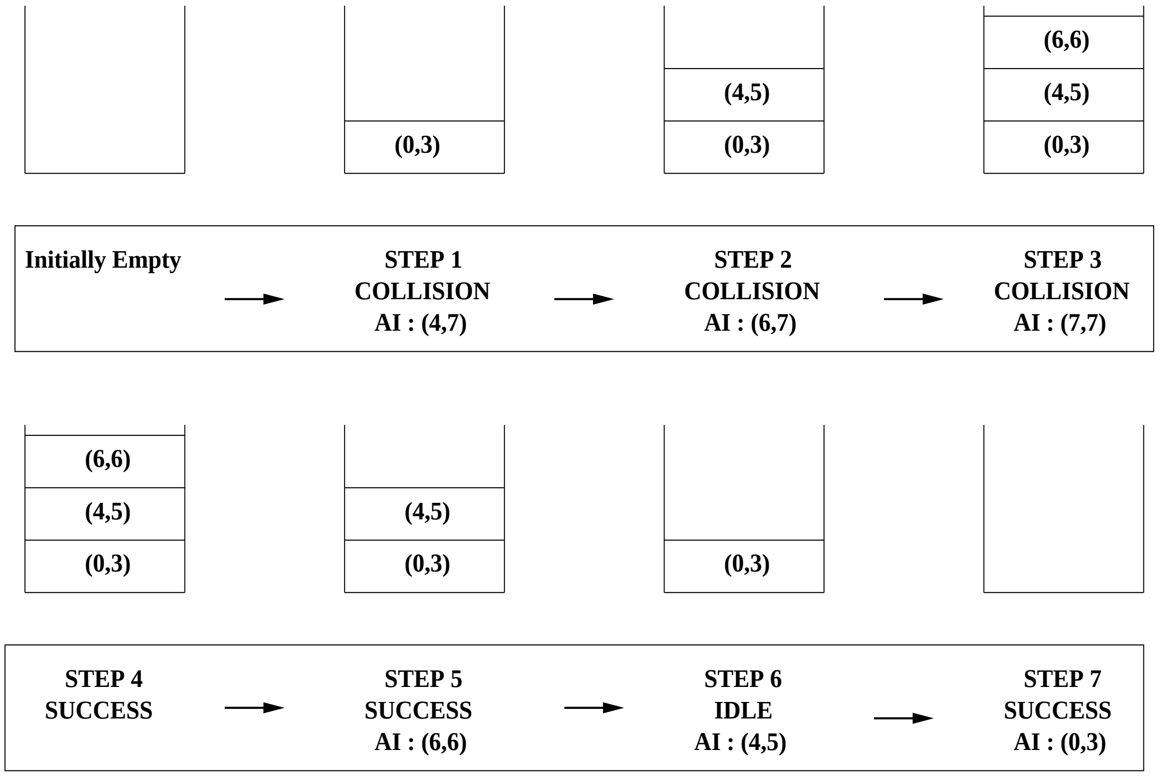


Figure 3.3: Example : Tree splitting Algorithm - Backoff Stack of Node 9

3.3.2 Example for Algorithm's Operation

In deterministic tree splitting algorithm each node has a distinct position in the leaves of the binary tree based on its ID. The root node of the tree is the node in the receiving mode, initiating the CRI and the leaf nodes of the tree are the 1-hop neighbors of the receiver. Consider a network consisting of 20 nodes. Let's say, a node with ID 9 is a receiver node and has an initial *AI*, defined by (0, 7). Suppose that node with ID's 1, 6 and 7 have data packets to send to node 9. The tree splitting algorithm proceeds as follows:

1. Node with ID 9, sends an RTR with (0, 7) as the allowed interval. All the potential senders 1,6 and 7 finds their ID into the *AI* and hence send an RTS. This results in step *Collision*. Node 9, divides the *AI* into 2 halves, pushes (0, 3) onto the backoff stack and sends out another RTR with the new *AI* as (4, 7). The backoff stack now looks like as shown in

Fig. 3.3.

2. Nodes 6 and 7 finds their ID in *AI* and hence, send RTS again which collide. The second collision resolution step again results in *Collision*. The *AI*, (4, 7) is further divided into (6, 7) and (4, 5) where (4, 5) is pushed onto the stack and (6, 7) is sent out with the next RTR.

3. RTSs from 6 and 7 again collide. This results in dividing off the *AI* as (7, 7) and (6, 6). The stack after this point looks like as shown in Fig. 3.3.

4. A new RTR is sent out with (7, 7) as the *AI*. Now only node 7 is within the *AI*, so a *Success* step occurs with node 7 sending a data packet to node 9.

5. This Success step is followed by popping off from the stack, whereby the interval (6, 6) is popped out and sent out with the next RTR. This results in successful transfer of data packet from node 6 to node 9.

6. Again the stack is popped out and interval (4, 5) is sent out in the next RTR, but this results in *Idle* step as none of the nodes in this interval have packets to send. Thus, this interval goes idle. *AI*, (0, 3) is popped out and sent out with next RTR.

7. Node 1, finds itself in the *AI* and hence, completes a successful data transfer with node 9. Node 1, being the last one had to wait for a long time, so, the wait time calculated by each sender node to wait for a RTR should be sufficient enough. This is guaranteed since every sender node re-initializes its *Wait Time*, given by Eq. 3.2, on every RTR it hears from its intended receiver during one entire CRI.

The tree splitting described above, had three collision slots, three successful transmissions slots and one idle slot. In this algorithm, the nodes that are part of the stacked entries are allowed to join the CRI when a collision resolution step is in progress. For example, if a message arrives at node 4 when Step 2 of the algorithm is in progress, it is allowed to participate for channel access in Step

6. In such a case, the total number of contending nodes is considered to be four, instead of three. By doing this, we increase the channel utilization as many nodes will get a chance to participate in the CRI and thus, complete a data transfer.

3.4 Subchannel Assignment

Further, we augment the proposed receiver-initiated MAC protocol with collision resolutions to use multiple sub-channels by following a subchannel assignment strategy. The proposed protocol in presence of multiple (sub)channels eliminates co-channel interference and mitigates the effect of contention interference. The objective of dividing the single channel into multiple subchannels is to further minimize the hidden terminal problem, thereby controlling the data packets drop and also to solve the exposed node problem thereby, allowing concurrent transmissions. Both these should contribute in further throughput improvements.

3.4.1 Need for Subchannel Assignment

In the Sect. 3.2.2 we discussed that following a receiver-initiated approach with collision resolutions, does not eliminate the hidden terminal problem. There are data packets lost because of the difficulty in the NAV vector updation in presence of collisions of RTRs. This situation occurs since, all the receiver nodes send RTRs in the same channel and they do this irrespective of the presence of any senders to them. So, the total flow of RTRs in a particular channel is very large. In order to control the hidden terminal problem, it would be ideal if all neighboring nodes send RTRs in separate distinct (sub)channels. In that case we would experience less number of RTR collisions and the chances of the NAV being correct and upto-date will increase. Thus, we may control the number of data packets being dropped because of interference from neighboring nodes and hence, see a rise in throughput. Also the proposed protocol with collision resolution mechanism does not handle the exposed terminal problem, thereby not allowing any concurrent transmissions. In the presence of multiple subchannels, we can further achieve throughput enhancement with many node-pairs initiating data transfer.

A channel that is occupied by a pair of hosts cannot be used by another pair of hosts if their communication ranges overlap. A pair of hosts within communication range of each other must communicate on different channels in order to complete their data transfers. This leads us to multiple sub-channels, obtained by subdividing the channel, where multiple nodes within range of the same receivers can transmit concurrently on different subchannels without interference. Multiple channels exhibit better delay characteristics than single channel networks [11, 30] and have better fault tolerance against fading and noise [11, 29]. Concurrent transmissions can take place in multichannel network because of reduced interference from neighbors. Also with one channel, the chances of the NAV being stale is high due to corrupted reception of control packets which leads to more data packets being dropped. So by introducing multiple subchannels, the co-channel interference decreases and hence, we can control the percentage of drops in data packets and may make more efficient use of bandwidth.

One drawback of multi channel networks is that not all the channels may be effectively used. In a multichannel network, the bandwidth gets subdivided with every subchannel now receiving the shared part of the bandwidth. Simplistically, we should have an N node network consisting of S subchannels with S = N, where every node is assigned a separate subchannel so that the co-channel interference is completely eliminated. Ideally, the number of subchannels used must be the radio chromatic number of the underlying network. However, in either case, the bandwidth of each subchannel is significantly reduced by a factor of N (in the simple case) or the radio chromatic number. This kind of network will allow concurrent transmissions thereby increasing the channel utilization and (hence) the throughput, but the bandwidth subdivision puts severe restriction on this throughput enhancement.

As all the subchannels share the fixed channel capacity allocated to the network, the number of subchannel used must not exceed a given bound. Also, multichannel networks require either transmitters or receivers to communicate over a multitude of subchannels and a node needs to know which subchannel to use in transmitting or receiving packets. The hardware of a node can be designed to transmit on only a fixed number of subchannels. There is an *optimum value of S* where maximum throughput is achieved along with reduced level of interference and then, the throughput starts choking as we increase the number of subchannels. This happens because the packet transmission time increases with number of subchannels and hence, the nodes are found to be busy more often. On the other hand, on a single channel network, with entire bandwidth, the throughput suffers because of the exposed node and hidden node problems. So, an efficient subchannel assignment strategy should attempt to minimize the number of subchannels used while eliminating the exposed node problem and at the same time gaining sufficiently high throughput.

3.4.2 Network Model and Problem Statement

A Wireless Local Area Network is modeled as a dynamic directed graph (digraph) *G = (V, E)* with *V* = $\{v_1, v_2, \ldots v_n\}$, the (current) set of nodes in the network. Each vertex v_i in *V* has a configuration defined by its current position coordinates *(*(x_i, y_i) in a 2-dimensional network) and a (variable) maximum transmission power range r_i which specifies the maximum distance from (x_i, y_i) that other nodes in the network can hear or are affected by interference from its transmissions. The set of edges *E* = $\{(v_i, v_j) : i \neq j \;\&\; d_{ij} \leq r_i\}$ consists of directed edges of the type $v_i \leftrightarrow v_j$ if and only if v_j is within $v_i's$ range, that is, if the distance d_{ij} between v_i and v_j is less than r_i.

The Subchannel Assignment problem is to assign a subchannel, *S* (equivalently, a color to every node, as we have modelled the network as a graph), which is essentially a positive integer, to each node in the network. The main intention in assigning a suchannel is to satisfy the following conditions throughout the network.

1. Condition C_1: For every edge $(v_i, v_j) \in E, S_i \neq S_j$. This condition takes care that no two,

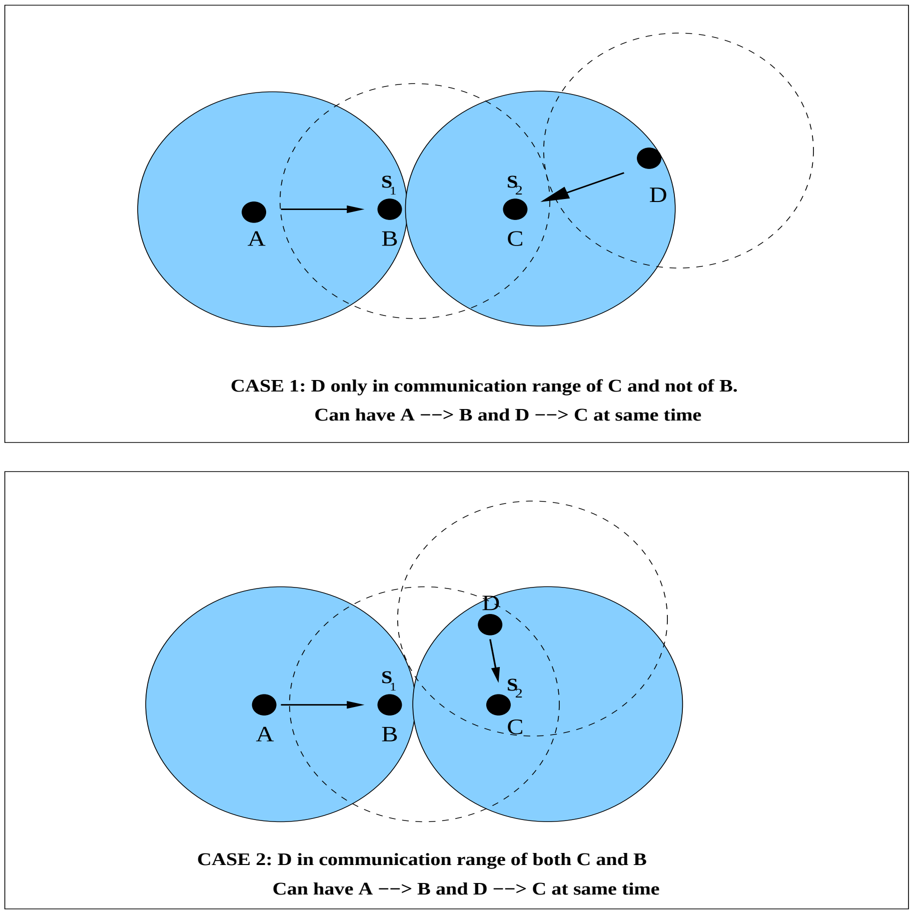

S_1
S_2
A
B
C
D
CASE 1: D only in communication range of C and not of B.
Can have A --> B and D --> C at same time
D
S_1
S_2
A
B
C
CASE 2: D in communication range of both C and B
Can have A --> B and D --> C at same time

Figure 3.4: Exposed node problem scenario because of two neighboring nodes

1-hop apart nodes are assigned the same subchannel.

2. Condition C_2: For every pair of edges $(v_i, v_k), (v_j, v_k) \in E, \&i \neq j, S_i \neq S_j$. This condition specifies that no two, 2-hop apart nodes are assigned the same subchannel.

Subchannel assignment suitable with our methodology of MAC will be a receiver oriented subchannel assignment satisfying the condition that every node is assigned a distinct subchannel than its 1-hop and 2-hop neighbors. So the sender node will be required to move to the subchannel of the receiver node in order to send a packet to that node. Satisfying condition C_1 allows 2,

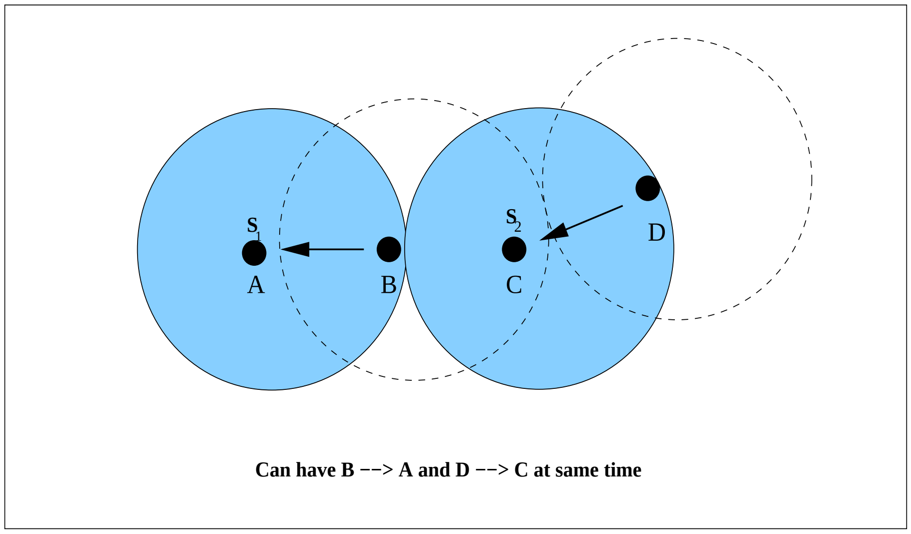


Figure 3.5: Exposed node problem scenario because of two, 2-hop apart nodes

neighbors to become receivers simultaneously and hence receive packets concurrently, solving the exposed node.

Fig. 3.4 explains this scenario. If B and C are assigned different subchannels, say S_1 and S_2 respectively, then transmissions from $A \rightarrow B$ and $D \rightarrow C$ can take place concurrently. In Fig. 3.4, Case 1 shows the scenario when D is not in communication range of B and Case 2, where D can listen to packets from both B and C. Node D with a packet for C will move to the subchannel S_2 for data packet transmission. In Case 1, since B and C will be communicating on different subchannels, concurrent transmissions can take place. In Case 2, since node D will be able to listen to packets only from C and not from B, interference can be avoided.

With a receiver oriented subchannel assignment, satisfying condition C_2 further helps in controlling the exposed node problem. If we violate condition C_2, and say that nodes A and C (2 hop neighbors both in receiver mode) are assigned same subchannel (Fig. 3.5), then a node B (that can hear both A and C) which is transmitting to A might interfere with the transmission from D to C. Fig. 3.5 captures such a scenario. We can see that if A and C (2-hop neighbors) are assigned same subchannel, then transmissions from $B \rightarrow A$ and $D \rightarrow C$ cannot take place at the same time. Because, the control packets from node C (RTR or CTS) will collide with the packets from A (RTR or CTS) at node B. However, if A and C have different subchannels, say S_1 and S_2 respectively, then $B \rightarrow A$ and $D \rightarrow C$ transmissions can take place successfully, as node B will receive only the control packets from node A on S_1 and not from C, which is communicating on S_2. Thus, allowing concurrent transmissions and minimizing the exposed node problem.

3.4.3 Subchannel Assignment Algorithm

Several heuristics exist [7] to assign subchannels to nodes. We use a simple heuristic by choosing the nodes with the highest degree first to assign a subchannel (ties are broken arbitrarily). The highest degree node would be assigned subchannel 1. As we proceed we ensure to assign a subchannel satisfying conditions C_1 and C_2. Since, there is a bound on the number of subchannels (say, S_{MAX}) that can be used for assignment, we may not be always able to assign distinct subchannels to all of the 1-hop and 2-hop neighbors of a node or in other words we might not always be able to satisfy the conditions C_1 and C_2. Sometimes we might need to reassign a subchannel that already appears in a node's 1-hop or 2-hop neighborhood. Hence, we need an objective function that can model the subchannel assignment in a node's neighborhood and allow us to pick up the most suitable subchannel available. We define a *Composite function, C* that decides what subchannel to be assigned to a particular node. The *Composite function* is defined as:

$$C[i] = \alpha T_1[i] + (1 - \alpha) T_2[i] \tag{3.3}$$

where, $T_1[i]$ and $T_2[i]$ are the number of 1-hop and 2-hop neighbors of a particular node that are assigned subchannel *i*. α is the bias factor ($0 \leq \alpha \leq 1$) that decides whether to give priority to 1-hop or 2-hop neighborhood. In order to handle the exposed node problem, we need to satisfy both the conditions C_1 and C_2 and thus, in cases when all the available subchannels are assigned to a nodes 1-hop and 2-hop neighborhood, we will have to reassign a particular suchannel. However, a choice of such a subchannel for reassignment should be based on the fact that, a subchannel that appears minimum number of times in 1-hop and 2-hop neighborhood be used for reassignment. Thus, while assigning a subchannel to a node, first we calculate the number of times each subchannel is assigned in the node's 1-hop and 2-hop neighborhood for all the available subchannels from 1 to S_{MAX} (upper bound on the subchannels available) and then pick up the subchannel with minimum assignment.

In Eq. 3.3 , the Composite function consists of two parts. The first part contributing to the calculation of a subchannel's assignment in 1-hop and the second part contributing for 2-hop neighbors. Together, they help in selecting a subchannel that appears minimum number of times in both 1-hop as well as 2-hop neighborhood. In some cases where the 1-hop neighborhood is more dense than the 2-hop neighborhood, we might want to select a subchannel that appears more in 2-hop neighborhood and thus, ease our decision making process. Thus, we might have to choose a value of $\alpha > 0.5$ in order to make more accurate subchannel assignment.

However, mere counting the number of times a particular subchannel is assigned in a nodes neighborhood has no meaning if a node has more 2 hop neighbors than 1-hop or vice-versa. In some situations, a subchannel assigned less number of times in 1-hop neighborhood sounds a better choice than a subchannel that is assigned more number of times in 2-hop neighborhood. However, if the 2-hop neighborhood is more dense than the 1-hop neighborhood, choosing the latter subchannel might make more sense if we are trying to assign a subchannel that appears minimum number of times in both the 1-hop and 2-hop neighborhood. We need to take into account how dense the 1-hop or 2-hop neighborhood of node is. We do normalization of the *Composite function* by dividing the composite term with the total number of 1-hop and 2-hop neighbors given by,

$$C[i] = \alpha T_1[i]/T_1 + (1-\alpha)T_2[i]/T_2 \tag{3.4}$$

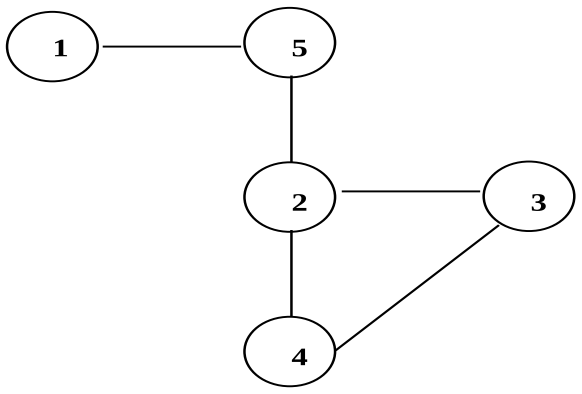


Figure 3.6: Example: Subchannel Assignment - 5 node random network

where, T_1 and T_2 are the total number of node's 1-hop and 2-hop neighbors respectively.

3.4.4 Example: Subchannel Assignment

Fig. 3.6 shows a random network consisting of 5 nodes. We have modelled the network as a graph *G* with (*V* = *1, 2, 3, 4, 5*). By equating subchannels with colors, the problem can be graph-theoretically formulated as that of coloring the vertices of the graph satisfying the conditions C_1 and C_2. Table 3.1 shows the degree of the nodes.

Node	**Degree**
2	3
3	2
4	2
5	2
1	1

Table 3.1: Degree of the nodes

First, we show how the subchannel assignment algorithm works with $S_{MAX} = 4$. The sub-channel assignment found by the proposed subchannel assignment algorithm with $S_{MAX} = 4$ is as depicted in Table 3.2. With $S_{MAX} = 4$, the assignment was fairly simple as the maximum number of subchannels required to assign distinct subchannels satisfying conditions C_1 and C_2 is 4. Node 2, with highest degree 3 is assigned subchannel 1. Now since, nodes 3,4 and 5 all have a degree of 2, these ties are broken arbitrarily. So, node 3 is chosen (randomly), and is assigned a subchannel 2 as 1 is already present in its 1-hop neighborhood with node 2. Similarly, node 4 and node 5 are assigned subchannels 3 and 4 respectively. For node 1, subchannel 2 was the accurate choice satisfying conditions C_1 and C_2. Thus, with $S_{MAX} = 4$, we can satisfy both the conditions C_1 and C_2 and hence, do an accurate subchannel assignment.

Node	**Subchannel Assigned**
2	1
3	2
4	3
5	4
1	2

Table 3.2: Subchannel Assignment with $S_{MAX} = 4$

Node	**Subchannel Assigned**
2	1
3	2
4	3
5	2
1	3

Table 3.3: Subchannel Assignment with $S_{MAX} = 3$

Let's consider a case with $S_{MAX} = 3$. It is worth nothing, that in this case, reassignment of a particular subchannel becomes necessary and the calculations by Composite function, C, becomes significant. The subchannel assignment found by the proposed subchannel assignment algorithm with $S_{MAX} = 3$ is shown in Table 3.3. Again, node 2, 3 and 4 are assigned subchannels 1, 2 and 3 respectively following the same pattern of assignment as mentioned in the above case with $S_{MAX} = 4$. However for node 5, now reusing an already assigned subchannel becomes a necessity. Table 3.4 shows the composite function calculations for node 5 with α = 0.5.

subchannel (i)	$T_1[i]$	$T_2[i]$	T_1	T_2	**C**
1	1	0	1	2	0.5
2	0	1	1	2	0.25
3	0	1	2	2	0.25

Table 3.4: Composite function calculation for node 5

As can be seen from Table 3.4 that the minimum value of Composite function, C, appears with subchannels 2 and 3. We pick up subchannel 2 and assign it to node 5. Node 1 finds subchannel 1 and 2, both assigned in its 1-hop and 2-hop neighborhood and thus picks up subchannel 3 satisfying conditions C_1 and C_2.

3.5 Transitions between Sending and Receiving Modes

In Sect. 3.2, we discussed a receiver-initiated MAC protocol that tries to mitigate the effect of hidden terminal interference by doing collision resolutions following a deterministic tree splitting approach. The interesting feature of this protocol is: it combines a known collision resolution algorithm (tree splitting) with RTS/CTS exchanges and provides a stable and very efficient protocol that does not require time slotting or availability of base stations capable of detecting multiple simultaneous transmissions. The decision of sensing the channel idle before initiating any transmission by a control packet (RTR in our case and RTS in IEEE 802.11) is not just based on NAV (as in IEEE

802.11 MAC). But, the transmission of first RTR (which initiates the floor acquisition process) is delayed sufficiently to avoid interference with ongoing transmission. This is helpful in cases when the NAV is stale due to receipt of corrupted RTR or RTS packets. However, as explained in Sect. 3.2, this approach cannot fully control hidden terminal problem and hence, requires further improvements to control the collisions of unproductive RTRs from neighboring nodes, transmitting on the same channel.

We further enhanced the protocol to work in multiple channels, thereby controlling collisions of RTRs and exploiting the feature of concurrent transmissions supported by multiple channels as described in Sect. 3.4. With multiple channels, we expect better throughput performance because of simultaneous transmissions from neighboring node-pairs and also, better packet transmission delay.

3.5.1 Need for deliberate Mode transitions

The nature of the receiver-initiated approach poses a certain requirement on every node, that could have some side-effects on the performance of the protocol. In our protocol, any node that is not having a data packet to send becomes a candidate of being in receiving mode. After moving in receiving mode, it senses the channel for some time given by Eq. 3.1 and thus, acquires the floor. This receiver node has to invite its neighbors actively to see if they have packets for it. This invitation is sent in the form of RTR to which the (interested) sender nodes respond. This is the time when neighboring nodes compete, if necessary, to send data to it. The rationale behind such protocols is that collision avoidance is more important at the receiver's side, given that the receiver needs to receive relatively long data packets successfully and such packets are more vulnerable to interference. It is reasonable to think that if the sender nodes (nodes being invited) always have packets for the receiver node (node sending invitation), receiver-initiated protocols with proper collision avoidance schemes can outperform sender initiated protocol. A node starts as a receiver and would become a sender as soon as it receives a packet to send. This node would continue behaving as a sender, until its transmission queue is empty, in which case it again moves to its receiving channel initiating a new CRI. This works good in simplistic cases, where every node receives some limited number of packets to send and thus, spends a balanced amount of time both sending as well as receiving.

Consider a scenario wherein the network is heavily loaded (high data rate) such as video conferencing. In such scenario, a node in sending mode will have its transmission queue always loaded with packets to send. In that case, the node would always keep sending data packets, thus, spending most of its time in the sending mode and less time in its receiving channel as a receiver. This will cause this node to behave as a *poor listener*. In real time services, where every node has a data packet to send to every other node, there might be several other nodes with data packets for this *poor listener*. This causes all these nodes to starve resulting in either little or absolutely no traffic flow between these nodes and the poor listener. This causes unfairness amongst nodes and if the number of poor listeners are more then, fairness issues becomes even more serious.

The network throughput in this case is not adversely affected as the traffic on some flows keep going and does not come to a stand still. We might expect to see some degradation in throughput as the sender nodes to poor listeners would waste some time (multiples of Wait Time given by Eq. 3.2) listening to their intended receivers receiving channel for an RTR. What suffers most, is the *Quality of Service assurance*. It is difficult to provide *Quality of Service assurance* without fairness problem solved. Presence of fairness problems hinders the deployment of high-demand applications that require Quality of Service assurances.

Similar problem of unfairness is not present in the standard 802.11 MAC protocol. 802.11 MAC follows a sender initiated approach and a particular node does not assume to have fixed roles: a sender does not wait for a permission from its intended receiver in order to send the RTS. So even under heavy loads, a node with its transmission queue full would still respond back with a CTS to some sender node's RTS, if it has finished its current data transmission and the channel is idle. However, despite its popularity, the standard IEEE 802.11 MAC protocol also suffers severe fairness problems [4, 5].

The commonly used binary exponential backoff (BEB) scheme, despite its robustness against repetitive collisions, can aggravate the fairness problem. In the 4-way handshake, if a node does not receive CTS response for its RTS request, it treats this as a collision and hence doubles its backoff window. This is irrespective of the status of the destination node. A node may defer to send back CTS if any other node in its vicinity has reserved the channel by sending an RTS or CTS to some other node. This results in success for one node and failure for the others. In this case, only the failed one performs the binary exponential backoff. For subsequent channel contention, the node which succeeded recently has a higher probability of winning the contention because of its lower backoff window. This prompts the successful nodes to access the channel in a more aggressive manner (because of lower backoff window) than the failed nodes. This skewed notion of backoff leads to an unfair access patterns on the channel. By not doing exponential backoffs after every collision that occurs and instead following collision resolutions, we try to give a fair chance to each node. All the nodes tend to have a similar congestion view of the network.

Fairness is an issue in our receiver-initiated approach as a result of some node spending majority of the time in its sending mode and little time as a receiver. We can circumvent this problem by forcing the nodes to move from sender to receiving mode or vice-versa; thereby controlling the time that a node gets to spend in each mode. By doing this, each node will get an equal chance of becoming a receiver or a sender and because of this, other nodes would get more fairer chance of becoming senders to these nodes. To enable this forceful transition from one from sending to receiving mode or vice-versa, we define a certain condition called as the *Condition for Mode Transition*. When a node satisfies this particular condition it makes a transition from one mode to the other.

3.5.2 Enhancing fairness by deliberate transition between modes

Every node is required to maintain a *Transmission Queue (TQ)* to enqueue the packets that need to be sent out, accumulated while the node is in the receiving mode. These packets will be dequeued in the First In First Out fashion. Every node starts as a receiver and remains in its receiving mode until one of the following occurs,

Conditions Receiver-to-Sender:

1. *Condition for Mode Transition* is satisfied.

2. After sensing the channel for Sense Time given by Eq. 3.1, the channel is found to be busy and the node's Transmission Queue is not empty. In this case, the node can go in its sending mode and try sending the packets from the Transmission Queue.

3. There are no more senders to this node. This is identified when a timeout occurs after sending an RTR, at the beginning of the CRI. A timeout can also occur when the node is in the middle of a CRI and the backoff stack is not empty, which usually occurs because there is no sender node in that particular AI of an RTR. To ensure that, a timeout occurred because there are no senders, the timeout considered for Conditions Receiver-to-Sender should occur just after initiating the CRI (the first RTR of a CRI) i.e when the backoff stack is empty.

In either of the above mentioned cases, the node then transits (deliberately) to the sending mode. In sending mode, the node tries to send all the packets in the Transmission Queue. Transition from sending to receiving mode now occurs when one of the following is satisfied,

Conditions Sender-to-Receiver:

1. *Condition for Mode Transition* is satisfied

2. The Transmission Queue is empty. Since, there is no packet to send, the node should make a transition to its own receiving channel and should initiate a new CRI by sending an RTR.

3. Timeout occurred because *Wait Time* for an RTR expired. Since, no RTR is heard during this entire duration of *Wait Time*, the node's intended receiver is not present on the receiving channel or is not in a receiver mode. We discussed in Sect. 3.2.2, that after waiting for a maximum of *Wait Time* for an RTR, the node makes a transition to its receiving mode. This was done to avoid indefinite waiting of RTRs. It would be very aggressive to just wait for an RTR from the intended receiver of the first packet in the Transmission Queue and make a transition, if RTR is not heard. There is a high possibility that this node has many packets in the Transmission Queue belonging to nodes who are present in their receiving channel.

   Instead, we follow a more cautious approach described as follows:

   (a) Mark the packet for which such a timeout occurred.

   (b) Scan the entire Transmission Queue, for all those packets that are not marked and try to send these data packets.

   (c) The node continues to be in sending mode until one of the mentioned conditions for sender to receiving mode transition (Conditions Sender-to-Receiver) are satisfied or the entire Transmission Queue is marked.

   (d) Before making the transition from sender to receiving mode, we unmark all the marked packets from the Transmission Queue. This is done so that all these packets are considered for data transfer the next time when we make transition from receiving to sending mode.

Condition for Mode Transition

While deciding upon the *Condition for Mode Transition*, we got to make sure that sufficient amount of time is given to a node in both the modes: sending or receiving. Such a decision could be based upon:

1. Fixed number of data packets sent successfully or fixed number of data packets received in Transmission Queue. Thus, with this as the *Condition for Mode Transition*, the node would make a transition from receiving to sending mode after it has a fixed number of data packets collected in Transmission Queue (Conditions Receiver-to-Sender). Transition from sender to receiving mode occurs when the node completes fixed number of data packet transmissions (Conditions Sender-to-Receiver).

2. Fixed amount of time spent in each mode. With this as the *Condition for Mode Transition*, every node is expected to spend equal amount of time in each mode.

Basing the decision of deliberate mode transition on some fixed value of data packets is a bit aggressive, as depending on the data rate, the time that a node spends in each mode will vary. With low data rate, *Condition for Mode Transition (fixed number of data packets in Transmission Queue)* for transition from receiver to sender mode will take longer to satisfy. As a result, the frequency of transitions from receiving mode to sending mode will be very less and the node will behave as a receiver most of the time. On the other hand, with high data rates, the fixed number of packets can be collected in the Transmission Queue in relatively short time. The frequency of transitions from receiving to sending will be more and the node will spend more time sending. The node would thus, behave as a poor listener. This is undesirable, as it might lead to some of the sender nodes to starve who have packets for this particular node.

Hence, we base these transitions contingent upon some fixed time called as the *Mode Time* that a node gets to spend in each mode and then transits when the *Mode time* is expired. With every node now requiring to spend *Mode Time* in each mode, we give fair chance to each node to be a sender as a well as a receiver and thus try to achieve fairness, even when the data rate changes.

The state transition diagram for a node in receiving and sending mode is shown in Fig.s 3.7 and 3.8 respectively.

Explanation: State Transition Diagram for receiving mode

1. The node in receiving mode sets its receive Mode Time, Recv Mode Time = *Mode Time* and begins by sensing the channel.

2. If the channel is found to be IDLE, the node sends an RTR. Depending upon, whether single RTS is heard or noise is heard or a timeout occurs, the node executes one of the paths as shown in Fig. 3.7.

   (a) **Heard RTS**:: If a single RTS is heard, the node completes a successful handshake of data transfer. After the data transfer, the node checks its backoff stack. The stack would be empty at this point since, only one RTS was heard and so, no collision resolutions takes place. The node checks to see, if the *Condition for Mode Time* is satisfied. i.e. whether the Recv Mode Time is expired. If the Mode Time is expired, the node makes a transition to the sending mode, if it has a packet to send or continues staying in the receive mode. However, if the Mode Time is not expired, the node again initiates a new CRI by sending out an RTR.

   (b) **Noise**:: If noise is heard, the node executes the collision resolutions as explained in Sect. 3.2.2. At the end of the CRI, when the backoff stack is found to be empty, the node makes a transition to sending mode if its Recv Mode Time is expired and it has a packet in Transmission Queue to send or continues in the receiving mode.

   (c) **Timeout**:: If a timeout occurs after sending RTR: this would mean no senders if the

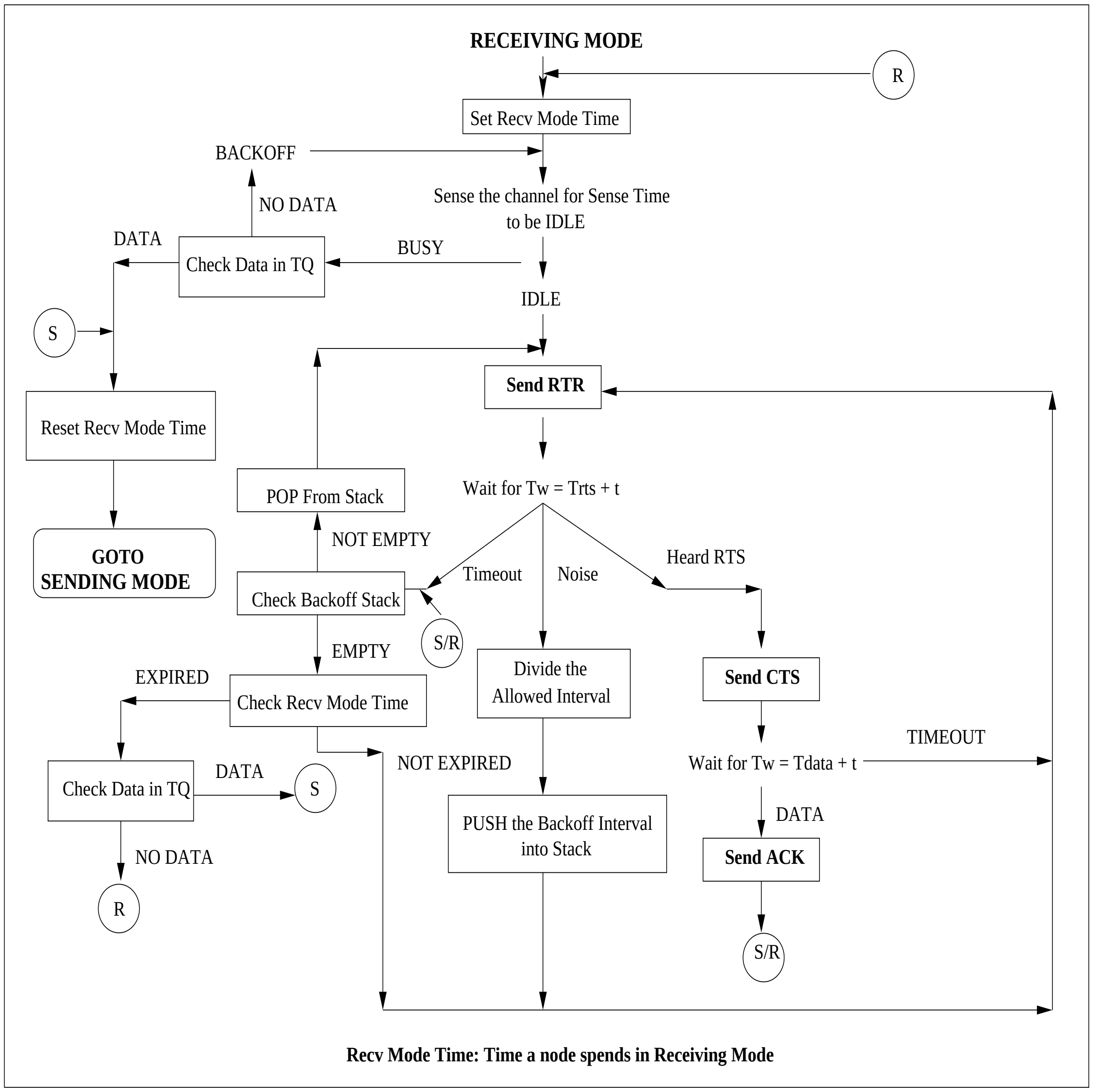


Figure 3.7: State Diagram for Receiving Mode

backoff stack is empty, in which case the node transits to the sender state if the *Condition for Mode Time* is satisfied. A timeout could also occur because of no sender nodes in the *AI* of the current RTR, in which case it pops out from the backoff stack and sends out the new RTR.

3. However, if the channel is found to be busy during the *Sense Time*, then the node essentially backsoff if the Transmission Queue is empty, or else transits to the sending mode. Though the Recv Mode Time for this node might still not be expired, but since the channel is found to be busy, we give this node the chance to become a sender. This is the reason we reset the Recv Mode Time to 0 even if it is not expired.

Explanation: State Transition Diagram for sending mode

1. The node in sending mode sets its send mode time, Send Mode Time = *Mode Time*, dequeues its first packet from the Transmission Queue and starts waiting for an RTR (Fig. 3.8).

2. **RTR**:: If an RTR is heard within *Wait Time*, Tw, the node completes its data transfer by participating in the CRI of the receiver node which takes place as explained in Sect. 3.2.2. After completing the data transfer, the node checks whether its Send Mode Time is expired or not.

    (a) **EXPIRED**:: If the Send Mode Time is expired, the condition for mode transition from sending to receiving mode is satisfied and so, the node resets its Send Mode Time to 0 and moves to its receiving channel.

    (b) **NOT EXPIRED**:: On the other hand, if the Mode Time is not expired, the node continues to remain in the sending mode, if it has data packets in its Transmission Queue. If the Transmission Queue is found to be empty, the node makes a transition to the receiving mode.

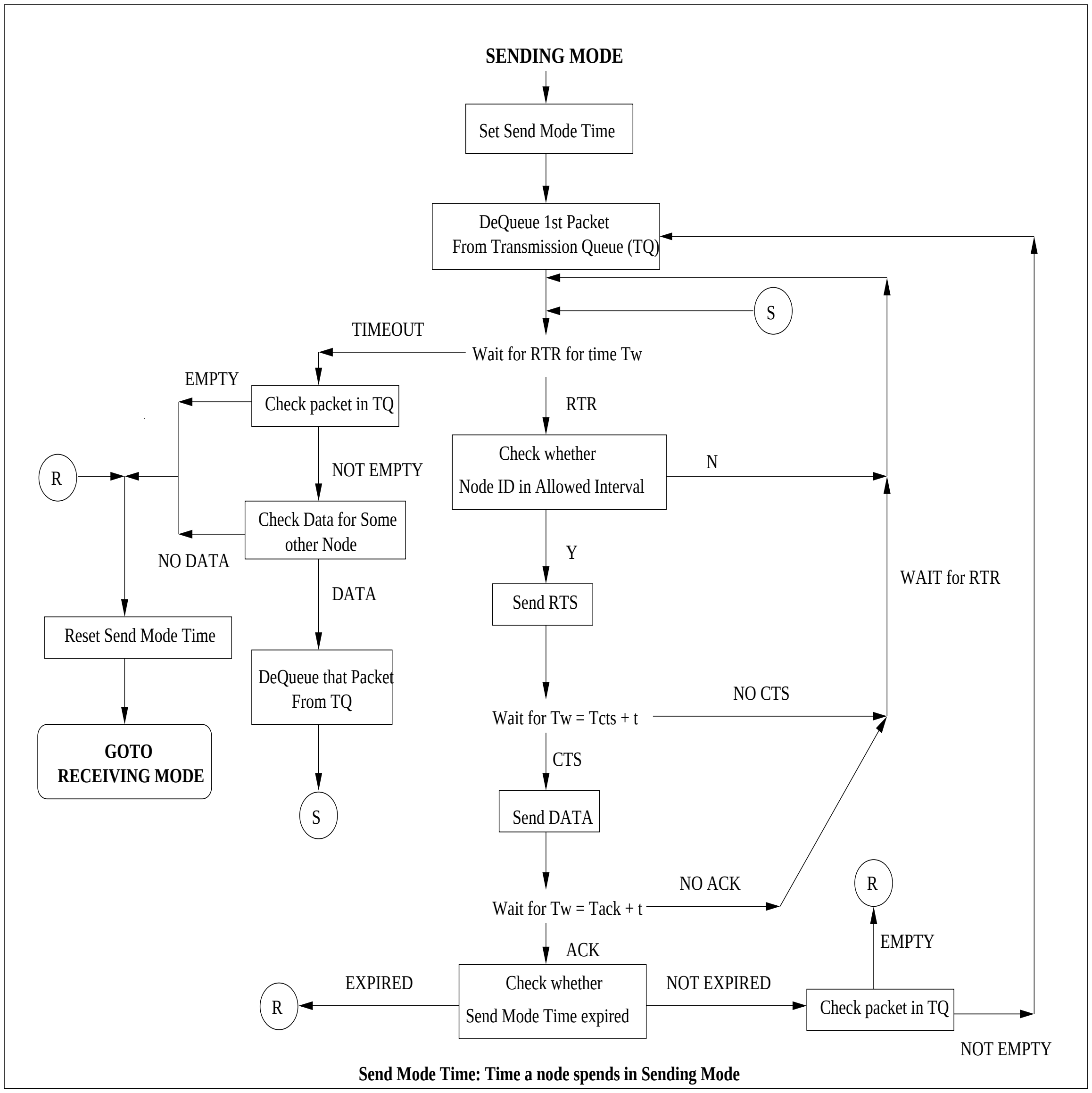


Figure 3.8: State Diagram for Sending Mode

3. **TIMEOUT**:: If no RTR is heard within the wait time, the node checks its Transmission Queue for a packet.

   (a) **NOT EMPTY**:: If the Transmission Queue is not empty, the node scans the Transmission Queue for a packet to some other node (other than the one for which timeout occurred). If such a packet exist, the node dequeues it out and starts waiting for an RTR from the intended receiver of this packet. If there is no such packet, the node essentially transits to the receiving mode.

   (b) **EMPTY**:: However, if the Transmission Queue is found to be empty and since, there is no response from the intended receiver in form of an RTR, the node makes a transition to the receiving mode.

   In all the above cases, when the node transits to the receiving mode, we reset its Send Mode Time to 0, even if it is not necessarily expired, so that the node now gets a chance to become a receiver.

3.5.3 Mode Transitions with overlap factor

If all the nodes were to spend same amount of time, *Mode time,* in each mode, possibilities of unfairness might still linger. A node might start as a receiver and would remain there till *Mode time* while a potential sender to this node might at the same time enter its receiving mode and be there for *Mode time*. So, after *Mode Time*, both the nodes would go into sending mode together and will never get a chance to listen to an RTR from each other. If this would occur repeatedly, the two nodes would never meet in the same receiving channel and hence, might result in chaos.

Following our approach of deliberate mode transition, each node is not just expected to spend some fixed time in each mode. The time that a node spends in each mode may also vary because of the roles that a node has to play in each mode. As can be seen from the conditions Receiver-to-Sender and Sender-to-Receiver, that a node transitions from one mode to the other due to several other reasons too. Thus, in sending mode, a node might spend certain time (less than *Mode time*) depending on the packets in the Transmission Queue or depending on the presence or absence of the node's intended receivers in their corresponding receiving channel (expiration of *Wait Time*). Similarly, in the receiving mode, a node might spend certain time (less than *Mode time*) depending on the usage of its receiving channel by its neighboring interfering nodes or depending on the number of senders it has. Thus, we can see that, the time a node spends in each mode is not fixed but it tends to vary and it will be different for all the nodes. In other words, the time that each node spends in any of the two modes is not static.

Each node follows a similar kind of approach. Thus, there is a high probability that, the sending node moves to its intended receiver's channel when the receiver is sending the RTR and gets a chance to do a data transfer. This helps in controlling the fairness problem. The time that each node spends in a mode being random automatically provides an overlap between a node's sending mode and its intended receiver's receiving mode. This overlap helps us to increase the number of data transfers between a receiver node and its senders. Idealistically, this overlap should be maximum in order to have more number of data transfers between any pair of nodes. This maximum overlap can occur, if the sender node enters its intended receivers receiving subchannel the same time when the receiver node initiates the CRI by sending the first RTR. So in this case, both these nodes might get a chance to spend the entire *Mode Time* sending and receiving packets. But, this maximum overlap is difficult to control due to the random nature of the time that every node spends in each mode.

We can still increase this overlap between the sending and receiving modes, if we can make a node spend more time in receiving mode than in its sending mode. This overlap will increase the chance of a sender node finding its intended receiver in its receiving channel most of the times. So, a sender node who entered its intended receivers receiving subchannel a little later than the receiver node night still get sufficient amount of time for data transfer.

To guarantee this increase in overlap, we introduce a term called *Overlap factor* (β), $0 < \beta \leq 1$. The *Mode Time* for senders is set to $\beta\times$ Recv Mode Time, where, Recv Mode Time is the time a node will spend in receiving mode. $\beta = 1$ implies same *Mode Time's* for senders and receivers and any value less than that implies less time sending than receiving. With $\beta < 1$, a sender node would have more chance of finding its intended receiver on its receiving channel; thus, contributing to minimize the fairness issue. It doesn't make sense, to have $\beta > 1$, as it is less useful to have a node in sending mode, if its intended receiver's *Mode Time* as a receiver is expired. This sender node would just waste its *Wait Time* (given by Eq. 3.2) waiting for an RTR only to find that its intended receiver is not present in the receiving channel (which left because it was suppose to spend less than time than its sender node).

By following a deliberate mode transitions approach, we are trying to control the fairness problem present because of the strict nature of the receiver-initiated protocol. However, from throughput perspective, we are introducing certain inefficiencies in the network. We are placing a restriction on every node to switch between modes. Thus any node, even with a heavily loaded *Transmission Queue* also will be required to switch to its receiver mode upon expiration of *Mode Time* in the sending mode; in some worst cases, only to find that there are no senders to it. In that case switching from sender to receiver mode every time after expiration of *Mode Time* places an overhead and might cause the performance to degrade. We would expect less packets being sent per flow in comparison with our earlier protocol without deliberate transitions. However, the simulation results presented later illustrate that, the receiver-initiated enhanced MAC protocol with deliberate mode transitions performs better in terms of fairness and almost the same or better in terms of throughput as compared to without deliberate transitions.

CHAPTER FOUR

PERFORMANCE ANALYSIS

This chapter presents the performance analysis of our receiver-initiated MAC protocol with enhancements of collision resolutions, subchannel assignment and deliberate mode transitions. We have conducted a series of experiments in order to analyze the performance of the protocol. Also, in order to study the effect of each enhancement on the receiver-initiated protocol, we have presented results at each stage of enhancement and compared the performance after each stage. Since, the protocol was designed as an improvement over the standard IEEE 802.11 MAC, we have compared the results obtained with the results from IEEE 802.11 MAC.

4.1 Simulation Overview

In order to test the performance of the proposed MAC protocol, a number of simulations were performed. Using the well known network simulator Ns2 and its topology generator, simulated wireless network environment was generated and experiments were conducted to study the behavior of our MAC protocol.

4.1.1 Ns2 Network Simulator

Ns2 is a discrete event simulator targeted at networking research. It was developed by the Information Sciences Institute at the University of Southern California with wireless extensions from the CMU Monarch project [2, 3]. Ns2 is an object oriented simulator, written in C++, with an OTcl interpreter as a frontend. Ns2 provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. The overall simulation is described by a Tcl Simulator. It provides a set of interfaces for configuring a simulation and for choosing the type of event scheduler used to drive the simulation. A simulation script generally begins by creating an instance of this class and calling various methods to create nodes, topologies, and configure other aspects of the simulation.

Ns2 implements an extended network stack that makes simulations of wireless networks possible. A packet sent down the stack flows through the link layer, the MAC layer and the physical layer. The packet then makes its way up the stack through the MAC and the Link layer. The MAC layer contains a set of functionalities such as carrier sense, collision detection, collision avoidance etc. An object called as the *Mac object* simulates the Medium Access Control protocols necessary in shared environment such as wireless local area networks. Ns2 further extends the Mac object to implement the IEEE 802.11 MAC protocol. The wireless extensions made to Ns2 provides a complete implementation of the IEEE 802.11 standard.

Several MAC schemes proposed in the past have used NS2 as the simulation tool for performance analysis and hence, the results obtained by using NS2 simulator can be more relied upon. Also, since Ns2 implements IEEE 802.11 MAC, implementing our protocol using Ns2 simulator, gives us results for direct comparison with 802.11 MAC.

4.1.2 Simulation Model

In our experiments, we have considered the wireless medium to be noiseless and error-free. Thus, the packet losses are only due to collisions at the destination node caused by interference from other neighboring nodes. Any packet, that is heard when the node was receiving some other packet, is detected as a collision and hence, both such packets are dropped.

Number of Nodes	**Physical boundary**	**Highest degree**
25	670 x 670	6
50	1000 x 1000	9
100	1000 x 1000	17

Table 4.1: Random network models

Network Setup

For the simulations we used different stationary network models: that is the nodes do not have any mobility. Tables. 4.1 and 4.2 shows the different random network and mesh network models used, and their configuration.

Number of Nodes	**Grid arrangement** (*x-axis* x *y-axis*)	**Physical boundary**	**Highest degree**
25	5 x 5	1000 x 1000	4
50	5 x 10	1000 x 2250	4
100	10 x 10	2250 x 2250	4

Table 4.2: Mesh network models

Network Traffic

For all network models, traffic was generated according to independent Poisson process at each node with identical mean arrival rates which was varied to change the offered load. The offered load was varied from as low as 10 packets/sec on each flow to as high as 100 packets/sec/flow. However, all the links were chosen to be of constant data rate i.e. Constant Bit Rate (CBR) flows on all the links with fixed size of data packet as 512 bytes. CBR is a type of traffic that requires a continuous, specific amount of bandwidth. It has the following characteristics: unidirectional, known packet size, known packet interval, a certain delivery rate that it has to conform to, causes stress in the network at high loads when the sender buffer could overflow. With TCP flows, the reliability is assured and this metric is not useful for performance comparison. For a MAC protocol to be stable and efficient, it would be more sensible to test it with CBR flows, where maintaining a certain *delivery rate* is required and where most of the MAC schemes would perform poorly, if there are large number of data packets drop. A MAC protocol will be said to be stable, if it can maintain the desired data rate and its throughput does not choke even at high data rates.

For the simulations, the sender-receiver node-pairs, within communication range of each other, were chosen at random. Each node was assumed to behave as a sender as well as a receiver. The results obtained are based on the average of 10 different simulations conducted for each set of experiments. The 10 different simulations were conducted by changing the seed fed to the Ns2. These averages should reflect the network more closely than individual data points. All simulations were run for a duration of simulated 10 secs.

Protocol configuration Parameters

Table 4.3 summarizes the parameters used for the simulations.

Simulation Parameter	**Size/Value)**
RTS	20 bytes
CTS	14 bytes
DATA	512 bytes
ACK	14 bytes
RTR	16 bytes
Propagation delay (τ)	5μs
Transmission Range	250m
Data rate of control and data packets	1Mb
SIFS	8μs
DIFS	16μs

Table 4.3: Protocol configuration parameters

4.2 Simulation results

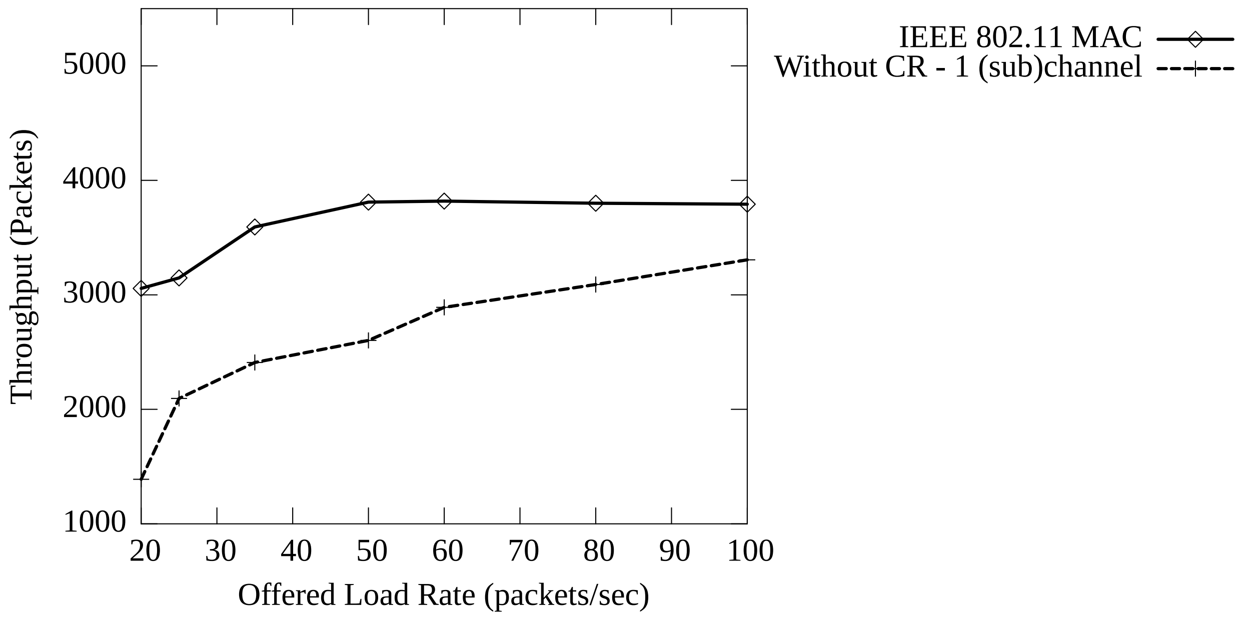
We conducted a series of experiments, to observe the performance of the our protocol by incorporating the enhancements we proposed, one by one. This makes it easy to compare and to understand the individual and cummulative effects of these changes and what we achieved after every enhancement.

4.2.1 Receiver-Initiated approach - without enhancements

We conducted a set of experiments in order to compare the throughput performance of the IEEE 802.11 protocol with receiver-initiated protocol without any enhancements. This was intended to just compare a normal sender-initiated approach (802.11 MAC) with our approach without any enhancements. The topology used for this experiment was a 50 node random topology with configuration as described in Table 4.1. We established 30 CBR flows between randomly chosen node-pairs.

Fig. 4.1 shows the plot of throughput versus the offered load for IEEE 802.11 MAC (the plot represented as “IEEE 802.11 MAC”) and our protocol without collision resolution on a single channel (the plot represented as “Without CR - 1(sub)channel”). The offered load is measured as the number of packets per second per flow. We described in Sect. 3.1, that the receiver-initiated approach without any collision resolutions follows a random backoff strategy for collision avoidance as in IEEE 802.11 MAC. The sender nodes do a random backoff on absence of a CTS from their intended receiver, similar to IEEE 802.11 MAC scheme.

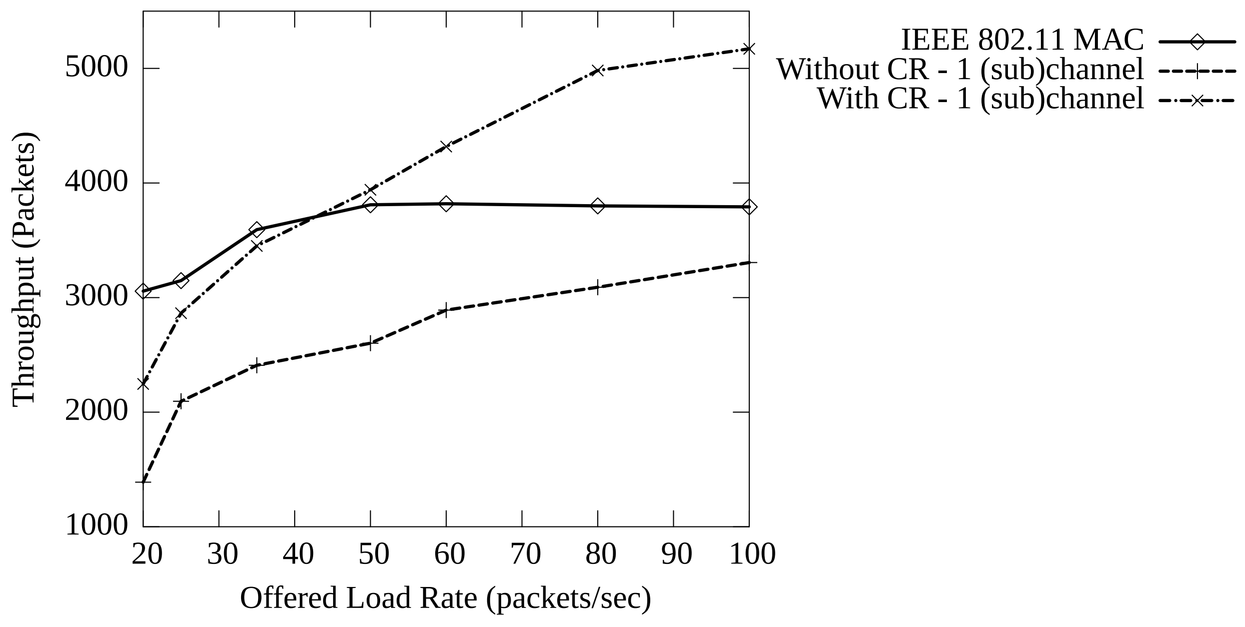
From Fig. 4.1, we can observe that the throughput obtained with the receiver-initiated approach without collision resolutions is less as compared to the IEEE 802.11 MAC. By doing backoff, first of all we waste significant time because of backoff’s. Secondly, the receiver-initiated approach places a strict requirement on every node to wait for an RTR from their intended receiver. Thus, these nodes who just did a random backoff, will have to wait again for a next RTR from their



(50-node random network,30-CBR flows)

Figure 4.1: Throughput without collision resolutions

intended receiver to try again. While these sender nodes backoff, the receiver node might transit from receiving to sending mode and thus, we cannot guarantee that the intended receiver will be still in its receiver mode after the competing sender nodes backoff time expires. Also while the sender nodes wait for an RTR, both these nodes backoff time might expire before one of them hears an RTR from the receiver and this might further result in collisions of RTSs. So, we see that collision avoidance, the main purpose of random backoff is not getting accomplished. Random backoff's do take place in IEEE 802.11 MAC, but since it is a sender-initiated approach, one of the competing nodes win first and without any further delay sends out an RTS to its receiver, if the channel is sensed idle. Thus, because of these inefficient backoff's in our approach without any enhancements, we can see throughput degradation when we move from IEEE 802.11 MAC to receiver-initiated approach.



(50-node random network,30-CBR flows)

Figure 4.2: Throughput with and without collision resolutions

4.2.2 Collision Resolutions

Next, we conducted experiments to understand the effect of first enhancement i.e. collision resolutions on receiver-initiated protocol. IEEE 802.11 MAC becomes inefficient at heavy loads, since with increasing traffic there is a higher wastage of bandwidth from collisions and random backoffs. When NAV becomes unreliable in cases of collision, the sender nodes can interfere with an ongoing transmission to its intended receiver: the hidden node problem effect. So, the performance of IEEE 802.11 MAC suffers, particularly at high loads where the channel contention is large. With a receiver-initiated approach, we try to avoid the interference caused by a sender to its intended receiver's ongoing transmission. We proposed the first enhancement, collision resolutions that will resolve the collisions among RTSs from multiple senders and hence, will grant access to the senders in order to send packets.

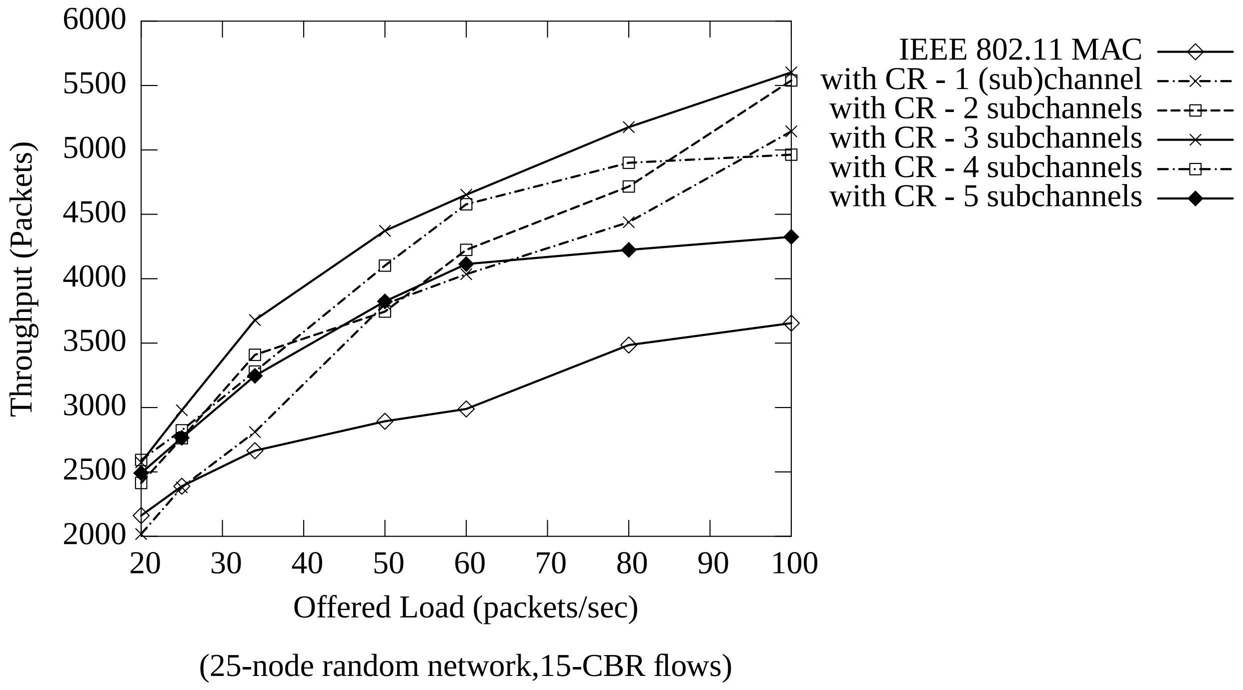


Figure 4.3: Throughput with collision resolutions and varying number of subchannels for 25 node network

Fig. 4.2 shows the plot of throughput versus the offered load for IEEE 802.11 MAC, receiver-initiated protocol without collision resolution on a single channel (the plot represented as “Without CR - 1(sub)channel”) and also with collision resolution using single channel network (the plot represented as “With CR - 1(sub)channel”) for a 50-node random network with 30 flows.

We see that there is a considerable improvement by doing collision resolutions for receiver-initiated approach. By introducing the collision resolution technique in receiver-initiated MAC approach, we gain firstly because there is no time wasted in doing random backoff’s. Secondly, since the propagation delays and the duration of RTSs and CTSs are less than the duration of data packet, resolving the collisions among RTSs were accomplished relatively quickly and hence, we can see better throughput. The receiver-initiated protocol with collision resolution behaves as standard 802.11 MAC with one subchannel but due to the overhead of collision resolutions, at low data rates with 1 (sub)channel, it performs worse than standard 802.11 MAC. However, as the offered

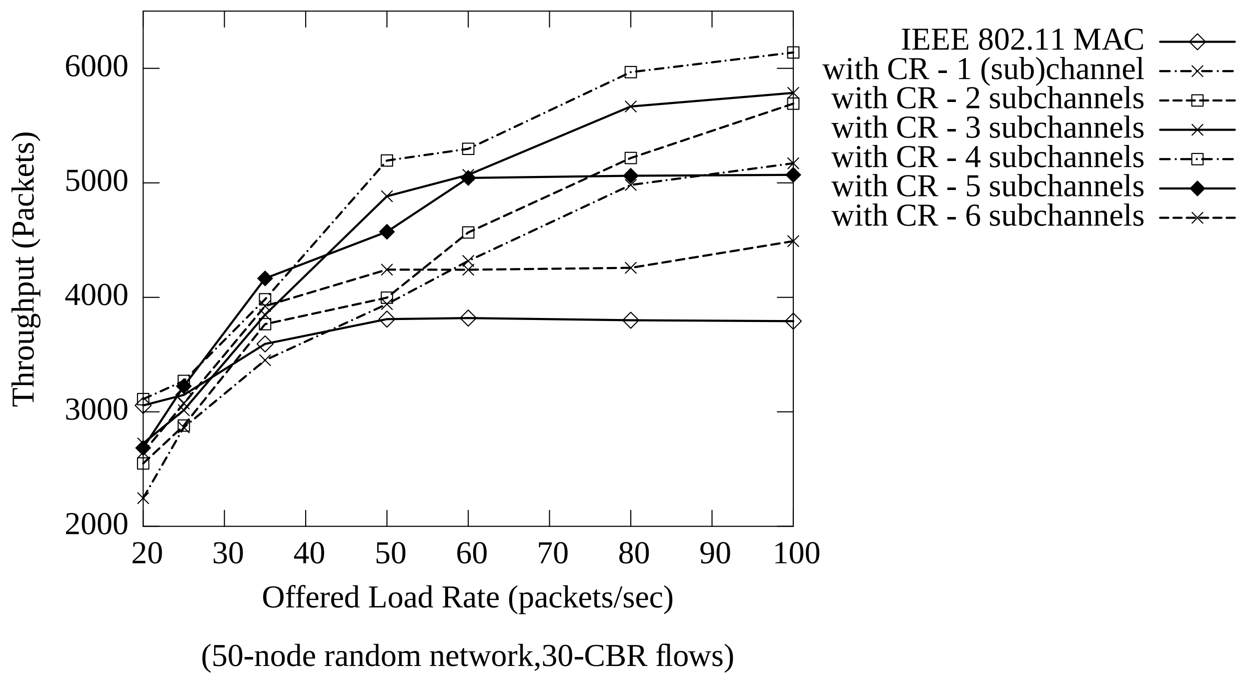


Figure 4.4: Throughput with collision resolutions and varying number of subchannels for 50 node network

load increases, the overhead becomes less significant and hence, we see an increase in throughput benefiting from fewer collisions with data packets.

4.2.3 Collision Resolutions and Subchannel Assignment

The next step is to compare the performance of our protocol with combined effect of two enhancements: collision resolutions and subchannel assignment.

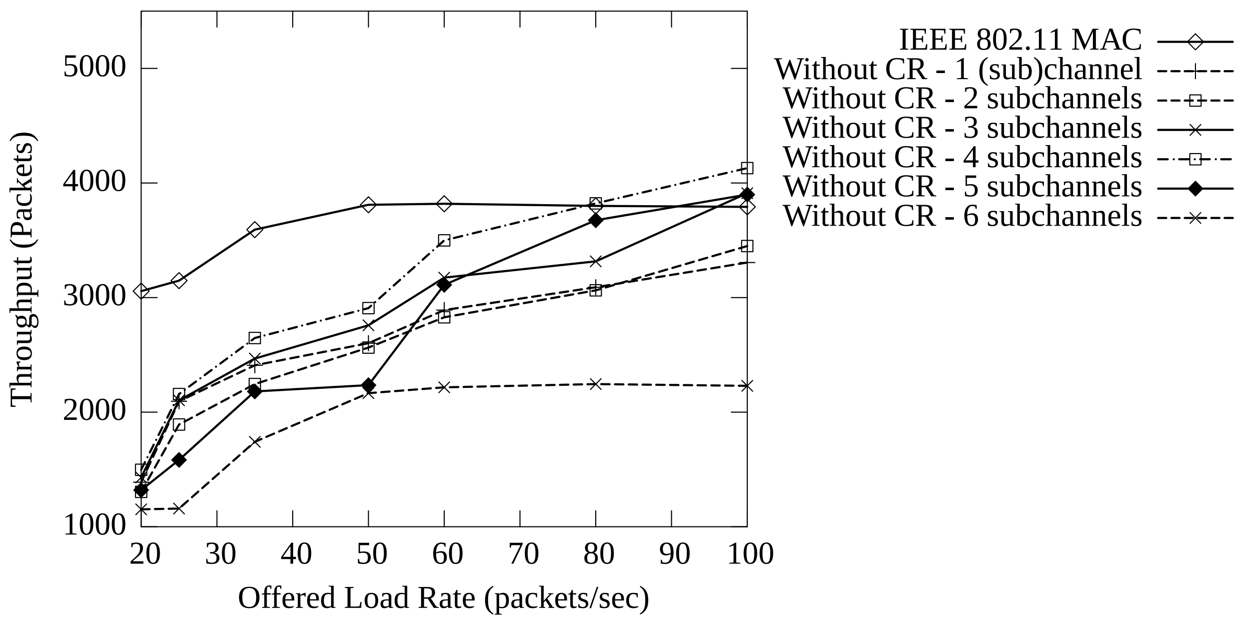
Throughput

The first result we consider is the throughput performance of the receiver-initiated protocol with both enhancements:collision resolutions and subchannel assignment. The number of subchannels used were varied and the corresponding throughput was calculated. The number of subchannels represent the total number of subchannels available for assignment. By varying the number of subchannels, we tried to understand the behavior of the protocol when using different number of subchannels. Fig.s 4.3 and 4.4 shows the throughput plot for 25 node and 50 node random network respectively.

From both the Fig.s 4.3 and 4.4, we can observe that, at low traffic loads, the effect of co-channel interference is not significant. With nodes having a limited number of packets in their transmission queue, the competition among the nodes for the channel is not high. Hence, having more subchannels does not help much at low loads. However, as we increase the offered load, more contention can be seen among neighboring nodes and so, we gain by making use of parallel transmissions as we introduce multiple subchannels. With increase in the offered load the throughput rises further as we minimize the co-channel interference.

With increase in the number of subchannels, the data rate the network can sustain decreases. Due to the bandwidth division among subchannels, the transmission times of the packet increases with the number of subchannels. Significant amount of time is spend for each data transfer. Thus, a node is found to be busy most of the times thereby decreasing the number of successful transmissions that would have taken place. For 25 node network (4.3), the throughput increases from 1 (sub)channel to 3 subchannels and then we see a fall in throughput. Also, from Fig. 4.4 for 50 node network, we can notice that the throughput rises as we go from 1 (sub)channel to 4 subchannels and then, the throughput starts decreasing with further increase in the number of subchannels. With 3 subchannels for 25-node and 4 subchannels for 50-node network, many concurrent transmissions were feasible and exploited mitigating the effect of exposed node problem. We can see an improvement of about 50% with 3 subchannels for 25 node network and with 4 subchannels for 50 node network when compared to standard 802.11 MAC.

To observe the effect subchannel assignment would have on the working of our receiver-initiated approach without collision resolutions, we conducted experiments using multiple subchannels and no collision resolutions. The topology used was a 50 node random network with 30



(50-node random network,30-CBR flows)

Figure 4.5: Throughput without collision resolutions - with subchannel assignment

CBR flows.

We can observe from the Fig. 4.5, that as we increase the number of subchannels, the throughput increases. At high offered load, the performance is slightly better than the IEEE 802.11 MAC. This is because, by throwing in more channels, we gain by allowing parallel transmissions, not possible in 802.11 MAC whose effect is significant only at high loads. However, as we discussed in Sect. 3.4.1, there exists a restriction on throughput enhancement as we increase the number of subchannels used. This is because, as we increase the number of subchannels, the transmission time of packets increase, without an increase in parallel transmissions and so we don't gain much. The throughput rises till a certain limit (subchannels = 4) and then we see a drop as we further increase the number of subchannels. The results are plotted only for subchannels varying from 1 to 6, as after subchannels = 4, we see fall in throughput and this continues even we increase the number of subchannels beyond 6.

Comparing Fig.s 4.2 and 4.5, we can also observe that if we don't perform collision resolutions in receiver-initiated approach and only have multiple subchannels for concurrent transmissions (Fig. 4.5), even at high loads we do not gain much as we do by resolving collisions even with a single channel network (Fig. 4.2). Thus, mere introducing multiple subchannels for concurrent transmissions does not help improving the throughput performance, if collisions are not resolved.

Packet Transmission Delay

The total time required for a node's packet to finally make up to the destination should be very less. This time includes the transmission time for the packet which is fixed and also the waiting time that this packet experiences in the transmission queue of the sender. Ideally speaking, this waiting time for the packet should be 0. An efficient MAC protocol should attempt to minimize the total time the node spends before it reaches its final destination. This time difference between two events: data arrival at source node and data received at destination node is defined as the *Packet transmission delay*. Hidden node problem which increases the number of retransmissions and exposed node problem which defers transmissions unnecessarily are the main contributors to the delay. IEEE 802.11 MAC in the presence of both these problems induces a significant amount of delay. If the number of retransmissions can be reduced, and if we can allow several packets to be sent at the same time, the delay incurred on the packets can be reduced.

We conducted experiments to study the effect of collision resolutions and subchannel assignment on the transmission delay experienced by the packets. The packet delay is calculated as the time difference between the time the packet arrived at the node's transmission queue and the time it was finally received at the station. Further, we average out the delay calculated for each flow by the total number of packets that were sent successfully on the corresponding flow. This gives the per flow average delay experienced by each packet on the corresponding flow. Inoder to get the

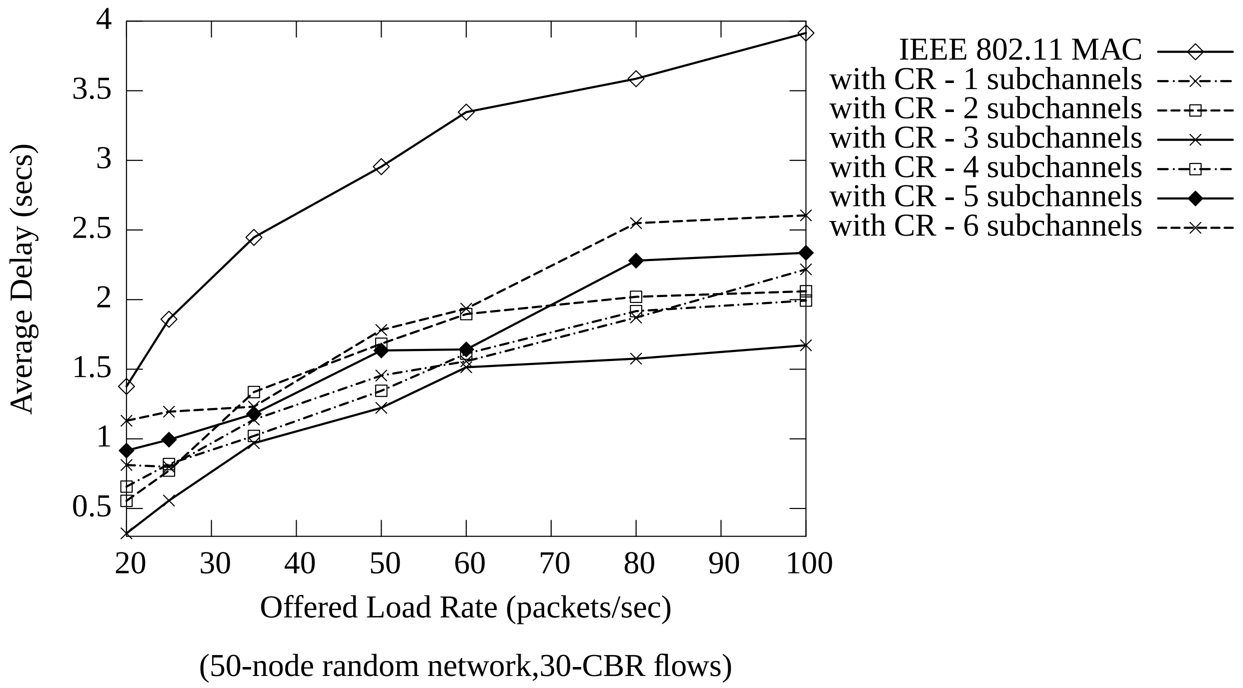


Figure 4.6: Delay comparison of standard 802.11 with proposed protocol

overall average delay experienced by a packet in the network, we further averaged out this per flow delay by the total number of successful data flows established in the network.

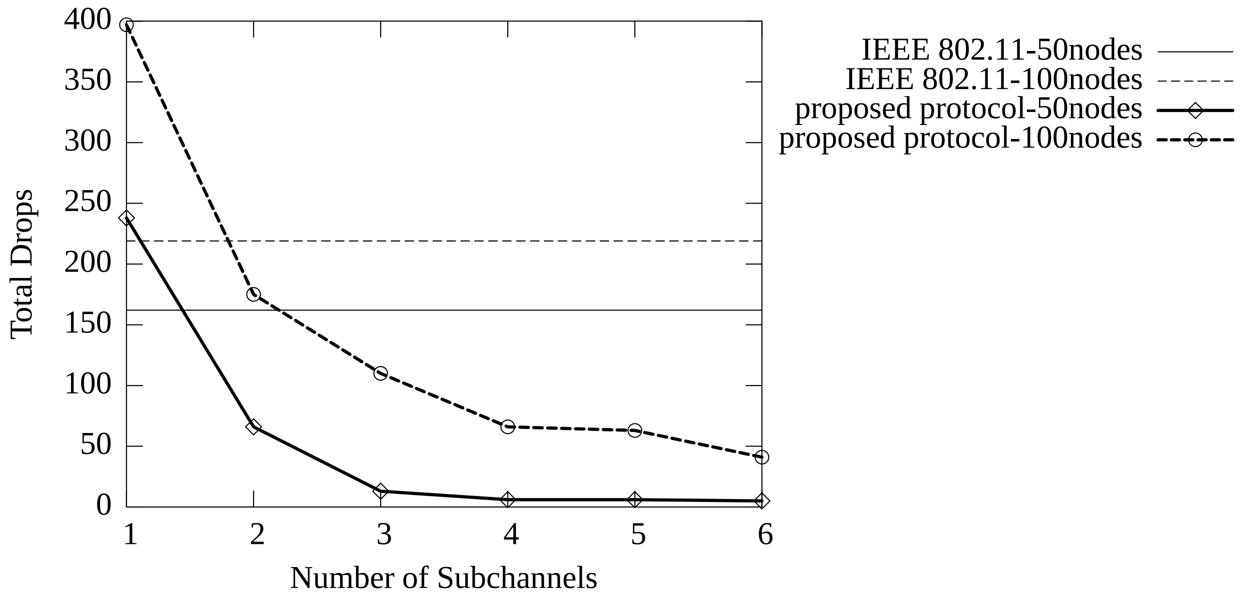
In Fig. 4.6 we plot the average packet transmission delay for a 50 node random network with 30 flows. The number of subchannels was varied from 1 to 6. So essentially, the plot shown as “with CR - 1 subchannels” represents the packet delay for the protocol only with collision resolutions and no subchannel assignment. At low offered load, the contention is less and so the effect of both hidden terminal problem and exposed terminal problem is not significant. Few number of data packets collision occur and hence, there are very few packet retransmissions. Thus, we see that at low traffic, the packet delay is slightly less than or almost similar to that of the IEEE 802.11. With increase in the network traffic, the channel contention increases and our protocol shows much better delay performance as compared to IEEE 802.11 because there are fewer packet collisions and hence, fewer retransmissions.

However, it should be noted that due to increase in the number subchannels when the bandwidth of each subchannel reduces, it takes more time for a packet to travel from sender to the receiver on a low bandwidth (sub)channel. Thus, the packet transmission times increases and hence, the delay could be more than the delay experienced on a single channel with the entire bandwidth as we increase the number of subchannels. We see that the delay decreases as we move from 1 (sub)channel to 3 subchannels and then, the delay increases as we further increase the number of subchannels. With reduction in the bandwidth allocated to each subchannel, the transmission time for the packets increased and hence, the time required for a packet to finally make up at the destination node was more. We see the least delay is experienced by the packets with 3 subchannels and the highest delay with 6 subchannels.

Dropping of data packets

The data packets drop because of collisions of some control packets with the data packet. This happens when nodes unaware of some ongoing transmission finds the channel to be idle and hence, interfere by sending out their request for floor acquisition. So due to increased channel contention the number of data packets drop increases. Comparing the data packets drop of our protocol to those of IEEE 802.11 MAC, will give us an idea of how much interference did we actually control and what is the effect of channel subdivision on the drops of data packets. These data packets drop were calculated by counting all the data packets that a particular receiver node has to drop, because it heard some interference from the neighboring nodes while it was receiving the data packet. The topologies chosen for the experiment was a 50-node and 100-node random network. For both the 50-node and 100-node network we chose the same physical boundary of the nodes as 1000 x 1000 and established 30 flows of CBR traffic. The maximum degree of the 50 and 100 node random network was 9 and 17 respectively.

Fig. 4.7 shows the data packets drop as we vary the number of subchannels. We can observe



(50-node, 100-node random network, Data rate per flow 100 pkts/sec)
(30-CBR flows)

Figure 4.7: Data packets drop comparison of standard 802.11 with proposed protocol

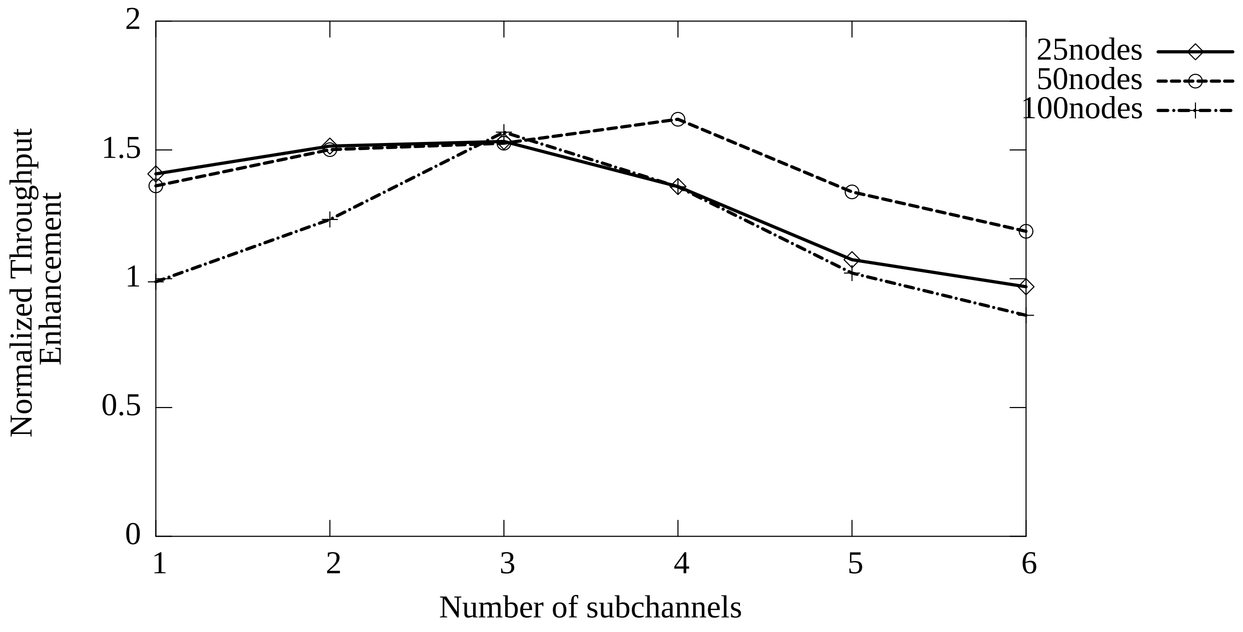
that we have more number of data packets being dropped with 1 (sub)channel as compared to IEEE 802.11 MAC for both 50 and 100 node networks. The receiver-initiated approach introduces overhead in the network because of an additional control packet that is present before every successful data transfer that takes place and also sequences of RTR in case of collisions. Every node irrespective of any senders to it, sends an RTR and hence, with a dense network, we expect the network to be loaded with unproductive RTRs. Thus, a node with dense 1-hop neighborhood, might miss out important information from an RTS or CTS packet about an ongoing transmission because of their collision with other RTRs from its neighborhood: the NAV becomes stale. In a highly dense network with 1 (sub)channel it becomes difficult to identify the current transmission state of the network. So, with no knowledge of an ongoing transmission multiple nodes find the channel to be free (even if it is not) and hence, interfere resulting in increase in the number of drops in data packets.

As we increase the number of subchannels, the neighboring nodes send RTR and receive data packets in their distinct subchannels and hence, the interference from the neighborhood is reduced. And hence, we see that there is a significant decrease in the number of data packets drop as we move from 1 (sub)channel to 6 subchannels with almost no packets being dropped with 6 subchannels. From this we can infer, that by doing collision resolutions and subchannel assignment, the interference is reduced to a level where we can see negligible packets drop with more subchannels. And this fact remains consistent for both the 50-node and 100-node network.

We chose, the same physical boundary of 1000 x 1000 for both the 100-node network and 50-node network, because, with double the number of nodes now in the same area, there will be more contention among neighboring nodes. The fact that the maximum degree of the nodes raised from 9 to 17 for a 100 node network shows the increased contention interference that every node will now face. From Fig. 4.7, we can see more data packets drop for 100 node network than with 50 nodes even though the number of flows is same which is expected when the contention among nodes increase. We can also notice that, even with such large contention, the data packets drop reduces drastically with the receiver-initiated approach following the two enhancements as compared to the IEEE 802.11 MAC. Thus, our receiver-initiated protocol with collision resolutions and subchannel assignment has strengths to control the interference as compared to 802.11 MAC even in cases when the contention is large.

Number of subchannels for maximum throughput

The next experiment was aimed to identify whether there exists a particular number or a range of numbers into which the channel if subdivided will give maximum attainable throughput. And also, if such a range of subchannels remains consistent for networks of varying sizes and nature. For this experiment, we used 3 sets of random network models; the first set consisting of 25nodes in a grid of 670 x 670 with 15 flows, the second and third set consisting of 50 nodes and 100 nodes

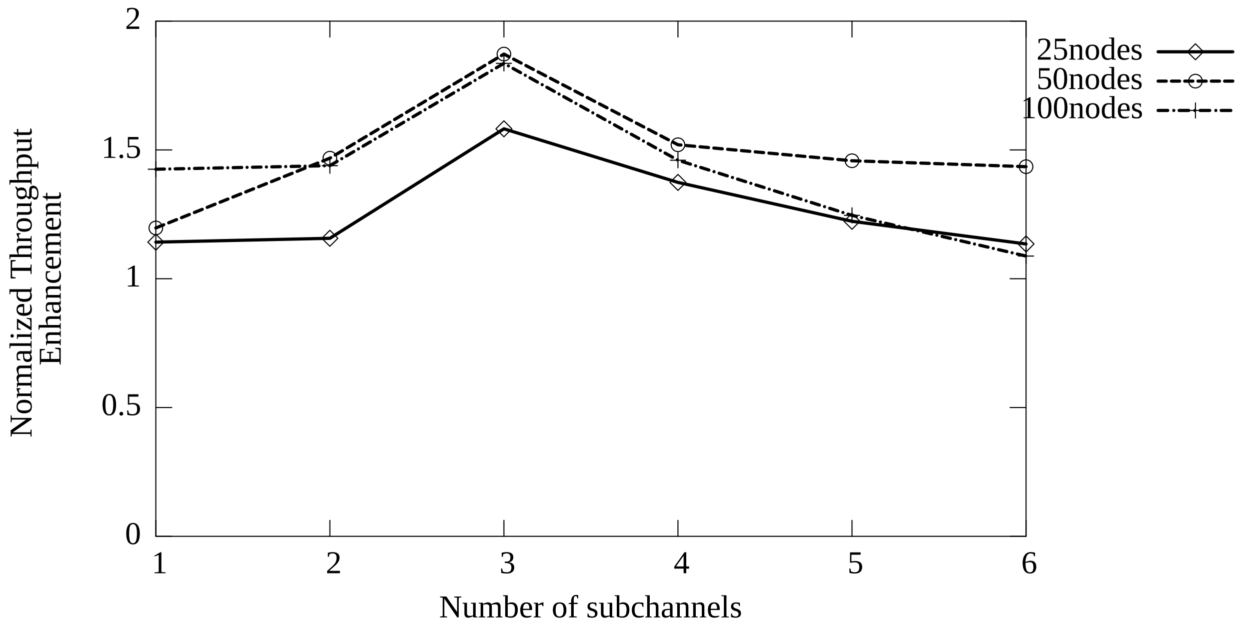


(25,50,100 node random network, Data rate per flow 100 pkts/sec)

Figure 4.8: Throughput for random topologies of varying sizes

respectively in a grid of 1000 x 1000 with 30 flows. We also used 3 sets of mesh network models, 25 nodes (5 x 5) in a grid of 1000 x 1000 with 15 flows, 50 nodes (5 x 10) in a grid of 1000 x 2250 with 30 flows and 100 nodes (10 x 10) in a grid of 2250 x 2250 with 60 flows. The main intention was here to find out, that as we increase the size of the network from 25 nodes to 100 nodes, will the number of subchannels required to achieve maximum throughput increase or does it remain the same. And also, to find out if this has any effect on the nature of the network when we move from a random network to a mesh network.

Fig. 4.8 and Fig. 4.9 show the plot for throughput enhancement for random and mesh topologies as we vary the number of subchannels. The plot shows the *normalized throughput enhancement* which was calculated by normalizing the actual throughput with the throughput of the IEEE 802.11 MAC for the corresponding network and offered load. Thus, the value of the *normalized throughput enhancement* greater than 1 implies how much we improve over standard IEEE 802.11 MAC



(25,50,100 node mesh network, Data rate per flow 100 pkts/sec)

Figure 4.9: Throughput for mesh topologies of varying sizes

and a value less than 1 implies the degradation in throughput.

From Fig. 4.8, we can observe that the maximum throughput is obtained with 3 subchannels for random network consisting of 25 and 100 nodes and with 4 subchannel for a 50 node random network. Thus, we can define the range of subchannels required to achieve maximum throughput enhancement to be [3, 4]. Fig. 4.9 shows that the maximum throughput for mesh network: 25, 50 and 100 nodes, was obtained with 3 subchannels which also falls in the range of [3, 4]. Thus, varying the size of the network does not have any effect on the number of the subchannels required for maximum throughput. Also, we can notice that even the nature of the topology (mesh or random) has no effect on the number of subchannels required for maximum throughput.

4.2.4 Deliberate Mode Transitions

Next, we need to evaluate the performance by adding the third enhancement of Mode Transitions to the protocol and observe the behavior in terms of throughput, fairness and drops of data packets. The main aim behind the third enhancement was to address the fairness issues raised by the proposed receiver-initiated protocol.

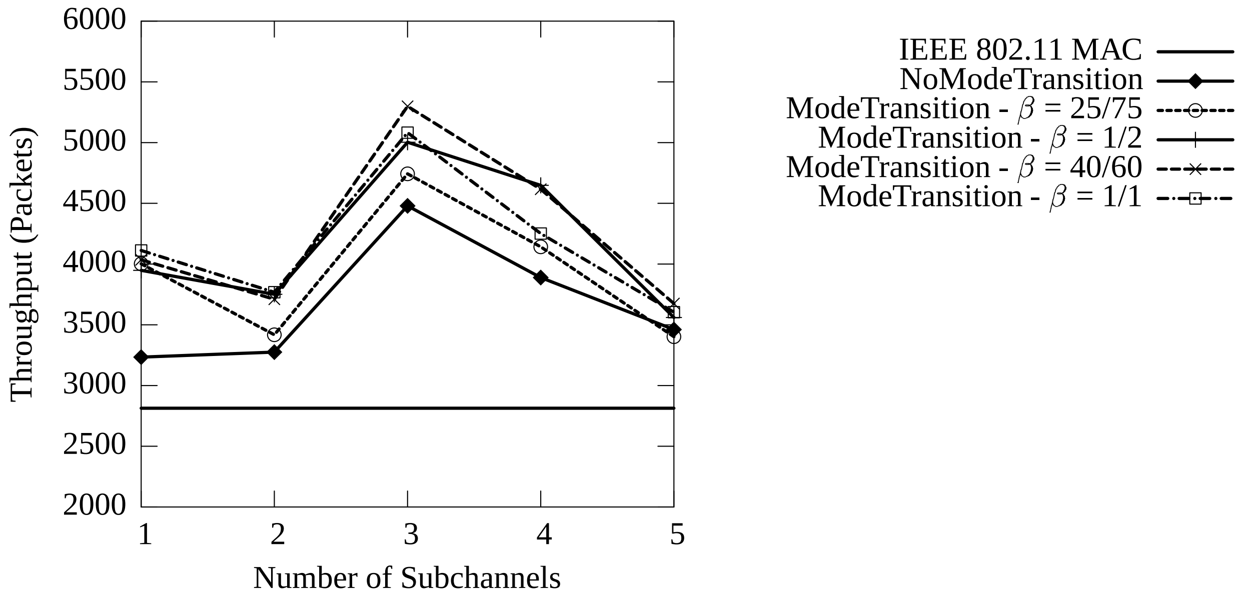
Throughput

First, we compare the throughput performance of the protocol with and without Mode transitions. We aim to identify the effect of deliberately making the nodes to switch modes, on throughput. We conducted several experiments by varying the *overlap factor*, β, that governs the overlap between the sending and receiving mode, and *Mode Time*, the time a node spends in each mode. The aim of these experiments was to investigate the possible combination of β and *Mode Time* that gave the maximum throughput and also to find out whether Mode Transitions introduce inefficiencies with respect to throughput as compared to when there are no Mode Transitions.

Effect of Varying overlap factor, β on throughput

We vary the *overlap factor (β)* to analyze the effect of different *overlap factors* on throughput of the protocol with Mode Transitions. For the experiment, we kept the *Mode Time* fixed = 1.0 secs. β was chosen to be a ratio of the overlap between sending and receiving mode, where $0 < \beta \leq 1$. Thus, a value of β = x/y implies that, if a node spends a maximum of T secs in receiving mode, in sending node the node will spend a maximum of (x/y * T) secs. Simulation results for a mesh topology of 25 nodes and a random topology of 50 nodes with 100 packets/sec sent on each flow are shown in Fig.s 4.10 and 4.11 respectively.

The straight horizontal line indicates the throughput for standard 802.11 MAC, and forms a



(25-node mesh network,Data rate per flow 100 pkts/sec)
(15-CBR flows)

Figure 4.10: Throughput with Mode transitions by varying *overlap factor*, β for 25 node mesh network

baseline for comparison. β = 25/75 implies that a particular node would spend most of the time receiving and less time sending. This will provide a high percentage of overlap between sender and receiver and thus, will increase the chances of a sender finding its intended receiver in receiving mode. But with β = 25/75, the sender node just spends $1/3^{rd}$ of the time the receiver node spends as a receiver. With β = 25/75, we achieved the lowest throughput among all the other bias factors as the sender spends too little time sending, even if the receiver is willing. β = 1/1 reduces the chances of overlap with every node spending same time in each mode, but the sender node gets equal time to send as the receiver node gets to receive. It performs better than with β = 25/75. With β = 40/60 and 1/2 we see better results with the maximum throughput being achieved with β = 40/60. These overlap factors provided sufficient overlap and also sufficient time to a sender node to be able to send data packets to its intended receiver.

We also note that with all the bias factors, our proposal with Mode transitions performs better

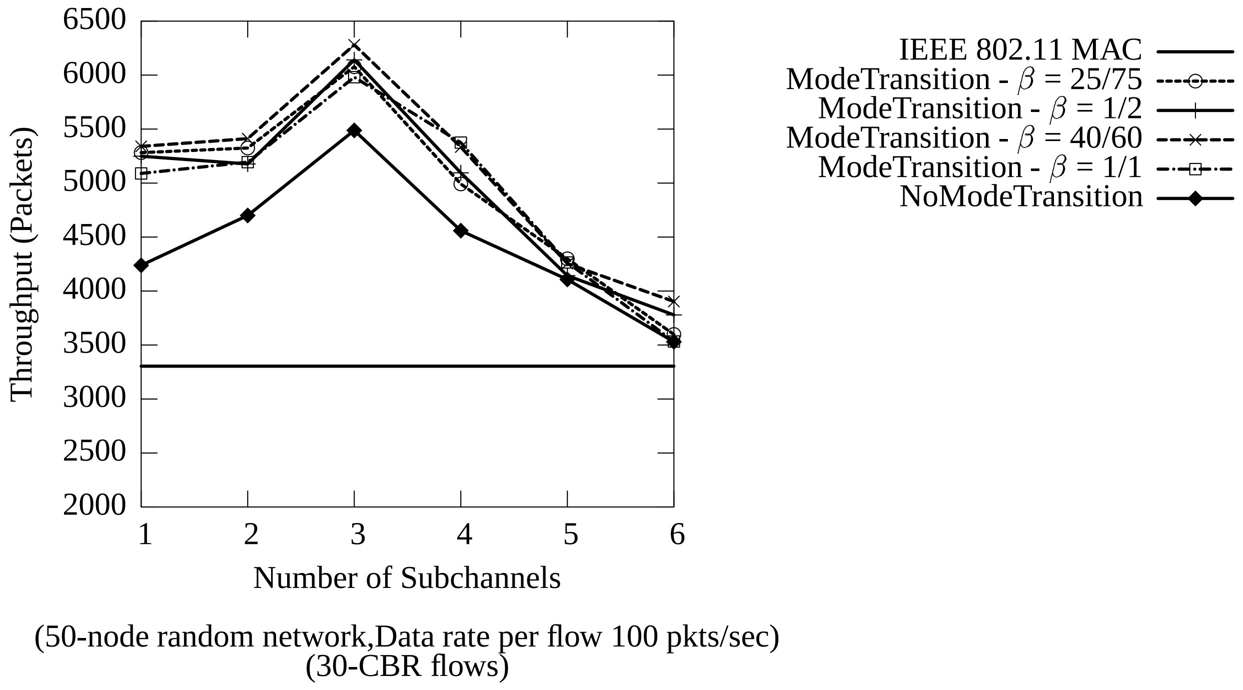


Figure 4.11: Throughput with Mode transitions by varying *overlap factor*, β for 50 node random network

than with no Mode transitions (the plot shown as “NoModeTransition”) at all subchannels. Thus, we can infer that, switching between modes even at high load of 100 packets per second when the transmission queues are expected to be full does not degrade the throughput performance. This could be accounted as, with no mode transitions some nodes spend significant amount of time waiting for their intended receiver, who itself might be in sending mode with a loaded transmission queue. And this would repeat every time the starved node completes a CRI as a receiver and moves to the intended receivers channel only to find that it is not present. Thus, large amount of time was wasted in waiting. Now, with Mode Transitions, considerably less time would get wasted as we increase the chances of a sender node finding its receiver in receiving mode and this waiting time even reduces further because of the overlapping we try to achieve between the sender and receiver. Hence, we see that the throughput does not degrade and we gain by giving each node a fair chance to be able to send and receive packets.

Effect of Varying *Mode Time* on throughput

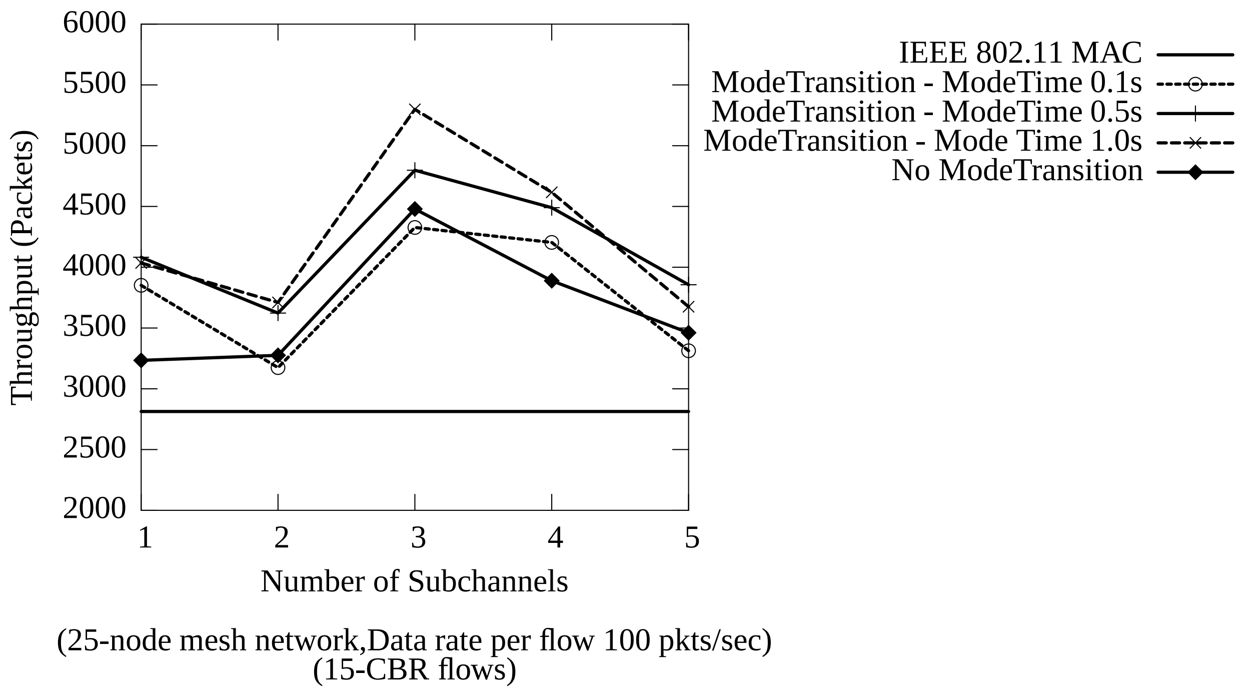


Figure 4.12: Throughput with Mode transitions by varying Mode Time for 25 node mesh network

Next, we choose β = 40/60 and vary the *Mode Time's*. *Mode Time* is the time the node would spend in receiving mode and for the sender this *Mode Time* is multiplied by the factor, 40/60. With β = 40/60, a sender node spends 2/3rd of the time it would spend as a receiver. We identified that β = 40/60 gives sufficient overlap between sender and receiver. But, at the same time we also want the sender node to spend adequate time in the receivers channel and do as many data transfers as possible in that time. Apart from β, the other factor that controls the amount of time the node would spend as a sender or receiver is the *Mode Time*. This experiment was aimed at identifying that particular value of *Mode Time* which along with β = 40/60 would give adequate results. Results for 25-node mesh and 50-node random network are as shown in Fig.s 4.12 and

4.13.

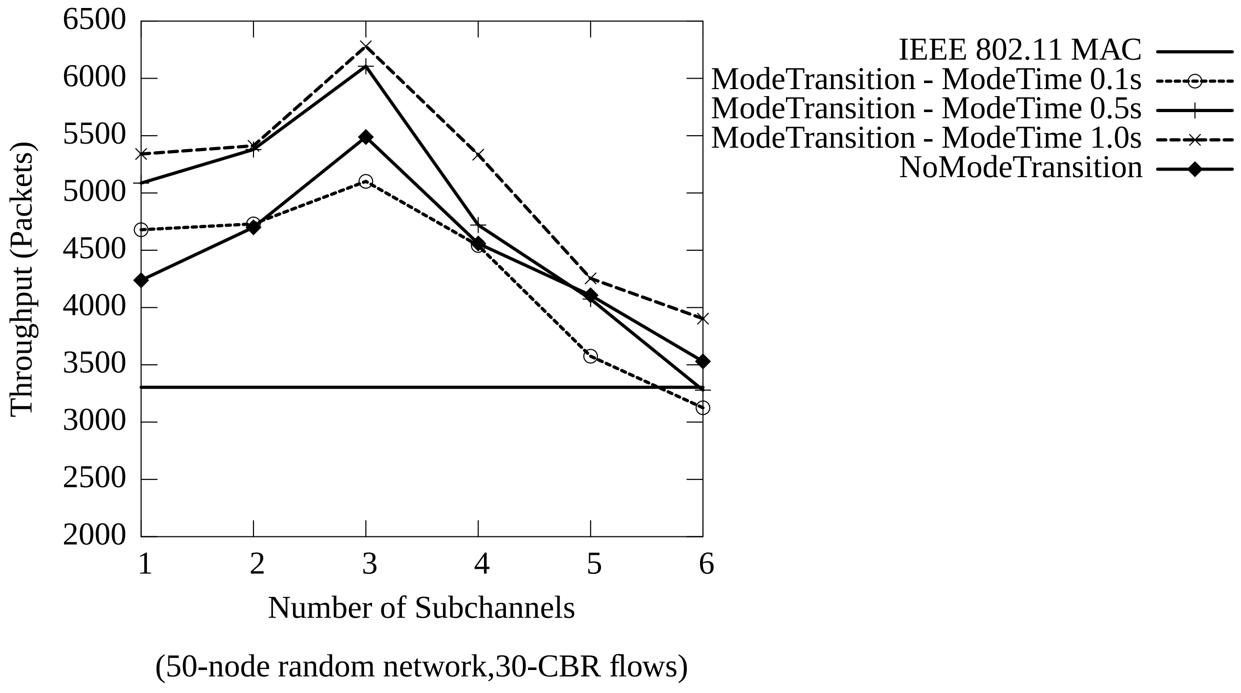
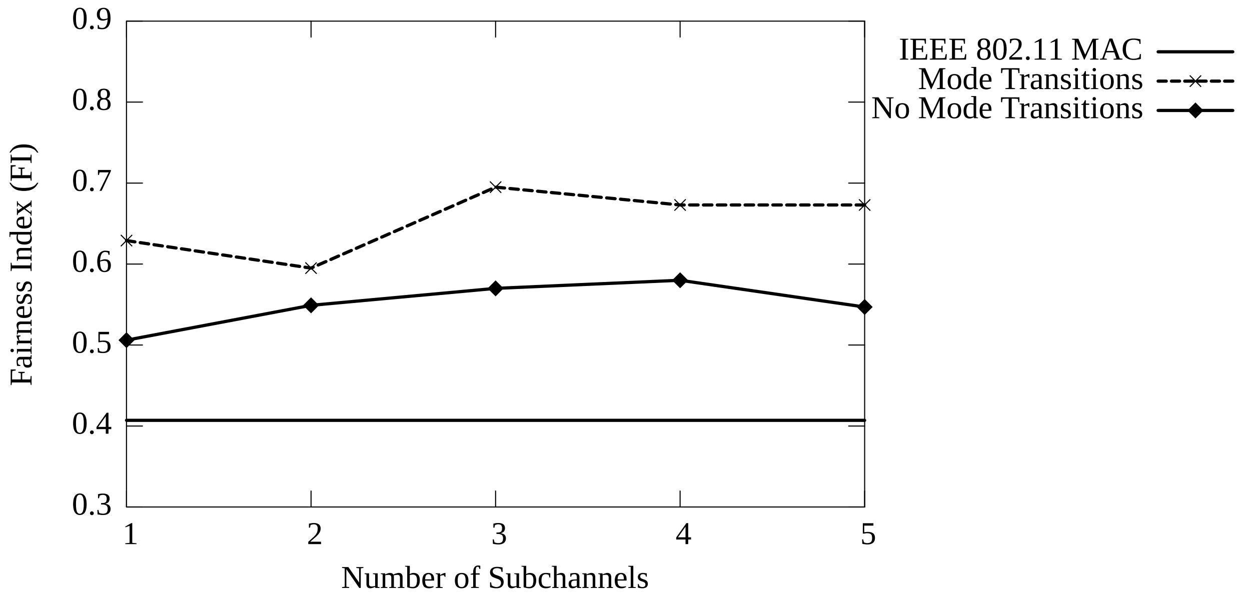


Figure 4.13: Throughput with Mode transitions by varying Mode Time for 50 node random network

Mode Time 0.1s is too fine-grained for successful data transmissions. Also, with *Mode Time* = 0.1s, every node would make several transitions from sender to receiver and vice-versa in a considerably short period of time. Every time the node makes a transition from sender to receiver, the node has to sense the channel for *Sense Time* before sending out its RTR. So with these frequent transitions, significant amount gets wasted and hence, we see that the throughput is very less. As we move from *Mode Time* 0.1s to 0.5s, we see a rise in throughput. By spending some more time in each mode and keeping the same *overlap factor*, we reduce the number of transitions and the time getting wasted in sensing the channel because of more frequent transitions. This is the reason that, as we further increase the *Mode Time* to 1.0s, the highest throughput is achieved. These results remain consistent as we increase the number of subchannels.



(25-node random network,Data rate per flow 100 pkts/sec)
(15-CBR flows)

Figure 4.14: Fairness comparison with and without Mode transitions for 25 node random network

Fairness

The intention behind doing Mode Transitions is to give fair chance to each node to become a sender and a receiver and thereby, eliminate the fairness problem. This Mode Transitions will force a poor listener node (node who spends most of the time in sending mode and very less time in receiving mode), to become a better listener (or a receiver) when the *Condition for Mode Transition* satisfies. And this will help reducing the fairness problem upto a certain degree. We still cannot guarantee complete avoidance of such unfairness, as the transitions between the receiver and its corresponding senders are not synchronized. So, its difficult to assure that a sender node, everytime it makes a transition from receiving to sending mode, will find its intended receiver on its receiving channel.

To evaluate the performance of the protocol in terms of fairness, we used a well known concept of Fairness Index. The fairness is measured by calculating the normalized deviation of the end-to-end throughput called as *Fairness Index* [24]. The *Fairness Index (FI)* is given as follows,

$$FI = \frac{\left(\sum_{i=1}^{n} f_i\right)^2}{n * \sum_{i=1}^{n} f_i^2} \tag{4.1}$$

where, n = number of the total flows on the network, f_i gives the throughput for the i^{th} flow. With the help of this formula, we can calculate how much similarity does the throughput on each flow has. This is inverse relation to standard deviation which calculates the deviation of throughput on each flow from the mean. Thus, higher the FI, the more fair the protocol is. Ideally a fair protocol would have a *Fairness Index* of 1. Fig.s 4.14, 4.15, 4.16 and 4.17, present the fairness aspects of the protocol with and without Mode transitions for 25 and 50 node random and mesh networks respectively. Experiments were conducted for topologies of different sizes and nature for more reliable inferences.

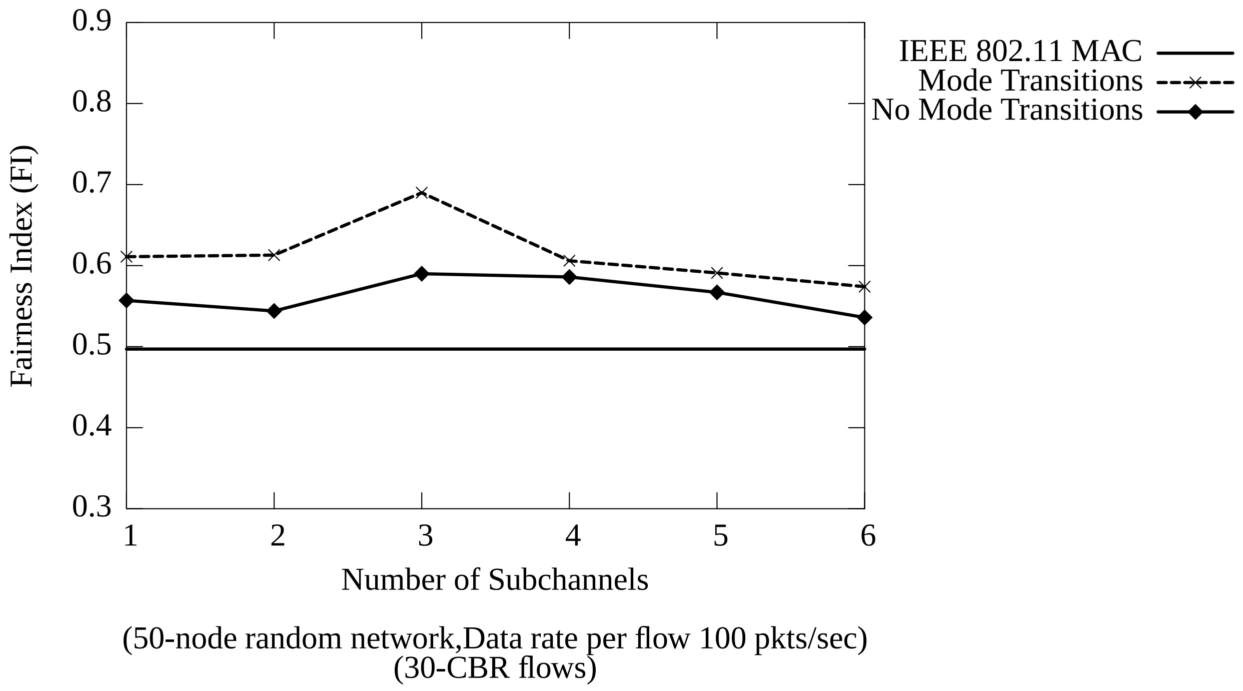


Figure 4.15: Fairness comparison with and without Mode transitions for 50 node random network

We can observe that standard 802.11 MAC exhibits a high degree of unfairness. The commonly used Binary Exponential Backoff (BEB) scheme, in 802.11 despite its robustness against repetitive collisions, raises fairness issues. Nodes suffer because the node that succeeded in the last transmission period will gain access to the shared channel again with much higher probability. Without deliberate Mode Transitions, fairness suffers: the roles that each node assumes is very strict making the protocol without Mode Transitions unfair.

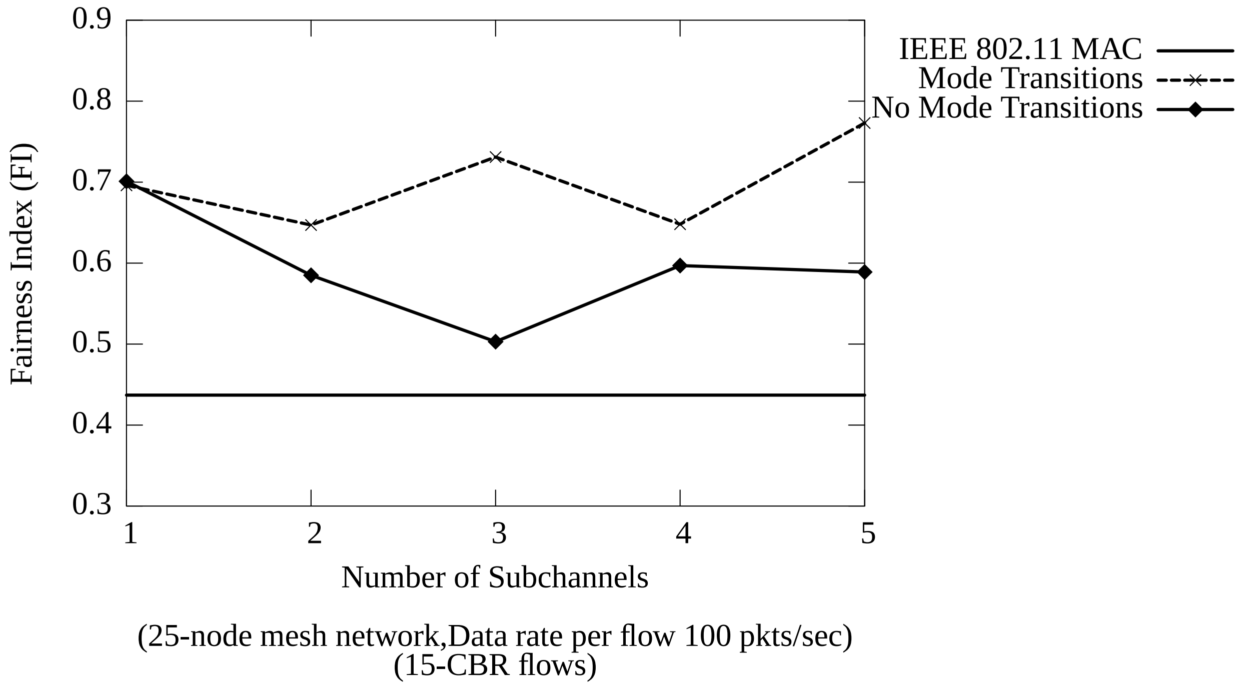


Figure 4.16: Fairness comparison with and without Mode transitions for 25 node mesh network

When we introduce deliberate Mode Transitions, first of all we ensure that a node does assume both the roles: a sender as well as a receiver. So, many of cases wherein a sender node will never find its intended receiver on the receiving channel now do not exist. We further try to increase the chance that, a sender node hears an RTR from its intended receiver most of the times with the help of the *overlap factor* β, and also by controlling the *Mode Time*. Thus, we expect to see almost the same number of packets sent successfully on each flow. From Fig.s 4.14, 4.15, 4.16 and 4.17, we can observe that the fairness curve for the protocol with Mode Transitions (the plot shown as Mode Transitions) is very high as compared to both the IEEE 802.11 MAC and the protocol without Mode Transitions. Also, the point worth noting is that, this behavior remains consistent with all the subchannels and varying the size or the nature of the network has no effect on the fairness aspect of the protocol with Mode Transitions.

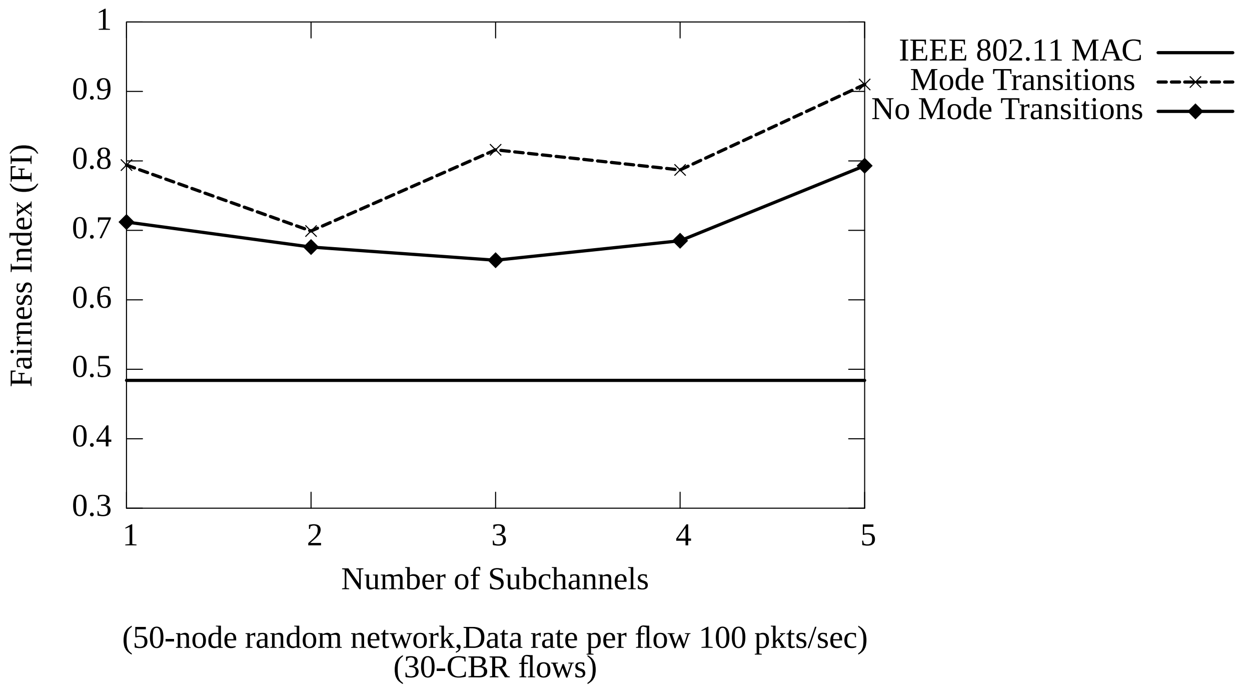


Figure 4.17: Fairness comparison with and without Mode transitions for 50 node mesh network

Dropping of data packets

We know that data packets drop because of collisions with other control packets and this happens when the NAV of the nodes becomes stale due to collisions and hence, fails to reflect the state of the network. We conducted a series of experiments as explained in Sect. 4.2.3, to identify the total number of data packets drop with the collision resolutions scheme and how this varies as we vary the number of subchannels. From the results we observed that, the number of data

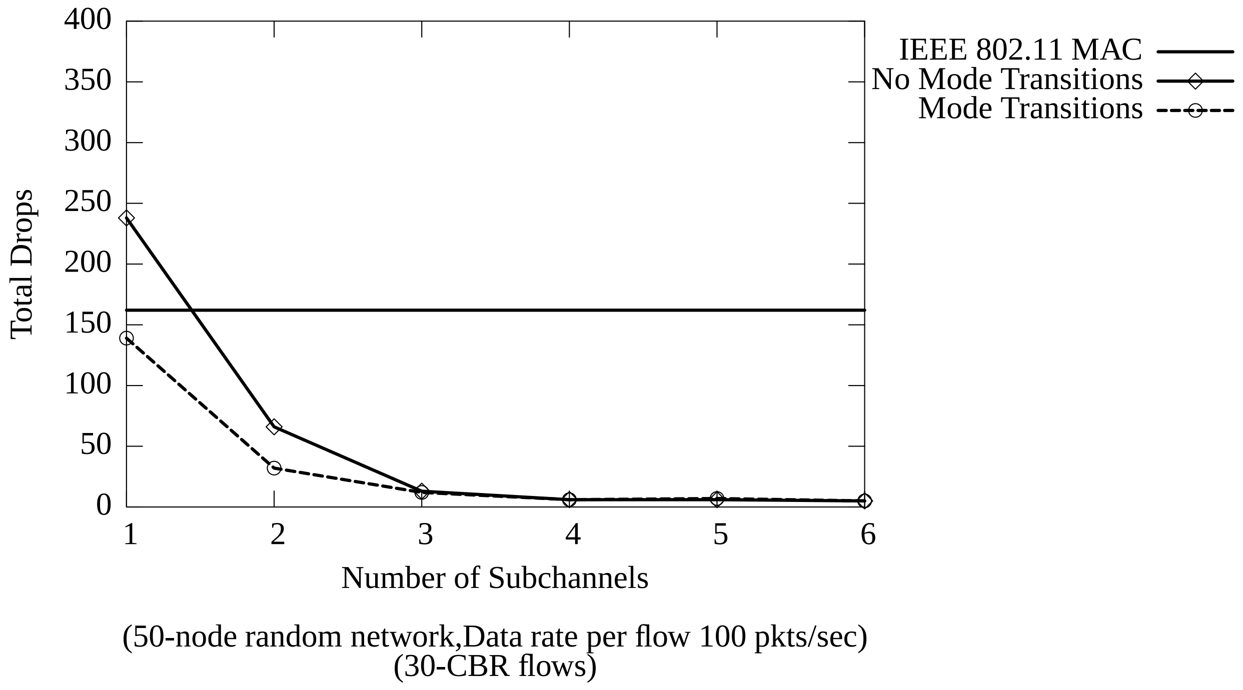


Figure 4.18: Data packets drop with and without Mode Transitions for 50-node random network

packets drop is comparatively more than 802.11 MAC with 1 (sub)channel and then, this number decreases significantly as we move from 1 (sub)channel to 6 subchannels. We also figured out that, this happens because of large number of RTR packets on the single channel which makes the NAV updation difficult and the presence of co-channel interference worsens the situation. As we increase the number of subchannels, the more we control the co-channel interference, the less number of data packets will drop.

We conducted experiments to calculate the number of data packets drop when Mode Transitions are introduced and to compare this number with, when no Mode Transitions take place. Random and mesh topologies with 50 nodes was chosen for the experiment with a traffic load of 100 packets/sec/flow and 30 CBR flows were established. For the Mode Transitions, the parameters used were: β = 40/60 and *Mode Time* = 1.0s. The plot for the total drops of data packets with and without Mode Transitions are as shown in Fig.s 4.18 and 4.19. The straight line shows the

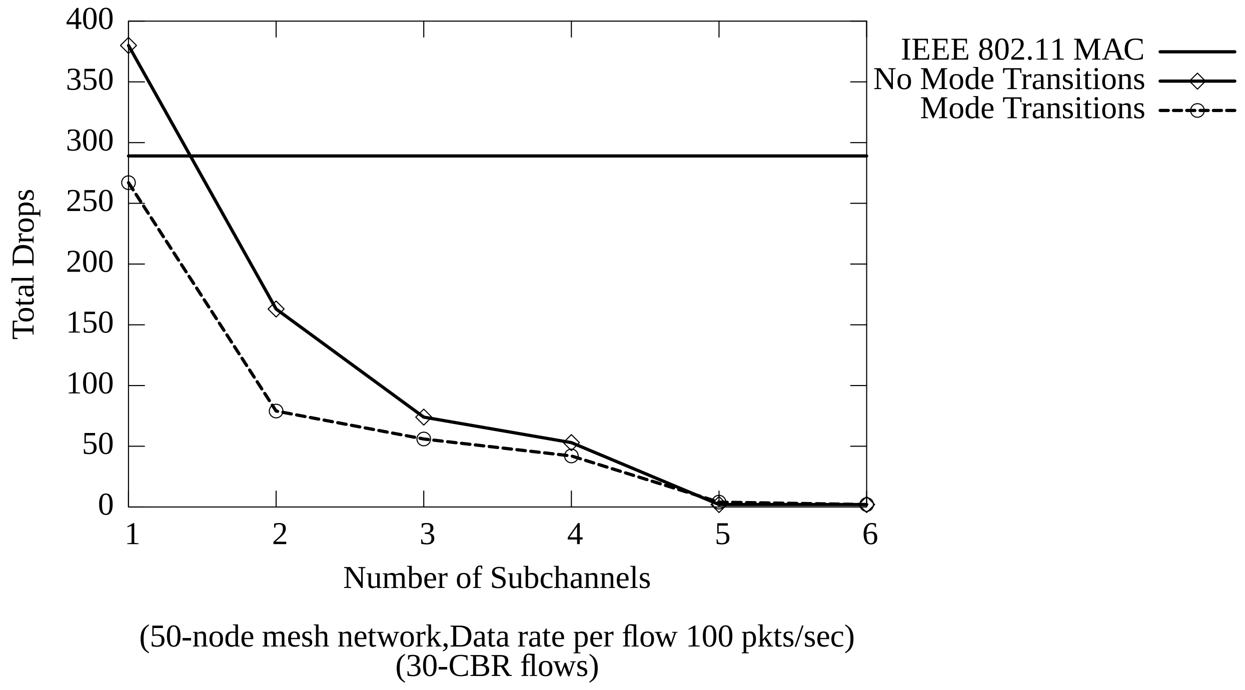


Figure 4.19: Data packets drop with and without Mode Transitions for 50-node mesh network

total number of data packets drop for the IEEE 802.11 MAC.

The first and foremost point to be noted from the Fig.s 4.18 and 4.19 is that, the total drops with Mode Transitions is less than the total drops for the IEEE 802.11 MAC. Next, we can observe that the number of data packets being dropped with Mode Transitions is also less as compared to the data packets drop without Mode Transitions even with 1 (sub)channel. With the notion of Mode Transitions, every node now switches its modes between sending and receiving. In this process of Mode Transitions, as we control the behavior of each node by forcing them into mode transitions, some of the RTRs collisions reduce. Because of this reduction in RTRs collision, the updation of NAV becomes slightly easier compared to when there were no Mode Transitions. Hence, we can see less number of data packets being dropped even with a single channel, where the number of RTR collisions is the maximum. This effect could be seen consistently both from Fig.s 4.18 and 4.19, when the number of subchannels is less.

When, we increase the number of subchannels, the chances that all the neighboring nodes will now send RTRs and receive data packets in their own assigned receiving channel increases. By doing this, we get a handle on the number on the data packets being dropped because of collision with some control packet. We also studied that ,the co-channel interference is not much of an issue when we have more number of subchannels. With more subchannels, the interference is minimized and so are the data packets drop. So, even without Mode Transitions, the number of RTR collisions are reduced in presence of more number of subchannels. This is reason that, when number of subchannels are more (beyond 3), there is not much significant reduction in the data packets drop, as we move from “No Mode Transitions” to “Mode Transition”. We can see almost same number of data packets being dropped with and without Mode Transitions for subchannels more than 3. However, this number is significantly very less as compared to the IEEE 802.11 MAC. From this we can infer that, by doing collision resolutions and subchanneling, we minimize the hidden node and exposed node problem, the main cause for data packets drop and this effect gets even better when we do the Mode Transitions.

In this chapter, we presented several simulation results particularly explaining the advantage of each enhancement we proposed. First, we plotted results for the receiver-initiated protocol without any enhancements and we saw that it performs worse than IEEE 802.11 MAC scheme. Next, we studied the performance of the protocol with the first enhancement of collision resolutions and observed that, there is throughput enhancement by doing collision resolutions as opposed to random backoff. Further from the simulation results we observed that our protocol in presence of two enhancements: collision resolutions and subchannel assignment outperforms IEEE 802.11 MAC scheme in terms of throughput and delay and has less percentage of data packets drop. Moreover, the simulation results also indicate that changing the nature of the topology or the size of the network has no effect on the performance of the protocol. Finally, we studied the performance of the protocol in presence of the third enhancement of Mode Transitions. We observed that keeping

Mode Time = 1.0s and the overlap factor β = 40/60 gives us sufficient amount of overlapping and time for senders to send packets, and provides more throughput as compared to both IEEE 802.11 and our protocol without third enhancement. The main purpose behind implementing Mode Transitions was to address the fairness issues raised because of the receiver-initiated nature of the protocol. The simulation results indicate that the fairness improves in presence of Mode Transitions. Thus, the combined effect of all the three enhancements with receiver-initiated approach improves the performance in terms of throughput, delay and fairness.

CHAPTER FIVE

CONCLUSIONS AND FUTURE WORK

CONCLUSIONS

Efficient MAC protocols are required to ensure efficient and fair sharing of the scarce wireless bandwidth. A good medium access control for wireless LANs should provide an efficient way to share limited channel resources, together with simplicity in operation, high throughput performance, low packet transmission delay, and fairness for serving all nodes. The performance of most common MAC protocols in use today rapidly degrades when nodes retransmit unsuccessful packets that repeatedly collide. The motivation behind this thesis has been to design and study a stable MAC protocol for wireless networks that mitigates the multiple access interference, thereby providing high throughput, low delay and better fairness.

This thesis addresses various issues regarding Medium Access Control (MAC) protocols. Our first contribution was to present a receiver-initiated, five-way handshake protocol for wireless networks based on collision resolutions of RTSs by well known deterministic tree splitting algorithm. In this receiver-initiated approach, we attempt to reduce the collisions of data packets by making each node, before sending RTR, sense the channel for sufficient time to pick up any information of ongoing traffic and not relying on the NAV solely. The collision resolution is done with the help of small control packets exchanged between receiver and its sender(s). We introduced a period of waiting time called *Wait Time* for which a node waits for an RTR from its receiver. *Wait Time* helps to avoid indefinite waiting which might occur a node's intended receiver remained in sending mode forever. We also assigned slightly different waiting times to all the nodes, in order to provide tie breaking in cases when two sender nodes are waiting for an RTR from each other.

Most of the previous work on Medium Access Control (MAC) protocols for wireless ad hoc networks focused on average packet delay, throughput and fairness as the performance metrics.

For more reliable comparisons, we chose the same performance metrics as in some of the previous works and have presented the comparison of our proposed protocol with the standard IEEE 802.11 MAC.

Simulation results show that our protocol performs better than IEEE 802.11 MAC in terms of throughput. The results also indicate that it is beneficial to do collision resolutions with a receiver-initiated approach. Without resolving collisions, senders waste significant time in backoffs, that can be otherwise used for data transfer. Since, the backoffs are not synchronized, at high loads and with many nodes wishing to send, it is hard to guarantee that after the first backoff, similar collisions won't repeat. From simulation results we can infer that, we were able to resolve the collisions among RTSs in time less than time wasted in random backoffs and hence, better throughput than IEEE 802.11 MAC was achieved. Also, the throughput significantly increases as we move from receiver-initiated no collision resolutions approach to that with collision resolutions.

We further identified that a receiver-initiated approach with collision resolutions alone is not sufficient to completely eliminate the hidden node problem. Also the approach still faces the limitations of concurrent transmission due to exposed nodes. Hence, we further extended the receiver-initiated MAC protocol with collision resolutions to utilize multiple subchannels. This enhancement was aimed at minimizing the hidden node problem, by trying to minimize the number of RTRs on each subchannel and hence, collisions due to RTR and at allowing concurrent transmissions in presence of subchannels. The main idea was to make every node send an RTR and further do every data transfer on a (sub)channel which is different than the neighborhoods (sub)channel so that, the data transfer suffers minimum interference. We proposed a subchannel assignment scheme which attempts to assign a subchannel distinct than the 1-hop and 2-hop neighbors but considering the fact that the number of subchannels available for assignment are limited: thus, reusing the subchannels already assigned in the neighborhood, if required. The simulation results show that, there is a considerable improvement in throughput performance by incorporating subchannels. Many concurrent transmissions take place and also, the number of data packets being dropped reduces as we increase the number of subchannels.

However, since the transmission times of the packet increases due to bandwidth subdivision in case of multiple subchannel networks, the enhancement in throughput will be limited. We identified that the throughput increases with the number of subchannels upto a certain limit and then drops. We also noted that this maximum throughput can be achieved only with a handful of subchannels. Experiments were conducted to study the effect of networks of different sizes and nature on this behavior. We observed the protocol gives maximum throughput within a certain range of subchannels even for topologies of varying sizes and varying nature. Multichannel networks also exhibit better delay characteristics. The delay characteristics was also studied for the proposed protocol with multiple subchannels and was observed to have significantly lower delay than that of IEEE 802.11 MAC.

In wireless networks, fair allocation of bandwidth among different nodes is one of the critical problems that affects the QoS of the entire system. The unfairness of MAC has a far reaching impact on the behavior of higher layer protocols and the applications using the network. When the underlying link behavior is unfair, some applications may suffer just because their share is unfairly distributed somewhere else. Thus, MAC level fairness is an important issue. The probability of collision or the number of data packets dropped because of collision and fairness in the allocation of channel to competing nodes affects the efficiency of the MAC protocols.

IEEE 802.11 MAC tries to resolve the collision problem by following a binary exponential backoff. But in this process it aggravates the fairness problem. Following an exponential backoff scheme, the node that succeeds in the last transmission period will gain access to the shared channel again with much higher probability. This might raise fairness issues for some other node whose chance to access the channel is getting thinner with repetitive backoffs. In our receiver-initiated approach we follow a collision resolutions approach instead of random backoffs and in this process handle some of the unfairness present in 802.11 MAC. We also identified that, the receiver-initiated approach may still suffer fairness problems. These occur due to the nature of the protocol, wherein any sender node has to wait for a permission from its receiver node, which itself might be busy sending and hence, will never take the role of a receiver. Fairness problems could be more serious particularly at high loads and in situations where every node has data packets to send.

In order to improve fairness, we proposed enhancements of deliberate transition between receiving mode and sending mode based on some fixed time. These deliberate transitions force each node to behave both a sender as well as a receiver, thus giving fair chance to every node to seek for floor acquisition. The proposed enhancement was aimed at addressing the fairness issues and at the same time maintaining the throughput. Our simulation results validate the fact that giving fair chance to each node to behave as a sender as well as a receiver targets the fairness problem and even improves throughput of the network.

We followed a centralized subchannel assignment strategy for stationary network models which does not account for mobility. Subchannels were assigned statically with the help of a *Composite Function* as described in Sect. 3.4. When nodes move to new locations, the network configuration changes. The degree of the nodes change and so does their 1-hop or 2-hop neighborhood. With these changes, the subchannels that were assigned initially, taking into account the initial configuration now no more remain effective. However, these node movements will not effect the functioning of the protocol. The nodes can still do data transfers based on the initial subchannel assignment. However, the efficiency of the protocol might decrease, because of interference from some new nodes in the neighborhood which were not accounted earlier (before the nodes moved) for subchannel assignment. If the subchannel assignment scheme is modified to account for mobile nodes, then the protocol would provide more efficient results even if the network configuration changes due to mobility.

The protocol works in an environment with multiple subchannels where the number of available subchannels to be used are fixed. We also identified that, even if we increase the number of nodes in the network or changed the configuration of the network (random or mesh), the protocol provides more throughput than 802.11 MAC only with a handful of subchannels. So, the emergence of new nodes does not require addition of new subchannels to handle the increased interference and maintain the same throughput. Since, the number of subchannels over which the nodes communicate are fixed, the transreceivers at each node will be less complex.

FUTURE WORK

The subchannel assignment strategy followed in the protocol is based on the structure of the network, where the subchannels are assigned in the order of highest degree nodes first and the conditions, that the subchannel assigned should occur minimum number of times in 1-hop and 2-hop neighborhood are satisfied. As mentioned earlier, that mobility might effect the efficiency of the protocol because of the increased interference from the new neighborhood. Also, when new nodes join the network they will have to identify their 1-hop and 2-hop neighbors and run the subchnanel assignment algorithm to find a suitable subchannel for assignment. Thus, a distributed approach which accounts for mobility is required so that, the new nodes that join the network or nodes that move to new locations, can be accommodated for subchannel assignment. However, the process of making the subchannel assignment dynamic would require addition of new packets for neighbor discovery and to communicate various information about subchannel assignment amongst 1-hop and 2-hop neighbors. And this might effect the throughput performance of the protocol.

Since the subchannel assignment scheme is static, at any given point it might not fully reflect the current usage of the network. Some subchannels might be absolutely idle or with very low traffic on it, which in the beginning was neglected as a candidate during subchannel assignment because of its occurrence in neighborhood. We can improve the performance of the protocol, if we can identify the current usage of the channel and do a reassignment of subchannels based on this info. This requires again a dynamic approach which is able to, as traffic patterns changes, do a reassignment by reassigning a subchannel with less demand to that with higher demand.

One of the future work includes dynamically adapting the subchannel used by a receiver in response to its neighborhood and further improving the fairness. By doing a reassignment to meet current demands clearly makes it possible to use the bandwidth more efficiently than is possible with a static allocation. However, the tradeoff is the additional computation that needs to be done in keeping track of the changing traffic patterns and in deciding which subchannels stand a better chance to be considered for reassignment. So, in order to gain sufficiently from dynamic subchannel assignment such that the performance improvement shows up, the additional computation needs to be minimized.

From the simulations results we observed that, by doing deterministic collision resolution amongst RTSs sent by multiple senders we make efficient use of bandwidth and hence, we can see an increase in throughput. Comparatively, very less time is wasted in doing collision resolutions as otherwise would have been wasted, if data packets were to collide. Some time is consumed in resolving collisions in every CRI and this would be large if the length of the CRI is large. Many a times, the same sender nodes compete to send RTS to a receiver node resulting in a similar kind of steps in each CRI. We can reduce the number of collisions among the same senders by resolving the collisions once and later remembering the order in which the collisions were resolved. With this, we will be able to allow the nodes who have already participated in the initial CRI's to now send data packets without asking any permission in the form of RTS.

To maintain this order, we might require some kind of a Transmission Group in which the nodes get added after every collision resolution. Advantages of Group Transmissions include that once a station has reserved a position in the group-transmission period, it will be able to transmit at or better than a guaranteed rate. Also, with Transmission Group the protocol might seem to be more stable under heavy loads, because it will permit stations in the transmission group to send packets independently of new requests for additions to the transmission group. So by maintaining Transmission Group the time required to solve the total collisions (particularly if they are from same sender nodes) would be less and instead can be significantly utilized for data transfer. However, maintenance of such a group is not easy and requires frequent updation when the nodes leave or join the group, particularly when deliberate mode transitions are allowed.

BIBLIOGRAPHY

[1] IEEE Computer Society LAN MAN Standards Committee ed., *IEEE standard for wireless LAN medium access control (MAC) and physical layer (PHY) specifications*, IEEE Std 802.11-1997, The Institute of Electrical and Electronics Engineers, New York, 1997.

[2] The Network Simulator - ns2, *http://www.isi.edu/nsnam/ns*

[3] The CMU Monarch Project. Wireless and Mobility extensions to ns, *http://www.monarch.cs.cmu.edu*.

[4] A. Acharya, A. Misra and S. Bansal, *MACA-P: A MAC for Concurrent Transmissions in Multi-hop Wireless Networks*, IBM Research Report, RC22528(W0207-086), July 18, 2002.

[5] B. Bensaou, Y. Wang and C. C. Ko, *Fair medium access in 802.11 based wireless ad-hoc networks*, Proceedings of the 1st ACM international symposium on Mobile ad hoc networking and computing, pg 99-106, 2000

[6] A. A. Bertossi and M. Bonuccelli, *Code Assignment for Hidden Terminal Interference Avoidance in Multihop Packet Radio Networks*, IEEE/ACM Transactions On Networking, Volume 3, No. 4, 1995.

[7] A. A. Bertossi, M. Brunato and R. Battiti, *Distributed Code Assignment in Multihop Radio Networks: Object-Oriented software Simulations*, Proceedings of SoftCOM 2000 - Rijeka(Croatia), 2000.

[8] D. Bertsekas and R. Gallager, *Data Networks*, Second Edition, Prentice-Hall, 1992.

[9] V. Bharghawan, A. Demers, S. Shenker and L. Zhang, em MACAW: A media access protocol for wireless LANs, Proceedings of ACM SIGCOMM, 1994.

[10] C. Y. Chang, C. T. Chang and P. C. Huang, *Dynamic channel assignment and reassignment for exploiting channel reuse opportunities in ad hoc wireless networks*, Communication Systems, ICCS, Volume 2, 2002.

[11] K. C. Chua, *Performance analysis of Multichannel CSMA/CD network with noisy channel*, IEEE ICC'91, 1991.

[12] J. Deng and Z. Haas, *Dual Busy Tone Multiple Access (DBTMA): A New Medium Access Control for Packet Radio Networks*, Florence, Italy, 1998.

[13] C. L. Fullmer and J. J.Garcia-Luna-Aceves, *Floor Acquisition multiple access(FAMA) for packet radio networks*, Proceedings of ACM SIGCOMM'95, Cambridge, MA, 1995.

[14] C. L. Fullmer and J. J.Garcia-Luna-Aceves, *Solutions to Hidden Terminal Problems in Wireless networks*, Proceedings of ACM SIGCOMM, 1997.

[15] C. L. Fullmer and J. J.Garcia-Luna-Aceves, *Floor Acquisition multiple access(FAMA) in single-channel wireless networks*, Proceedings of ACM Mobile Networks and Applications, Volume 4, Issue 3, New York, 1999.

[16] R. Garces and J. J.Garcia-Luna-Aceves, *Collision Avoidance and Resolution Multiple Access-First Success Protocols,* INFOCOM, 1997.

[17] R. Garces and J. J.Garcia-Luna-Aceves, *Collision Avoidance and Resolution Multiple Access (CARMA),* Cluster Computing, Volume 1, No. 2, pages 197-212, 1998.

[18] J. J.Garcia-Luna-Aceves and R. Garces, *Collision Avoidance and Resolution Multiple Access with Transmission Groups*, supported by Defense Advanced Research Projects Agency, 1997.

[19] J.J. Garcia-Luna-Aceves and J. Raju, *Distributed Assignment of Codes for Multihop Packet Radio Networks*, Proceedings of IEEE MILCOM '97, Monterey, California, 1997.

[20] J. J.Garcia-Luna-Aceves and R. Garces, *Collision Avoidance and Resolution Multiple Access for Multichannel Wireless Networks*, Proceedings of IEEE INFOCOM, 2000.

[21] A. C. V. Gummalla and J. O. Limb, *Wireless medium access control protocols*, IEEE Communications Survery, *http://www.comsoc.org/pubs/ surveys*, 2000, Second Quarter.

[22] H. Y. Hsieh and R. Sivakumar, *Improving throughput and fairness in multi-hop wireless networks*, in Proceedings of ICN, Colmer, France, 2001.

[23] N. Jain, S. Das and A. Nasipuri, *A Multichannel CSMA MAC protocol with Receiver-based channel selection for Multihop Wireless Networks*, in Proceedings of the 9th Int. Conf, on Computer Communications and Networks (IC3N), 2001.

[24] R. Jain, A. Durresi, and G. Babic, *Throughput Fairness Index: An Explanation*, ATM Forum/99-0045, 1999.

[25] S. Jiang and T. Hsiao, *Performance Evaluation of a Receiver-Based Handshake Protocol for CDMA Networks*, IEEE Transactions on Communication, 43, 1995.

[26] P. Karn, *MACA - a new channel access method for packet radio,* in ARRL/CRRL Amateur Radio 9th Computer Networking Conference, pages 134-140, ARRL, 1990.

[27] L. Kleinrock and F. A. Tobagi, *Packet switching in radio channels: Part I - Carrier sense multiple-access modes and their throughput-delay characteristics*, IEEE Transactions on Communications, Volume 23(12), pages 1400-1416, 1975.

[28] C. E. Koksal, H. Kassab and H. Balakrishnan, *An analysis of short-term fairness in wireless media access protocols*, Measurement and Modelling of Computer Systems, pages 118-119, 2000.

[29] C. G. Lof, *Packet delay for CSMA and Multi-channel ALOHA muticast schemes in WLANs with fading and co-channel interference*, IEEE International Conference on Universal Personal Communications, 1996.

[30] A. M. Marsan and F. Neri, *A simulation study of delay in multichannel CSMA/CD protocols*, IEEE Transactions on Communication, Volume COM-39, No. 11, 1991.

[31] A. Muir and J. J. Garcia-Luna-Aceves, *Group Allocation multiple access with collision detection*, Proceedings of IEEE INFOCOM'97, Kobe, Japan, 1997.

[32] A. Nasipuri, J. Zhuang and S. R. Das, *A Multichannel CSMA MAC protocol for Multihop Wireless Networks*, in IEEE Wireless Communications and Networking Conference (WCNC), 1999.

[33] T. Ozugur, M. Naghshineh, P. Kermani and J. A. Copeland, *Fair media access for wirless LANs*, Proceedings of IEEE GLOBALCOM, December 1999.

[34] D. Shukla, L. Chandran-Wadia and S. Iyer, *Mitigating the exposed node problem in IEEE 802.11 adhoc networks*, Accepted in IEEE International Conference on Computer and Communication Networks (ICCCN) , Dallas, USA, 2003.

[35] R. Srinivasan and A. K. Somani, *On Achieving Fairness and Efficiency in High-speed shared medium access*, IEEE/ACM Transactions on networking, Volume 11, No.1, 2003.

[36] F. Talucci, M. Gerla and L. Fratta, *MACA-BI (MACA by invitation) - a receiver oriented access protocol for wireless multihop networks*, Proceedings of IEEE PIMRC, 1997.

[37] Z. Tang and J. J. Garcia-Luna-Aceves, *Hopreservation multiple access (HRMA) for multichannel packet radio networks*, Proceedings of the IEEE IC3N'98, Seventh International Conference on Computer Communications and Networks, 1998.

[38] F. A. Tobagi and L. Kleinrock, *Packet switching in radio channels: Part II - The hidden terminal problem in carrier sense multiple-access modes and the busy-tone solution*, IEEE Transactions on Communications, Volume 23(12), pages 1417-1433, 1975

[39] A. Tzamaloukas and J. J.Garcia-Luna-Aceves, *Receiver initiated Collision Avoidance in Wireless Networks*, Wireless Networking, Volume 8, Issue 2/3, 2002.

[40] S. L. Wu, C. Y. Lin, Y. C. Tseng and J. P. Sheu, *A New Multichannel MAC protocol with on-demand channel assignment for Multihop Mobile Ad Hoc Networks*, IEEE Wireless Communications and Networking Conference (WCNC), Chicago, IL, 2000.