A Cross Layer Analysis of TCP Instability in Multihop Ad hoc Networks

Ehsan Z. Hamadani

A Thesis Submitted for the Degree of

Doctor of Philosophy

to the School of Engineering and Mathematical Sciences,

City University, London

August 2007

To My Family

Dad, Mum, Amin and Marjan

And To My Dearest Wife

Sanaz

Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my deepest and sincere appreciation to my supervisor Dr. Veselin Rakocevic for his constant and comprehensive support during last years. From the very beginning stages of applying for doing the PhD and sorting my grant to insightful comments and helpful advices throughout this work, he has been always extremely supportive and patient. Our discussions and arguments helped greatly in refining the ideas in this thesis from their original vague form into ones that can be illuminated clearly and concisely. I must further extend my gratitude to Dr. W. Boyle and Dr. M. Rajarajan for their help and advise.

In addition, I was quite lucky to benefit from a friendly and enthusiastic environment in our office, which could not exist without my friends Takis, Kalid, Ibrahim, Dasun, Peyman, Sorush and Dia. I would also like to thank Mostafa for his excellent ideas and collaboration in helping me with mathematical analysis throughout the thesis. Special thanks should also go to Pierre and Thomas for spending considerable amount of time to setup and build the testbed.

Lastly and more importantly, I would like to give my deepest gratitude to my family for all their guidance and support they have given me throughout my life. They have, more than anyone else been the reason I have been able to get this far. Words cannot express my gratitude to my parents, my brother and sister who give me their support from across the country. Thank you all for giving me the support and courage to take the path in life that I have chosen.

Declaration

The University Librarian of the City University may allow this thesis to be copied in whole or in part without any reference to the author. This permission covers only single copies, made for study purposes, subject to normal conditions of acknowledgements.

Abstract

Multihop ad hoc networks are collection of wireless nodes dynamically forming a temporary network without the use of any preexisting network infrastructure or centralized administration. Consequently, ad hoc networks and their wireless links are fundamentally different from conventional stationary wireless and wired computer networks. In particular, incorporating the concept of TCP end-to-end congestion control for wireless networks is one of the primary concerns in designing ad hoc networks since TCP was primarily designed and optimized based on the assumptions for wired networks. In this thesis, our interest lies on tackling the TCP instability problem since due to the nature of applications in multihop ad hoc networks (e.g. emergency operation and battlefield communication), connection instability or starvation even for a short period of time can have a devastating impact on the Quality of Service and may not be acceptable for the end user. Through a detailed analysis and simulations, it is shown in this thesis that the main causes of TCP instability lie in three factors: overloading the network by sending more packets than the capacity of the channel, TCP sensitivity to out of order packets, and the channel access unfairness when multiple TCP connections are sharing the medium using 802.11 MAC protocol. To reduce TCP instability caused by excessive packet contention drops, a novel algorithm has been proposed that aims to reduce packet contention by optimizing the amount of outstanding data in the network. To reduce TCP sensitivity to out of order packets, a new algorithm is proposed with the aim of performing more local recovery rather than end-to-end recovery. Finally, to address the TCP instability caused by channel access unfairness, the 802.11 binary exponential backoff algorithm is replaced with a more conservative approach. In addition to addressing the problem of TCP instability, a 3-dimensional Markov model of 802.11 MAC is presented in this thesis to accurately analyze the 802.11 MAC throughput in ad hoc networks.

Acronyms

ACK	Acknowledgments
AIMD	Additive Increase Multiplicative Decrease
ARQ	Automatic Repeat reQuest
BDP	Bandwidth Delay Product
BEB	Binary Exponential Backoff
CCD	Cumulative Contention Delay
CDF	Contention Delay Field
CDH	Contention Delay per Hop
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
CTS	Clear to Send
ctwnd	TCP ConTention Window
CW	Link layer Contention Window

cwnd	TCP Congestion Window
DCF	Distributed Coordination Function
DCR	Delayed Congestion Response
DIFS	DCF Inter-Frame Space
DSR	Dynamic Source Routing
DSSS	Direct Sequence Spread Spectrum
EIFS	Extended Inter-Frame Space
ELR	Extended Link layer Retransmission
FAMA	Floor Acquisition Multiple Access
FBA	Fair Backoff Algorithm
FTP	File Transfer Protocol
IFS	Inter-Frame Space
IP	Internet Protocol
ISN	Initial Sequence Number
IWS	Initial Window Size
LRL	Long Retry Limit
MAC	Medium Access Control
MACA	Multiple Access with Collision Avoidance

MACAW	Multiple Access with Collision Avoidance for Wireless
MACK	Medium Acknowledgment
MCDH	Mean Contention Delay per Hop
MSS	Maximum Segment Size
NAV	Network Allocation Vector
OLSR	Optimized Link State Routing
PCF	Point Coordination Function
PDA	Persona Digital Assistant
QoS	Quality of Service
RED	Random Early Detection
RREP	Route REPly
RREQ	Route REQuest
RTHC	Round Trip Hop Count
RTO	Retransmission TimeOut
RTS	Request to Send
RTT	Round Trip Time
rwnd	TCP Receiver Window
SIFS	Short Inter-Frame Space

SNR	Signal To Noise Ratio
SRL	Short Retry Limit
SRTT	Smoothed Round Trip Time
ssthresh	TCP Slow Start threshold
SWM	Sliding Window Method
swnd	TCP Send Window
SYN	Synchronization
TCP	Transport Control Protocol
TCTC	TCP ConTention Control
UDP	User Datagram Protocol
WLAN	Wireless Local Area Network

Chapter 1

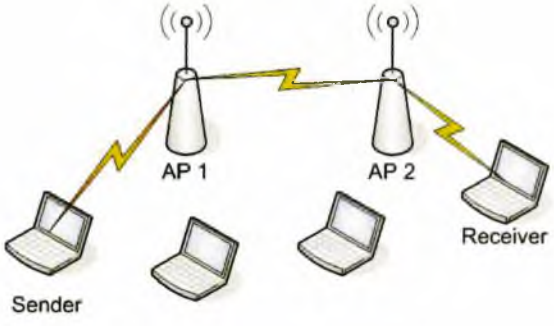
Introduction

Wireless communications have experienced an explosive growth in recent years thanks to the wide availability and rapid deployment of wireless transceivers in a variety of computing devices such as Personal Digital Assistant (PDAs) and laptops. In particular, the integration of cellular systems, wireless local area networks (WLANs), and personal area networks (PANs) are just the beginning of the "wireless revolution" with the ultimate goal of uncompromis connectivity [1, 2]. Although infrastructure-based wireless networks enable mobile devices to get network services, it takes time to set up the infrastructure network, and the costs associated with installing the infrastructure can be quite high. There are, furthermore, situations in which user-required infrastructure is not available, cannot be installed, or cannot be installed in time in a given geographic area. Moreover, with the advance of wireless communication technology, portable computers with radios are being increasingly deployed in common activities. Scenarios such as conferences, meetings, lectures, crowd control, search and rescue, disaster recovery, and automated battlefields which typically do not have central administration

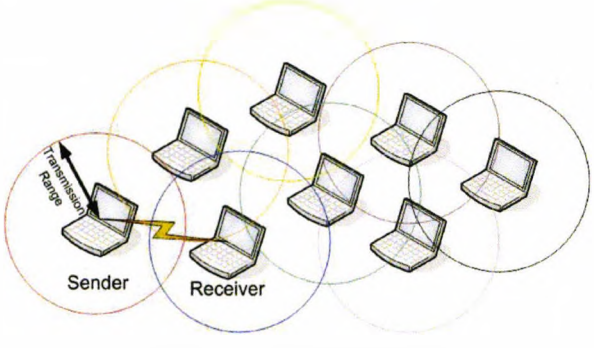
or infrastructure are becoming more and more available [3]. The nature of these applications should allow mobile computer users to set up a possibly short-lived network just for the communication needs of the moment.

For all of these reasons, *ad hoc networks* have gained increased attention in recent years thanks to their easy deployment, maintenance and application variety. Since in ad hoc networks no fixed infrastructure is dedicated for relaying packets from sender to receiver (end-to-end), every node should act both as a host and as a router. To be more precise, if the end-to-end sender and receiver in ad hoc networks are in the transmission range (as would be defined later in this chapter) of each other, they form a *single-hop ad hoc network*. However, due to transmission limitations in mobile devices, in most of the cases several mobile nodes might be involved in relaying packets from sender to receiver. This will result into establishing *multihop ad hoc network*.

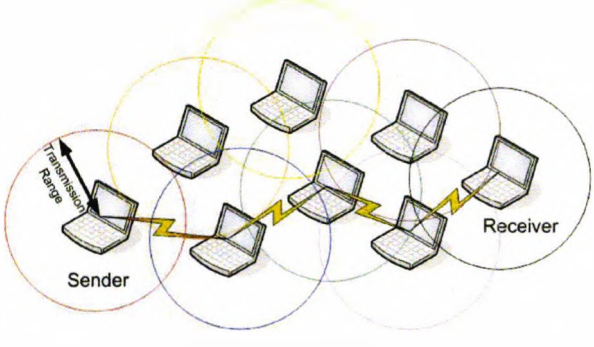
Figure 1.1 depicts the differences between the infrastructure, single-hop and multihop mobile ad hoc networks.



(a) Infrastructure WLAN



(b) Single Hop Ad hoc Network



(c) Multi Hop Ad hoc Network

Figure 1.1: Different WLAN formation

Since the formation of multihop ad hoc networks in real scenarios is more likely than single-hop ad hoc networks, throughout this thesis we mainly focus on multihop ad hoc networks and use the terms ad hoc networks and multihop ad hoc networks interchangeably.

1.1 Ad hoc Networks Characteristics

Multihop ad hoc networks share many of the properties of infrastructure WLANs but also posses certain unique feature which are derived from the distributed operation and the multi-hop nature of ad hoc networks. In this section, the main challenges that face contention based ad hoc networks (such as IEEE 802.11) are reviewed briefly.[1]

1.1.1 Channel Interference and Signal Attenuation

Contrary to wired networks, wireless networks are prone to error losses and the channel is unprotected from outside interferences. In particular, signals over wireless medium are prone to high interference from external sources that operate in the same frequency range. For instance, microwaves and Bluetooth devices that operate in the same frequency range as 802.11 devices (2.4 GHz) may cause interference [4]. In addition, multipath fading [5] and path asymmetry [6] can seriously deteriorate the signal quality and system performance.
Furthermore, as the transmitted signal spreads out from the antenna in all directions, it attenuates as distance increases. More precisely, for an omnidirectional

[1]The challenges discussed in this section are mainly from the lower layer point of view namely link layer and physical layer

transceiver, three ranges from the sender's perspective may be identified [7] due to signal attenuation effect:

1. *Transmission Range (R_{tx})*: The range within which a transmitted frame can be successfully received by the intended receiver. Within this range, the Signal-to-Noise Ratio (SNR) is high enough for a frame to be decoded by the receiver. The transmission range is mainly determined by transmission power and radio attenuation.

2. *Carrier Sensing Range (R_{cs})*: The range within which the transmitter triggers carrier sense detection. When this happens, the medium is considered busy and the sensing node defers transmission. This is usually determined by the antenna sensitivity.

3. *Interference Range (R_i)*: The range within which an intended receiver may be subject to interference from an unrelated transmission, thereby suffering a loss. This range largely depends on the distance between the sender and the interfering node.

We should note that in general $R_{tx} \leq R_i \leq R_{cs}$, as the energy required for a signal to be decoded is greater than what is needed to cause interference [8].

1.1.2 Hidden and Exposed Terminal Problems

Contention based multihop ad hoc networks are subjected to two well-known problems known as hidden terminal and exposed terminal [9, 10]. First to see the cause of the hidden node phenomenon, let us consider figure 1.2 where the

transmission range of each node is determined by a circle around the node [2].

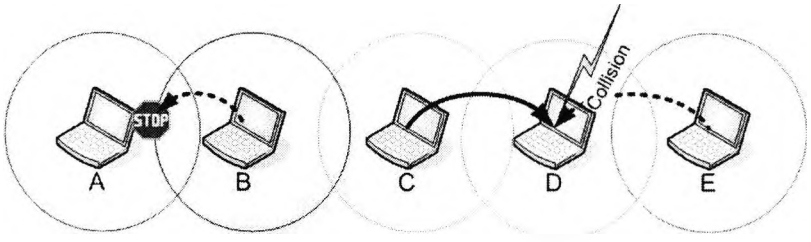


Figure 1.2: Illustration of hidden and exposed terminal problems

Here node C is transmitting a data packet towards node D. Meanwhile, following the physical carrier sensing, node E detects the channel as idle (since C is hidden from E). Consequently, E also start transmitting data to node D, causing collisions and bandwidth wastage as both data transmissions are destroyed. This is known as the hidden terminal problem.
In the same example, after node C reserves the channel for transmitting packet to node D, node B is put in the silent mode by sensing the channel idle. Hence, node B is unable to send any packet towards node A even this transmission does not interfere with node C to D ongoing transmission (since the collision occurs in the receiver). This phenomenon known as exposed terminal problem clearly results to channel usage under-utilization and substantial throughput degradation.

We should note that both the hidden and exposed terminal effects are related to the transmission range. As the transmission range increases, the hidden terminal effect becomes less prominent because the sensing range increases. Nonetheless, the exposed terminal effect then becomes more prominent as a greater area is "reserved" for each transmission.

[2]In this examples and for simplicity throughout the rest of this thesis, unless otherwise specified, the R_{tx}, R_i and R_{cs} ranges are all assumed to be equal

1.1.3 Spatial Channel Reuse

Spatial channel reuse refers to the number of concurrent transmissions that may occur in the network without interfering with each other. To get a better understanding of spatial channel reuse, consider a chain topology in figure 1.3 where only communication between pairs could happen concurrently in turn, as long as each pair is 3 hops apart from the other due to transmission interference between nodes. For instance, in this example, nodes A→B and D→E may communicate simultaneously

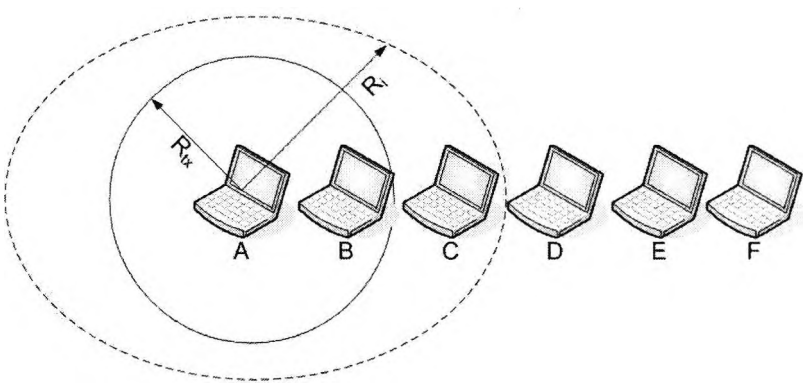


Figure 1.3: Illustration of channel spatial reuse

Since at most two pairs can transmit at the same time without affecting each other, the spatial reuse of this string topology is 2. It should be noted that the spatial reuse in a particular scenario represents an optimal level of concurrency and any attempt of more simultaneous transmission in the network will result in interference and cause contention drops as would be explained thoroughly in chapter 4. This is clearly in contrast with wired networks where packet drops are mainly caused by buffer overflows at the routers.

1.2 Motivation

As mentioned earlier, during recent years multihop ad hoc networks have attracted considerable research interest. In particular, the de facto adoption of the popular IEEE 802.11 standard [11] has further fuelled the deployment of wireless transceivers in a variety of computing devices by ensuring inter-operability among vendors thereby aiding the technology's market penetration. However, as initially the deployment of these wireless technological advances came in the form of an extension to the fixed LAN infrastructure model, the 802.11 standard was mostly evolved and optimized for infrastructure-based wireless LANs rather than ad hoc networks.

In the transport layer, to enable seamless integration of ad hoc networks with the Internet, TCP [12] seems to be the natural choice for users of ad hoc networks that want to communicate reliably with each other and with the Internet. Here also, despite in theory TCP should not care whether the network layer is running over wired or wireless connections, in practice, this does matter because TCP has been carefully optimized based on assumptions that are specific to wired networks. For instance, since bit error rates are very low in wired networks, nearly all TCP versions assume that packet losses are due to congestion and therefore invoke their congestion control mechanism in response to such losses. On the other hand, because of wireless medium characteristic and multihop nature of ad hoc networks, such networks exhibit a richer set of packet losses, including medium access contention drops, random channel errors and route failure where in practice each are required to be addressed differently. Ignoring these properties of wireless ad hoc networks can obviously lead to poor TCP performance as shown in many research studies (e.g. [13–21])

Not surprisingly, multihop ad hoc networks exhibit serious performance issues when TCP runs over IEEE 802.11 as neither protocols have been designed based on the properties of such networks. This also implies that TCP's poor performance in 802.11 based ad hoc networks is truly a cross layer problem which needs to be addressed by considering the interaction of multiple layers with each other [22]. In other words, any attempt to alleviate the TCP problems in ad hoc networks should be done through investigating the characteristics and impact of deployed link layer and routing protocols on TCP.

During recent years, a number of research studies have highlighted some of the main problems TCP encounters in ad hoc networks [6, 15, 21, 23–29]. However, one of the key areas that has not attracted enough attention and needs to be addressed further in the deployment of TCP in ad hoc networks is the problem of TCP instability. This becomes more clear if we realize due to the nature of ad-hoc network applications (e.g. emergency operation, battlefield communication) the disconnectivity or starvation of one or more connections for even a short period of time is not acceptable and can have a devastating impact on QoS. More specifically, the ad hoc network users are more willing to receive a continuous and stable flow of data rather than sending/receiving large bulk of data instantly. This observation has motivated us to fully investigate the cause of TCP instability in ad hoc networks and propose a set of cross layer solutions to address the problem throughout this dissertation.

One valid argument that might arise from the above discussion in analyzing the TCP in ad hoc networks throughout this thesis is the legitimacy of the choice of IEEE 802.11 as the adopted MAC protocol in this thesis. The answer to this issue lies in the fact that during recent years, 802.11 technology has successfully

dominated the commercial market for wireless transceiver devices and very likely it will keep its share. On the other hand, ad hoc networks are mostly temporary short lived networks that in most of the situations are not the primary mode of connectivity for the users [30]. Therefore, the choice of adding new wireless technology optimized for ad hoc networks might not look feasible and cost-effective for users at least in coming years. In addition, from the performance wise point of view, the study in [14] has shown that in most cases 802.11 came on top in terms of both TCP throughput and fairness in comparison to different MAC protocols, namely CSMA, FAMA [31], and MACAW [32]. Therefore, we believe the IEEE 802.11 MAC will keep its dominance in future WLAN generations and hence is considered as the underlying link layer protocol in the thesis.

1.3 Contributions

The goals set for this thesis derive from the motivations as listed in the previous section and can be outlined as follows:

- The first contribution of the thesis is a thorough and detailed analysis of TCP instability problem in ad hoc networks. In particular, by identifying different cause of TCP instability, we show the complex interaction of TCP and 802.11 MAC in creating instability. The significance of the results from this part is to identify and locate the root of the TCP instability issue in ad hoc networks.

- The second contribution of this work is developing an accurate analytical model of 802.11 MAC using 3-dimensional Markov chain. In addition, by using the model, the analytical values of 802.11 MAC parameters for which the maximum link layer throughput is achieved are calculated. The importance of such contribution is its comprehensibility in taking into account realistic channel conditions (e.g. random packet errors) and 802.11 MAC parameters (e.g. short and long retry limits) in contrast to most models developed so far. On the other hand, the way in which analytical link layer throughput is calculated has been improved by excluding the share of control packets throughput from the achieved throughput.

- The third contribution of the thesis is developing an algorithm that estimates the optimum amount of outstanding TCP segments in the network for which the level of contention experienced by TCP packets are kept as low as possible without compromising TCP throughput. The main implication of this algorithm is to tackle a major source of TCP instability by

adjusting the TCP transmission rate from various sources.

- The fourth contribution of the thesis is to further alleviate TCP instability by substituting the binary exponential backoff algorithm used in 802.11 MAC with an alternative backoff algorithm that provides more fairness among contending stations while keeping the channel idle slots reasonably low.

- Finally, the last contribution of the thesis is to validate the correct functionality of the TCP and 802.11 modules of the simulator with the results from testbed which was built and developed throughout this thesis.

1.4 Structure of the Thesis

The rest of this dissertation is organized as follows. Chapter 2 covers in details the IEEE 802.11 MAC and TCP protocol. Further, a description of the routing protocol used in the simulation and testbed is presented.

Chapter 3 presents the analytical model of the operation of 802.11 MAC in ad hoc networks. In addition, the impact of different 802.11 parameters on maximum throughput theoretically achievable are investigated and discussed in detail.

Chapter 4 analyzes the underlying cause of TCP instability by giving a number of simple but yet important examples that will shed light on the roots of the problem. In particular, we first break down the TCP instability into intra-flow and inter-flow instability. Then, an in depth cross layer analysis of the chain of events occurring between different layers are discussed. Finally, the chapter reveals some interesting facts which will make the building blocks of our solutions in the next chapter to eliminate and tackle the TCP instability in multihop ad hoc networks.

Chapter 5 presents a set of proposed cross layer solutions that alleviate the instability issue in multihop ad hoc networks. Specifically, the chapter first introduces TCP Contention Control that adjusts the amount of outstanding data in the network by taking into account the level of contention experienced by packets. To address the TCP instability caused by channel access unfairness seen among contending nodes, the chapter introduces the Fair Backoff Algorithm. In addition, to decrease TCP sensitivity to out-of-order packets, an extended version of the Delayed Congestion Response algorithm with the aim of performing more

localised recovery than end-to-end recovery is proposed.

Chapter 6 presents the simulation tool and verification methodology used to evaluate the results. In this chapter, the simulation tool and its parameters used to evaluate the proposed schemes against default operation of TCP in 802.11 ad hoc networks are first introduced. This is then followed by a explantation of the testbed platform that was setup to confirm and validate the simulation tool.

Chapter 7 contains the simulation results of the proposed techniques in a variety of scenarios. In this section, a wide range of metrics are monitored in multiple layers to investigate the overall effectiveness and pros and cons of the new algorithms proposed in chapter 5 when they all work together.

Finally, Chapter 8 summarizes the results presented in this study and offers suggestions for future research in this area.

Chapter 2

Deployment of TCP/IP in Multihop Ad hoc Networks

2.1 Introduction

The main objective of this chapter is to provide background on the characteristics of the TCP/IP protocols used in ad hoc networks, which are referred throughout this thesis. As such, the chapter is organized as follows. In section 2.2, we investigate the IEEE 802.11 MAC standard in fine detail. Section 2.3 describes the operation of TCP and its end-to-end error detection/recovery mechanisms. Finally, section 2.4 contains a succinct description of the main operations of the DSR routing protocol.

2.2 IEEE 802.11 MAC

2.2.1 General Description

The IEEE 802.11 MAC [11] specifies two modes of MAC protocol: contention-based service provided by the Distributed Coordination Function (DCF) and an optional contention-free service implemented by the Point Coordination Function (PCF). The PCF is based on a polling scheme and uses a Point Coordinator (such as access point) that cyclically polls stations, giving them the opportunity to transmit. The major advantage of PCF is that it can guarantee maximum packet delay and thus provide QoS. However, since a centralized coordinator should do the polling this scheme cannot be adopted in the ad hoc mode. Therefore, in this section we just describe the DCF mode.

The DCF in IEEE 802.11 is based on the principles of Carrier Sensing Multiple Access with Collision Avoidance (CSMA/CA). To overcome the hidden terminal problems in CSMA/CA as described before, IEEE 802.11 MAC uses the same technique as proposed by Karn in MACA [33]. In essence, in 802.11, the nodes wishing to communicate with each other use two short signaling packets called request to send (RTS) and clear to send (CTS) as a handshake prior to actual data transmission. All nodes that hear the RTS or/and CTS message(s), defer for the amount of time specified in Network Allocation Vector (NAV) before they can sample the channel again for idle status. This procedure is generally known as virtual carrier sensing and the channel is marked busy if either the physical or virtual carrier sensing mechanisms indicate the channel is occupied. In addition, the use of the RTS/CTS mechanism is under control of the RTS-Threshold attribute that can be set on a per station basis. This mechanism allows

stations to be configured to use RTS/CTS either always, never, or only on frames longer than a specified length in the RTS-Threshold.

The next important feature of the 802.11 is to provide reliable link-to-link data delivery. This implies that the packet receiver should immediately send a back positive acknowledgment so that retransmission can be scheduled by the sender if no Medium ACK (hereafter called MACK) is received. We should note that the NAV contains a time value that represents the duration up to which the wireless medium is reserved because of the transmission of the actual data packet and its corresponding MACK packet.

2.2.2 Timing

One of the key deterministic factors in accessing the medium is the priority to access the channel, which is controlled by inter-frame space (IFS) time intervals between the transmissions of frames [11]. The IFS intervals are mandatory periods of idle time on the transmission medium and are used for smooth functioning of the medium access algorithm. Three IFS intervals are specified in the IEEE802.11 DCF standard: short IFS (SIFS), DCF-IFS (DIFS), and Extended IFS (EIFS). The SIFS interval is the smallest IFS, followed by DIFS and EIFS, respectively.

SIFS is used when stations have seized the medium and need to keep it for the duration of the frame exchange sequence to be performed. This implies that stations that are sending MACK, and CTS can access the medium after SIFS. Using the smallest gap between transmissions within the frame exchange sequence prevents other stations, which are required to wait for the medium to be idle for a longer gap, from attempting to use the medium, thus giving priority to complete

the frame exchange sequence in progress.

DIFS is used to transmit normal data frames. A station using the DCF can transmit if its carrier-sense mechanism determines that the medium is idle and its backoff timer has expired. The duration of the DIFS is defined as

$$DIFS = SIFS + 2\delta \tag{2.1}$$

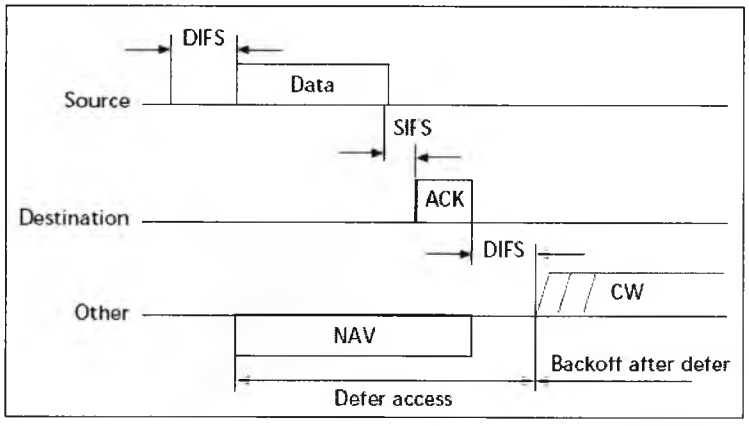
where δ is the value of one slot time.[1]

EIFS is used by the DCF whenever the physical carrier sensing has indicated to the MAC that a frame transmission was begun that did not result in the correct reception of a complete MAC frame. The EIFS is defined to provide enough time for another station to acknowledge what was, to this station, an incorrectly received frame before this station commences transmission. Reception of an error-free frame during the EIFS terminates the EIFS timer and brings the station to the normal medium access state (using DIFS). Figure 2.1(a) and 2.1(b) depict a timing diagram illustrating the successful transmission of a data frame without and with using RTS/CTS messages, respectively [11].

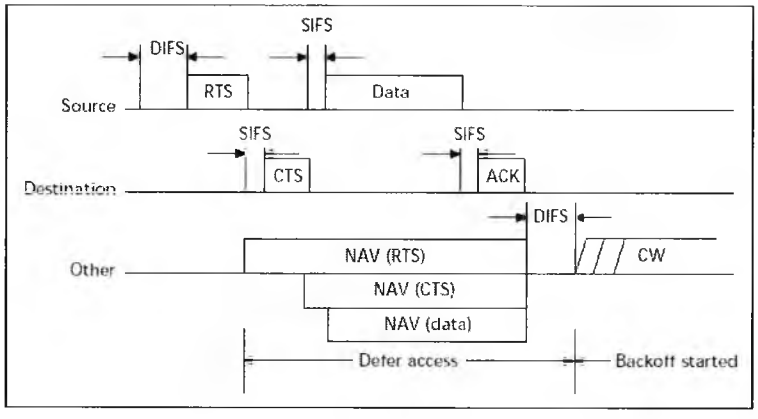
2.2.3 Random Backoff and Basic Access

As mentioned earlier, a station desiring to initiate transfer of data frames should invoke the carrier-sense mechanism to determine the busy/idle state of the medium. Now, if the medium is busy, the station should defer until the medium is determined to be idle without interruption for a period equal to DIFS (when the

[1]Slot-time depends on the technology deployed in physical layer. For instance, in FHSS the value is 50μs and for DSSS it is 20μs



(a) Transmission timing of frame without use of RTS/CTS



(b) Transmission timing of frame with use of RTS/CTS

Figure 2.1: IEEE 802.11 timing specifications

last frame detected on the medium was received correctly) or EIFS (when the last frame detected on the medium was not received correctly). After this DIFS or EIFS medium idle time, the station generates a random backoff period for an additional deferral time before transmitting, unless the backoff timer already

contains a nonzero value, in which case the selection of a random number is not needed and not performed. This random backoff aims to minimize collisions during contention between multiple stations that have been deferring to the same event. More precisely:

$$BackoffTime = Random() * SlotTime \tag{2.2}$$

where Random() is a random integer number of time slots drawn from a uniform distribution over the interval [0,CW]. Here, CW (Contention Window) is an integer number within the range of CW_{min} and CW_{max} where the value of CW_{min} and CW_{max} depend on the adopted physical layer. For example, in DSSS physical layer CW_{min} and CW_{max} values are 31 and 1023, respectively [11]. At the beginning of each frame transmission, the CW takes an initial value of CW_{min} and is doubled every time the node is unsuccessful to transmit its frame until the CW reaches the value of CW_{max}. Once it reaches the CW_{max}, the CW remains at the value of CW_{max} until it is reset. On the other hand, the CW is reset to CW_{min} after every successful attempt to transmit a frame or the frame is dropped after several tries (specified by Max-Retry-Limit as would be explained later). This backoff scheme is called the Binary Exponential Backoff (BEB).

After choosing the new backoff time, the node uses the carrier-sensing mechanism to determine whether there is activity during each backoff slot or not. If no medium activity is indicated, the backoff timer is decremented by one slot-time. If the medium is determined to be busy at any time during the backoff process, then the backoff timer is suspended. When the medium becomes free again, the node waits for another DIFS and if it is still free, it starts to decrement its backoff timer slot by slot. This process is repeated until the time the backoff counter

reaches zero where the frame would be emitted then. At the receiver side (next hop), if the packet is received without error, the receiver issues a MACK packet after SIFS, confirming the receipt of the packet. This receipt of the MACK will also reset the CW at the sender to CW_{min}. Otherwise, the sender enters the error recovery procedure that is described in the following section.

2.2.4 Recovery Procedure and Retransmit Limits

As stated earlier, 802.11 MAC is based on the CSMA/CA scheme in which stations are not able to detect a collision by hearing their own transmissions (in contrast to the CSMA/CD protocol used in wired LANs). Therefore, an immediate positive acknowledgement scheme is employed to ascertain the successful reception of a frame. If the MACK (or CTS when a RTS is sent) is not received at the sender within the ACK-Timeout (CTS-Timeout), the data frame is presumed to have collided, and the emitter schedules a frame retransmission with a new random backoff. This Automatic Repeat reQuest (ARQ) retries continue, for each failing frame exchange sequence until the transmission is successful or until the Max-Retry-Limit is reached, whichever occurs first. There are two Max-Retry-Limit value defined in 802.11:

- Short-Retry-Limit (SRL)
- Long-Retry-Limit (LRL)

Every station maintains the current number of short retries as well as the number of the long retries, where these counters are incremented and reset independently of each other and both of them take an initial value of zero. The short retry counter is incremented whenever a control frame or a short packet frame (i.e.

smaller than RTS-Threshold) is retransmitted. Similarly, long retry counter is incremented whenever a long packet frame (i.e. larger than RTS-Threshold) is retransmitted. Retries for failed transmission attempts will continue until the short retry counter is equal to SRL or until the long retry counter is equal to LRL. When either of these limits is reached, the higher layer data packet is discarded from the sender MAC buffer and the CW will be reset to CW_{min}. On the other hand, whenever a MACK is received in response to the transmitted data frame, both short and long retry counters are reset to 0.

2.3 Transmission Control Protocol (TCP)

2.3.1 General Description

Most applications such as web browsing, E-mail, and FTP require reliable and in-order delivery of packets between two endpoints. However, the "best effort" packet delivery service provided by the IP does not give such guarantees to the end hosts. To combat the above problems and provide extra services for applications, Transmission Control Protocol (TCP) [12] has been designed and developed to run on top of unreliable IP. In other words, TCP builds reliable and in-order delivery of data between two end hosts over an unreliable and best effort IP service.

Although TCP was originally designed and optimized for wired networks, in order to enable seamless integration of ad hoc networks with the Internet, TCP also seems to be the natural choice for users of ad hoc networks that want to communicate reliably with each other and with the Internet. However, as it will be shown in the next chapter, such deployment does not come without cost and if not carefully used can severely harm the applications in ad hoc networks. To have a better understanding of the TCP functionality, the details of TCP protocol will be reviewed in this section.

2.3.2 Process-to-Process Reliable Data Delivery

As mentioned earlier, on top of the unreliable, connectionless IP service, TCP is a connection-oriented protocol. To achieve this goal, TCP needs to address the following two issues:

1. Connection Management

2. Data Loss Recovery

Although connection management refers to both connection establishment and closure, here we only briefly discuss the connection establishment procedure. More information on TCP connection management can be found in [12,34]. Connection establishment as shown in Figure 2.2 involves an exchange of synchronization messages, known as the three-way handshake before data packets can be exchanged between end hosts. Here first the client-side chooses a random

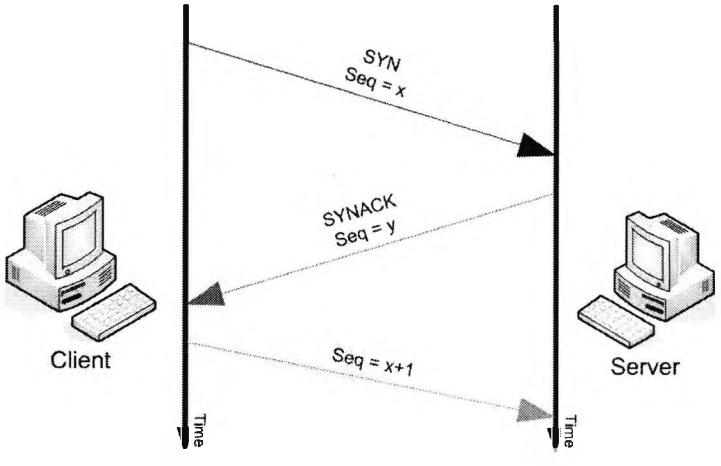


Figure 2.2: TCP connection establishment

number x for its Initial Sequence Number (ISN) and puts this number in the Sequence Number (Seq) field of the SYN segment. On the receipt of the SYN, the server host allocates the TCP buffers and variables to the connection, and sends a connection-granted segment (SYN-ACK) to client TCP including server ISN which is again a random number. Upon receiving the SYN-ACK, the client also allocates buffers and variables to the connection. The client host then sends the server yet another segment that acknowledges the server's connection-granted

segment. In nutshell, the main purpose of three-way-handshaking is to prevent old connection initializations and data packets, from causing confusion. In addition, the endpoints may exchange some extra parameter and option information during this phase, such as the Maximum Segment Size (MSS).

Since the TCP data loss detection and recovery procedure play an important role in TCP performance in multihop ad hoc networks, the rest of this section is dedicated to this issue.

2.3.3 Data Loss Detection and Recovery

TCP provides reliable data delivery by using cumulative acknowledgments (ACKs) that are sent by the receiver to the sender for all received data. A cumulative ACK from a receiver for k^{th} byte implies that all bytes prior to k have been successfully received, and that a segment beginning at byte k has not yet arrived. A direct advantage of the cumulative ACK scheme is that a later acknowledgment covers an earlier one, which gives some robustness against the loss of ACKs. A complement to reliable delivery is the loss recovery; that is the ability of the sender to deduce that certain packets did not reach the receiver and retransmit them, either without any explicit request from the receiver (timer-driven retransmission) or when explicitly requested by the receiver (data driven retransmission) as will be explained, respectively.

I- Timer Driven Retransmission (TCP Timeout)

In timer-driven recovery when the TCP sender does not receive a cumulative ACK for a segment within a certain interval (called Retransmission TimeOut or

RTO), it implicitly retransmits the missing data. In other words, the RTO is the amount of time the sender will wait for a given datagram to be acknowledged before a retransmission is triggered. The value of RTO is dynamically calculated using a Round Trip Time (RTT), which is the interval between the sending of a datagram with a certain sequence number, and the time that sequence number is acknowledged. In practice, instead of measuring every single value of RTT, TCP uses RTT samples for only one segment per window. Each RTT sample (denoted by RTT_{sample}) is used to update a smoothed RTT value (SRTT), which retains an exponentially weighted moving average of the history from recent samples. the new SRTT is computed from the formula in equation 2.3:

$$SRTT_{estimate} = (1 - \alpha) * SRTT_{old} + \alpha * RTT_{sample} \tag{2.3}$$

Where SRTT_{old} is the current estimate of the round-trip time, $\text{SRTT}_{estimate}$ is the expected RTT value for new segments and α is a constant between 0 and 1 that controls how rapidly the SRTT response to recent RTT samples (the recommended value for α is $\frac{1}{8}$ [35]). The RTO is then calculated as the expected smoothed RTT plus a variance factor:

$$RTO = SRTT_{new} + \max\{G, K * RTT_{VAR}\} \tag{2.4}$$

Where G is the TCP clock granularity (e.g. 500ms) and K is generally set to 4. The RTT_{VAR} is calculated as the RTT_{VAR} plus a difference between expected RTT ($\text{SRTT}_{estimate}$) and sample RTT (RTT_{sample}) as follows [36]:

$$RTTVAR_{new} = (1 - \beta) * RTTVAR_{old} + \beta * |SRTT_{estimate} - RTT_{sample}| \tag{2.5}$$

Here β determines the sensitivity of the RTT_{VAR} to the recent segment delay variations and is normally set to $\frac{1}{4}$. When a timeout occurs for a transmitted segment, the RTO is doubled (TCP exponential back-off) and the segment is retransmitted.

II- Data Driven Retransmission (TCP Fast Retransmit)

When a TCP receiver receives an errorless segment, it first checks the received sequence number against a (sliding) window of acceptable sequence numbers. To see how the sliding window works at receiver, let us consider figure 2.3; Here

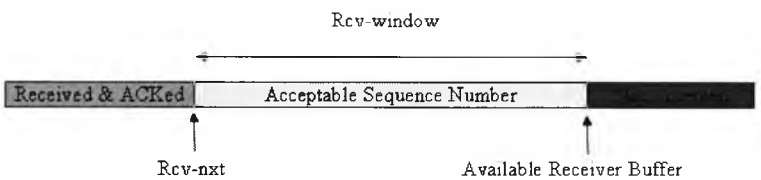


Figure 2.3: TCP sliding window

the left edge of the receiver window (Rcv-window) corresponds to the next expected byte (denoted by Rcv-nxt), which is the first among the packets required to complete the sequence of packets in the sequencing buffer. On the other hand, the right edge of the Rcv-window corresponds to the available buffer size at the receiver (for flow control purposes as would be explained shortly). A segment whose sequence number does not fall within the Rcv-window is discarded. Segments that their sequence number fall within the window but do not coincide with the left edge of the window are buffered. In both cases, the receiver sends an ACK asking for the current left edge of the window (Rcv-nxt). These ACKs are called duplicate ACKs and are mostly piggy backed on empty segments from

the receiver. The receipt of multiple duplicate acknowledgments explicitly stimulates the TCP sender to retransmit the segment that appears to be lost without waiting for the associated timeout event to expire. This early packet retransmission algorithm prior to RTO is therefore known as Fast Retransmit in the TCP standard.

2.3.4 Flow control

In addition to the reliable process-to-process data delivery, TCP provides a mechanism for the sender to recognize that if it is overwhelming the receiver's buffer by sending data too rapidly. This mechanism known as flow control enables the TCP sender to react to such an event by slowing down its transmission rate. Therefore, flow control scheme allows a slow receiver host to throttle the sending rate of a faster source host, in order to avoid buffer overflow and packet drops at the destination. TCP uses the sliding window explained before to provide flow control, whereby the TCP at the receiver side, indicates in each ACK the number of bytes it can accommodate in its receive buffer. This advertised value is called "receiver window" (rwnd) and effectively limits the number of bytes that the sender can have outstanding (i.e. unacknowledged) at any time.

2.3.5 Congestion Control

In a broad sense, congestion control is the ability of the sender to recognize that the network is overloaded and respond to this by reducing its transmission rate. It was added to TCP in 1987, after the Internet had suffered from what is called "congestion collapse" [37]. The congestion control was first standardized in

RFC2001 [38] and then updated in RFC2581 [39]. The goal of adding congestion control mechanism was to prevent congestion collapse by finding an appropriate rate of transmission for each connection dynamically. In order to dynamically control TCP's transfer rate, an additional window limit called "congestion window" (*cwnd*) was introduced which varies based on the network conditions. Then, the effective limit on outstanding data, called "send window" (swnd), is set as the minimum of the "receiver window" (rwnd) and the congestion window (*cwnd*).

$$swnd = \min\{cwnd, rwnd\} \tag{2.6}$$

In other words, TCP's effective window has a maximum value equal to the receive window, and the role of the congestion control mechanism is to find a window value between one Maximum Segment Size (MSS) and receiver window dynamically regarding the network conditions. To achieve this goal, TCP operates in the following two phases:

- Slow Start
- Congestion Avoidance

I- Slow Start

When a connection starts, resumes after a certain idle time[2] or a timeout occurs, slow start is performed. At the start of this phase, the congestion window is set to a small initial window size (IWS) which typically is one MSS [39]. Then the congestion window is increased by one MSS for each acknowledgment for the new

[2]Generally an idle time larger than RTO [39]

data that is received. This results in the window size doubling after each window worth of data is acknowledged[3].

The main goal of the slow start is to avoid a problem in the operation of the original TCP, where the sender at the start of a connection may transmit up to receiver window worth of data in one burst [38]. More precisely in the slow start phase, TCP increases the congestion window gradually toward the maximum value, rather than sending a full window worth of packets as soon as the connection is established. That is why it is called slow start despite the fact that the rate of increase during this phase is exponential to limit the performance loss while the connection operates at a small send window. After a certain threshold (called the slow start threshold or *ssthresh*) is reached, the connection moves into the congestion avoidance phase as will be explained in the next section. Note that the initial value of *ssthresh* can be arbitrary large and is generally set to the receiver window at the beginning of the connection [39]

II- Congestion Avoidance

In general, congestion avoidance consists of Additive Increase-Multiplicative Decrease (AIMD) that aims to make TCP to operate cautiously as the congestion window gets close to the value at which loss previously occurred. The TCP in its congestion avoidance phase probes the network for resources that might have become available by continuously increasing the window, albeit at a lower rate than in slow start. At the start of this phase, TCP gently probes the available bandwidth (Additive Increase). More precisely, in additive increase, if the congestion window is in units of packets, after each ACK is received the window is

[3]When the receiver implements delayed ACKs [40], the exponential rate of increase is reduced to 1.5

increased as:

$$cwnd = cwnd + \frac{1}{cwnd} \tag{2.7}$$

During this time if TCP detects the packet loss, it decreases the *cwnd* by a factor of two (Multiplicative Decrease) or it drops the *cwnd* to 1 MSS and goes to slow start depending on the type of TCP version used. It is important to note that in multiplicative decrease, the sender decreases its sending rate by half only once within one RTT regardless of the number of packet losses in that round. In addition, after loss detection, the *ssthresh* is set to half the value of *cwnd* at the time of loss.

2.3.6 TCP Versions

TCP congestion control mechanisms have evolved over time, as more was known about their behaviour and performance in the network, resulting in the known TCP versions. The most common TCP variants introduced and adopted during recent years include TCP Tahoe [12], Reno [35], NewReno [41], and Vegas [42]. In the following we briefly explain the key features of the above TCP variants, with special emphasis on TCP NewReno as recent traffic monitoring over the Internet, has confirmed the popularity of TCP NewReno [43]. In addition, since in the same study, the deployment of the TCP selective Acknowledgment (SACK) [44] has been shown to be very popular in Internet, throughout this thesis TCP NewReno with SACK option enabled has been used. This choice can be justified as such a combination is the likely candidate for adoption as the reliable transport protocol for use over ad hoc networks and readily implementable.

I- TCP Tahoe

In TCP Tahoe, when the congestion is detected through either TCP timeout or TCP receipt of three duplicate ACKs, the *cwnd* is reduced to 1 MSS and the sender enters slow start. Therefore, TCP Tahoe includes only Slow Start, Congestion Avoidance, and Fast Retransmit stages. Figure 2.4 shows the reaction of TCP Tahoe to packet loss.

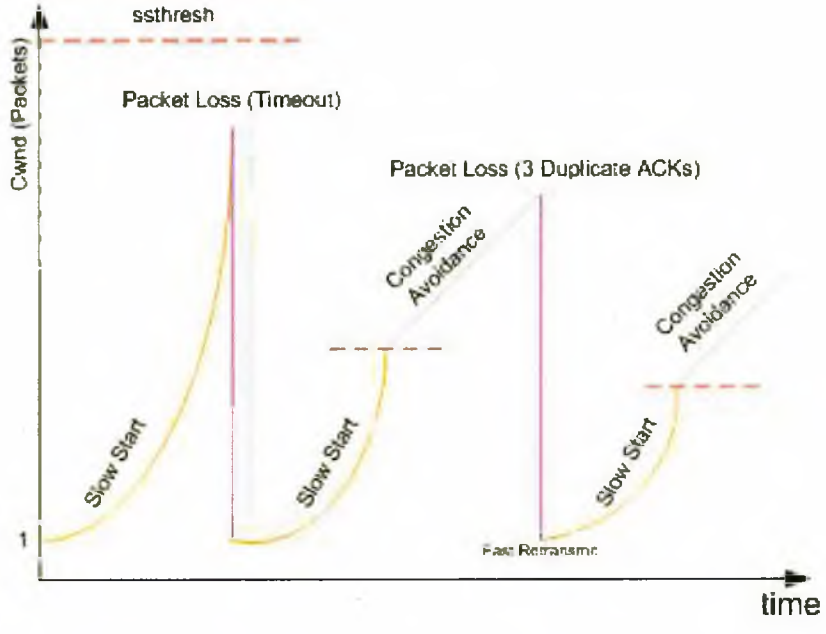


Figure 2.4: TCP Tahoe reaction to packet loss

II- TCP Reno

As was explained earlier in this chapter, when a time-out occurs, TCP interprets this as severe congestion in the network. On the other hand, when duplicate ACKs are received TCP interprets this as less severe congestion since it shows that the

receiver has received some out-of-order packets but has failed to receive the next expected segment. However, TCP Tahoe treats both type of the congestion in the same way and sets the congestion window to 1 MSS, and enters Slow Start. This can lead to bandwidth underutilization in the case of the fast retransmit, mostly triggered by the random packet loss or packet reordering. In order to avoid this, TCP Reno [35] introduces a new phase, called Fast Recovery, which is entered after a Fast Retransmit instead of slow start in TCP Tahoe. The fast recovery algorithm works as follows. After receiving 3 duplicate ACKs and sending what appears to be the missing segment by the fast retransmit at the sender, the *cwnd* is inflated to:

$$cwnd = ssthresh + 3MSS \tag{2.8}$$

Therefore, TCP stays in congestion avoidance in contrast to TCP Tahoe that enters Slow Start. Here the *ssthresh* is half of the *cwnd* at which the loss occurred. Afterward, for each additional duplicate ACK received, the window is increased by one packet. The congestion avoidance is left after the receipt of the first ACK, which acknowledges new data. When that happens, the window is deflated back to *ssthresh* and TCP proceeds in the congestion avoidance stage [39]. The basic idea behind Fast Recovery is that a duplicate ACK is an indication of available channel bandwidth since a segment has been successfully delivered. Thus, during Fast Recovery the TCP sender is able to make intelligent estimates of the amount of outstanding data. Figure 2.5 shows the stages in TCP Reno that include slow-start, congestion avoidance, and fast retransmit which were carried over from the TCP Tahoe, and additionally the fast recovery algorithm.

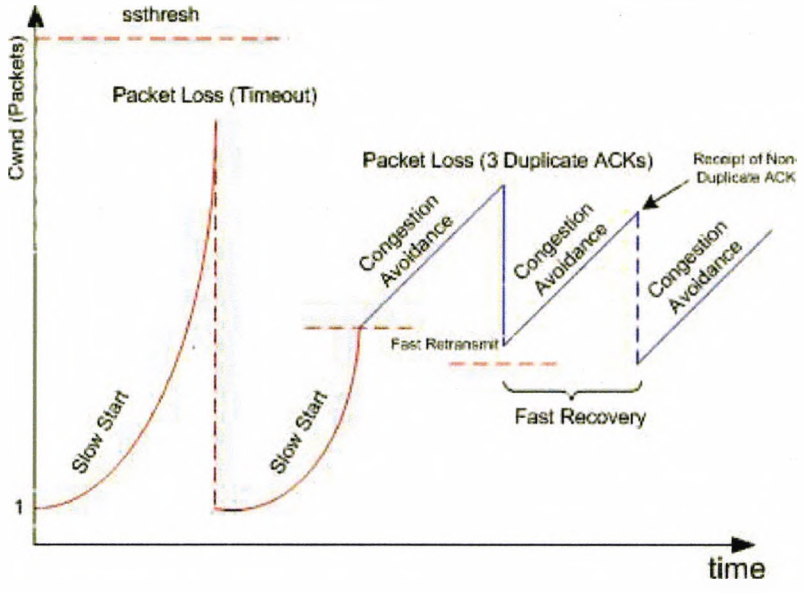


Figure 2.5: TCP Reno reaction to packet loss

III- TCP NewReno

The NewReno algorithm [41] is functionally very similar to TCP Reno. The difference between the two variants can be distilled to the treatment of a loss event during the congestion avoidance phase. TCP Reno performance significantly suffers when multiple packets are dropped from a window of data. This is because the only way Reno can recover further lost packets without timing out is through performing a Fast Retransmit for each. However, if the TCP sender doesn't receive three duplicate ACKs after a loss (for example, because the congestion window is less than 4 segments), then the TCP sender has to wait for a retransmit timer to expire. However, retransmit timeouts have the cost of introducing a possibly considerable delay of waiting for the transmit timer to expire. This

delay can be particularly long and severe in a low bandwidth network, as the TCP sender may not receive sufficient samples to estimate an effective upper bound on the RTT. For this reason, the performance of Reno can be worse than Tahoe's in small RTT situations where the window size is not big enough to trigger fast retransmit and TCP has to wait for a timeout retransmission. To address this issue, TCP New-Reno includes a small change to the Reno algorithm at the sender that eliminates Reno's wait for a retransmit timer timeout when multiple packets are lost from a window. The change concerns the sender's behaviour during Fast Recovery when a partial ACK is received [4]. In Reno, partial ACKs take TCP out of Fast Recovery by "deflating" the usable window back to the size of the congestion window. In New-Reno, partial ACKs do not take TCP out of Fast Recovery. Instead, partial ACKs received during Fast Recovery are treated as an indication that the packet immediately following the acknowledged packet in the sequence space has been lost, and should be retransmitted. Thus, when multiple packets are lost from a single window of data, New-Reno can recover without a retransmission timeout, by retransmitting one lost packet per round-trip time until all of the lost packets from that window have been retransmitted. New-Reno then remains in Fast Recovery until all of the data outstanding when Fast Recovery was initiated have been acknowledged.

A descriptive summary of TCP Tahoe, Reno and New Reno versions is shown in Figure 2.6

[4]Partial ACK is the ACK that does not acknowledge the highest sequence number sent at the point when Fast Recovery was initiated. It is thus an indication that not all data sent before entering Fast Recovery has been received

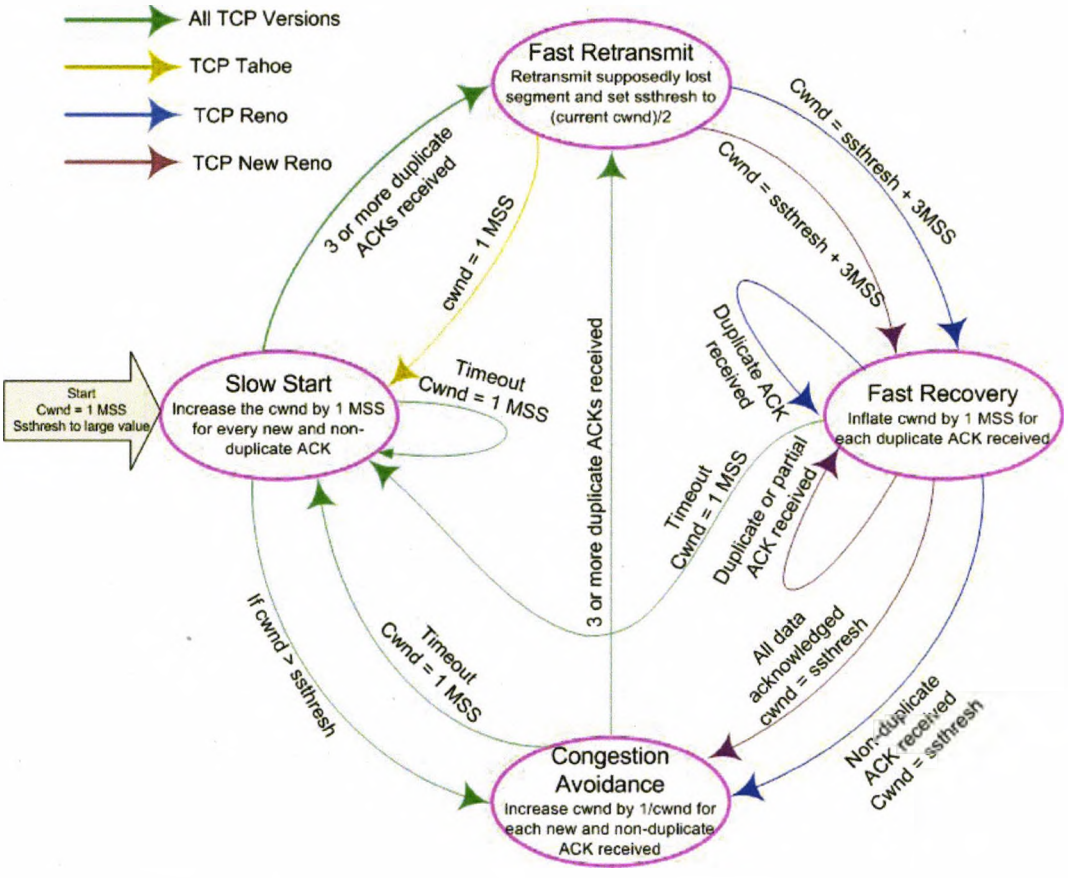


Figure 2.6: Descriptive summary of TCP Tahoe, Reno and New Reno

IV- TCP SACK

TCP SACK [44] is a Reno-based TCP variant which makes use of the facilities provided by the Selective Acknowledgements (SACK) option of TCP [45]. The SACK-enabled segments provide the TCP sender with some indication of the status of the destination's receiving buffer. To achieve this, the data receiver generates SACK information for every ACK response it produces that does not cover the highest sequence number in the data receiver's queue. Hence, when reception of a non-contiguous segment occurs, instead of returning a duplicate

ACK, the receiver produces a reply which contains further information in the header of the segment in the form of an option. The information embedded in the SACK response contains a list (in the form of block pairs) of some of the isolated data blocks in the receiver's buffer, which have not been passed on to the application layer, as additional data segments are required to fill the gaps in the receiving sequence within the receiver's window. Hence, in the event of packet loss the sender can re-send only the exact packets that have been lost in transit and avoid producing unnecessary retransmissions.

V- TCP Vegas

In contrast to all the above TCP versions that need to experience losses in order to find the available bandwidth of the connection (reactive approach), TCP Vegas, which was introduced in 1994, tries to predict congestion and reduce the congestion window accordingly (proactive approach) [42]. More specifically, TCP Vegas extends New Reno's mechanisms by using two techniques. In the first technique, TCP Vegas reads and records the system clock each time a segment is sent. When an ACK arrives, Vegas reads the clock again and does the RTT calculation using this time and the timestamp recorded for the relevant segment. Then it uses this more accurate RTT estimate to decide whether to retransmit the packet. The second technique gives TCP the ability to anticipate congestion, and adjust its transmission rate accordingly. TCP Vegas does this by monitoring the difference between the throughput it is expecting to see (Expected Throughput, $T_{expected}$) and the throughput it is actually realizing (Actual Throughput,

T_{actual}). In practice, $T_{expected}$ is calculated as follow:

$$T_{expected} = \frac{\text{Congestion window size}}{\text{smallest measured RTT}} \tag{2.9}$$

In the place when TCP Vegas is started in Slow Start, the window is increased exponentially every other RTT. In between each consecutive increment, the window remains fixed, and the achieved throughput is compared to the expected throughput. If ($T_{actual} < T_{expected}$), the Congestion Avoidance phase is entered. The Congestion Avoidance algorithm is based on the calculation of the "Diff" that is considered as the amount of extra data:

$$Diff = T_{expected} - T_{actual} \tag{2.10}$$

The goal of Vegas is to maintain the right amount of extra data in the network. To reach this aim, it defines two thresholds α and β . Then based on equation 2.11, the value of *cwnd* during next RTT is determined:

$$cwnd = \begin{cases} cwnd + 1 & if \ \ diff < \alpha \\ cwnd & if \ \ \beta < diff < \alpha \\ cwnd - 1 & if \ \ diff > \beta \end{cases} \tag{2.11}$$

This means that when the actual throughput gets far from the expected throughput, the more congestion there is in the network. On the other hand, when the actual throughput rate gets too close to the expected throughput, the connection is in danger of not utilizing the available bandwidth. Note that all these calculations are done once per RTT. In addition, TCP Vegas also make some minor modifications to some parameters, such as reducing the congestion window by

1/4 instead of 1/2 after a Fast Retransmit, or starting with a window of 2 MSS even after a retransmit timeout rather than 1 MSS.

2.4 Dynamic Source Routing

2.4.1 General Description

In general, there are two main routing approaches in ad hoc networks, namely the reactive and proactive routing paradigms. Proactive protocols (such as Optimized Link State Routing [46]), involve attempting to maintain routes between nodes in the network at all times, including when the routes are not currently being used [47]. They are traditionally classified as either distance-vector or link-state protocols and are based on exchanging periodic information regardless of outside events. In this approach, updates to the individual links within the networks are propagated to all nodes, or a relevant subset of nodes in the network, such that all nodes in the network eventually share a consistent view of the state of the network. The advantage of a proactive protocol is that there is little or no latency involved when a node wishes to begin communicating with an arbitrary node that it has not yet been in communication with. The disadvantage is that the control message overhead of maintaining all routes within the network can rapidly overwhelm the capacity of the network in very large networks, or situations of high mobility.

Reactive protocols (such as Dynamic Source Routing Protocol [48]), also known as on-demand protocols, involve searching for routes to other nodes only as they are needed [47]. More specifically, a route discovery process is invoked when a node wishes to communicate with another node for which it has no route table entry. When a route is discovered, it is maintained only for as long as it is needed by a route maintenance process and inactive routes are removed at regular intervals. Reactive protocols require less control traffic to maintain routes

that are not in use than in proactive methods. On the other hand, the main drawback of these methods is that the routes are often unavailable at the time an application first needs them. This means that applications in networks using one of these protocols often experience an initial delay during the time it takes to establish a route between the communication endpoints.

There exists another class of ad-hoc routing protocols, called hybrid protocols which employ a combination of proactive and reactive methods. The main idea behind these protocols is to maintain groups of nodes in which routing between members within a zone (cluster) is via proactive methods, and routing between different groups of nodes is via reactive methods.

In this study, Dynamic Source Routing protocol (DSR) [48] has been chosen as the routing protocol for several reasons. Firstly, DSR is very efficient in finding (learning) routes in terms of the number of control packets used, and does not use periodic control messages [49–51]. Secondly, as DSR is a source initiated routing protocol (in contrast to hop by hop routing), intermediate nodes do not need to get involved in route discovery and maintain up-to-date routing information in order to route the packets. This is because every packet carries a source route description of a path through the network. Therefore, with a cost of no additional packets, every node overhearing a source route learns a way to reach all nodes listed in the route. This form of active learning is reduces the overhead in the network [49, 52]. Finally, DSR does not use any periodic routing advertisement, link status sensing, or neighbour detection packets, and does not rely on these functions from any underlying protocols in the network. [51, 52]

The DSR protocol consists of two basic mechanisms: Route Discovery and Route Maintenance which will be explained briefly in the following section.

2.4.2 Route Discovery

In DSR, route discovery is performed only when the sender attempts to send a packet to the destination and does not already know a route. The routes that DSR discovers and uses are called source routes. That is, the sender learns the complete and ordered sequence of network hops necessary to reach the destination. Each packet sent from the sender carries this list of hops in its header. As a result, the packet size in DSR depends on the distance between the communicating nodes.

To find a route to the destination, DSR uses Route REQuest (RREQ) and Route REPly (RREP). More specifically, to establish a route, the sender transmits a RREQ message as a single local broadcast packet, which is received by all nodes currently within wireless transmission range of the sender. Each RREQ message identifies the initiator, the final destination, and also contains a unique Request-ID, determined by the initiator of the RREQ. Each RREQ also contains a route record that lists the address of each intermediate node through which this particular copy of the RREQ message has been forwarded. This route record is initialized to an empty list by the initiator of the route discovery. As the RREQ propagates, each host adds its own address to a record list in the RREQ packet, before broadcasting the RREQ on to its neighbours. Then at each node, RREQ message is forwarded only if all of the following conditions are met:

1. The node is not the target (destination) of the RREQ packet
2. The node is not listed in source route
3. The node has not received another RREQ with the same sender and Request-ID number

4. No route information to the target node is available in its route cache.

If all are satisfied, then the relaying node also appends its IP address to the source route and broadcasts the packet to its neighbours . On the other hand, if the intermediate node finds a valid route to destination in its cache, it produces RREP message and sends that back towards sender.

In a nutshell, when a mobile node has a packet to send, it first consults its route cache to determine whether it has a valid route to the destination or not. If it has one, then it starts sending data. Otherwise, it initiates route discovery by sending RREQ packet. Each node receiving this packet checks whether it has an unexpired route to mentioned destination or not. If it does not have one, it first checks for its own address in the route record of the packet. If it does not find that as well, it adds its own address to route record and forwards the packet; otherwise, it will simply discard the packet.

2.4.3 Route maintenance

As the name indicates, the route maintenance is responsible for maintaining and monitoring the already established routes. Route Maintenance is again an on-demand operation and is only used when sender is actually sending packets to the ultimate destination. In other words, in DSR each node transmitting a packet is responsible for confirming that the packet has been received by the next hop along the source route. This confirmation of receipt to the next hop in the case of using 802.11 MAC is provided at no cost to DSR (due to the link-level acknowledgement frame). If the packet is retransmitted until the maximum number of times (defined in MAC protocol) and no receipt confirmation is received, that

node returns a route error message to the original sender of the packet, identifying the link over which the packet could not be forwarded. When the route error packet is received by the sender, the sender attempts to use any other route it happens to know to the destination (if it had received multiple RREPs), or it invokes Route Discovery (after performing exponential backoff) again to find a new route.

2.5 Summary

This chapter reviewed the details of TCP (in the transport layer), DSR (in the network layer), and IEEE 802.11 MAC (in the link layer) protocols that are referred in the thesis and are commonly used in ad hoc networks.

Chapter 3

Analysis of IEEE 802.11 DCF Operation

3.1 Introduction

It is well known that the performance of 802.11 DCF can have a significant impact on the achievable throughput, delay, and scalability of the ad hoc network [53–56]. In particular, as shown in [56], the parameters of 802.11 can have a critical impact on upper layer performance such as TCP in ad hoc networks. To have a better understanding of the operation of the 802.11 DCF, this chapter presents an analytical model of 802.11 to evaluate the impact of different parameters on the achievable throughput of 802.11 ad hoc networks. To this aim, section 3.2 first reviews some of the related work in modelling the operation of 802.11. Section 3.3 presents details of the proposed model, including assumptions and model components. In section 3.4, the analytical throughput of the 802.11 is derived from the 3-D Markov chain model developed in section3.3. Section 3.5 validates

the accuracy of this model by simulations and comparison with other analytical models. Finally, in section 3.6, the impact of different 802.11 parameters on maximum throughput theoretically achievable are investigated and discussed in detail.

3.2 Related Work

Since the introduction of the IEEE 802.11, a considerable amount of work has been done on the performance evaluation of the protocol in ad hoc networks and possible ways to enhance its performance especially from a throughput point of view. In particular, one of the earliest analyses of the throughput of DCF was carried out in [57] using a simplified geometrically distributed backoff model. A more realistic model was proposed by Bianchi [58] where the evolution of the back-off stage at each node is described by a Markov process; The key importance of Bianchi model is to use a Markov chain to capture the effect of the contention window and binary slotted exponential back-off procedure used by DCF in 802.11. However, his model did not take into account the retry limits and the effect of timeout values in the throughput calculation. Since then, many improvements to the initial Markov chain have been proposed to capture different aspects of the operation of 802.11 in ad hoc networks. For instance, the study in [59] further improved the model by considering the retry limit in the Markov model. However, it still did not address the issue of multiple retry limits as specified in the 802.11 standard. The authors in [55] further improved the model by evaluating the analytical performance of 802.11 in ad hoc networks under unsaturated

traffic conditions. Finally, the work by [60] takes into account the impact of hidden terminals in the analytical model. Despite the analytical improvement, the main shortcoming of the previous studies is a lack of comprehensive and accurate analysis of the impact of different 802.11 MAC parameters (e.g. maximum short and long retry limits, minimum contention window) on the achievable link layer throughput. To accurately model the performance of the 802.11 and to be able to take into account different 802.11 parameters, in the next section we extend the Bianchi two dimensional model into a three dimensional model by considering different retry limits.

3.3 Model Assumptions and Components

In order to analyze the protocol, the following assumptions are made:

- The network consists of finite number of n contending stations.
- Every station always has a packet to transmit after the completion of each successful transmission. In other words, the system works in throughput saturation that is defined as the limit reached by the system throughput as the offered load increases.
- The probability of a packet collision is constant and independent of the number of retransmissions attempts of this frame.

To describe the behaviour of 802.11, let us first denote the state space F as:

$$F = \{(i, j, k) : 0 \leq i \leq L, 0 \leq j \leq B \leq S, k \geq 0\} \tag{3.1}$$

where i is the current number of long retries, j is the current number of the backoff state, and k is the current backoff counter value which can take any value from 0 to W_j, where

$$W_j = \begin{cases} W_0 * 2^j & \text{j} \leq \text{B} \\ W_0 * 2^B & \text{j} > \text{B} \end{cases} \tag{3.2}$$

Also, in our space, L is the maximum number of long retry limits, S is the maximum number of short retry limits and B is the maximum number of backoff stages in the IEEE 802.11 protocol[1].

Figure 3.1 demonstrates the schematic of the proposed 802.11 three dimensional Markov chain.

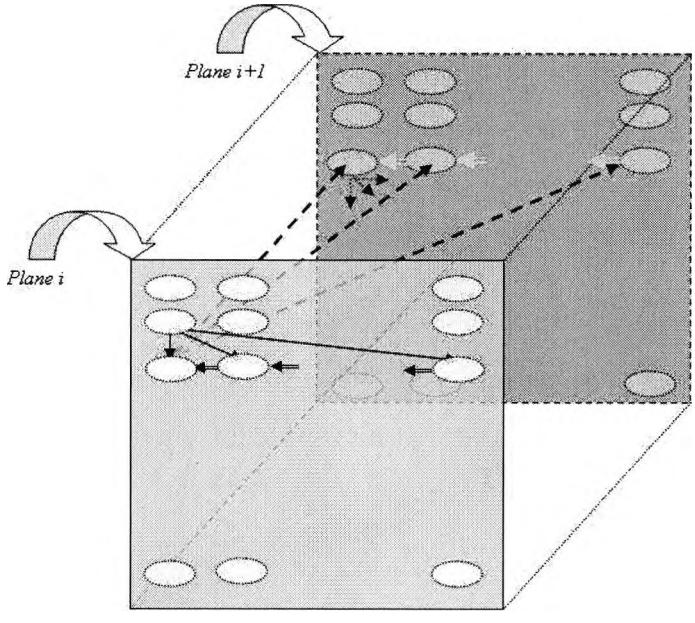


Figure 3.1: Three Dimensional Markov chain model

To better understand the behavior of the proposed Markov chain in figure 3.1, it is important to note that one of the main features of this model which

[1]In default operation of 802.11, we have L=4, S=7, and B=5

has been mostly ignored previously, is the distinction made between packets that require RTS/CTS exchange prior to their transmission and other packets (including 802.11 control packets) in state transitions. We believe such a distinction should be accommodated in the model since the probability of packet transmission and collision for packets that do not use RTS/CTS is clearly different from packets that perform channel reservation prior to their transmission. To this aim, in figure 3.1 if a control packet or data packets smaller than the RTS-Threshold collides, the new state is chosen from the next row of the current plane (i.e. a transition along y axis). However, if a data packet larger than RTS-Threshold is dropped, the new state is chosen from the next row of the next plane (i.e. a transition along z axis). In other words, if a collision results to an increase in the number of short retries, the new state is chosen in the same plane. However, if the packet drop results in the increase in the number of long retries, a new state is picked up from the next plane. The rules of these transitions will be covered in more detail later in this section.[2]

Now let $b_{i,j,k}$ be the stationary distribution of the Markov chain stochastic process for a given station. Figure 3.2 depicts in more detail the states of the first plane (i=0) in the proposed Markov chain. As it can be seen, with the total probability of P_{col}^{RTS}, the station chooses a new backoff counter from the next row. However, in case of a successful RTS/CTS transmission (with the probability of $(1 - P_{col}^{RTS})$), if the station successfully transmits its DATA packet with the probability of P_{suc}^{Data} the chain is reset to initial stage; alternatively if with the probability of P_{col}^{Data} the DATA packet is dropped, the station chooses its new backoff value from the plane $i = 1$ and the process continues. Also, it

[2]For simplicity and sake of argument, in the rest of this chapter we assume the RTS/CTS exchange is performed for all higher layer data packets and refer to such packets as DATA packets to distinguish them from 802.11 control packets.

is worth mentioning that in the first plane, there are total number of S retries (i.e. short retry limits) before a station drops the DATA packet. In addition, according to equation (3.2), the number of backoff states (W_j) will not increase after a station reaches stage B (maximum backoff stages) and remains fixed.

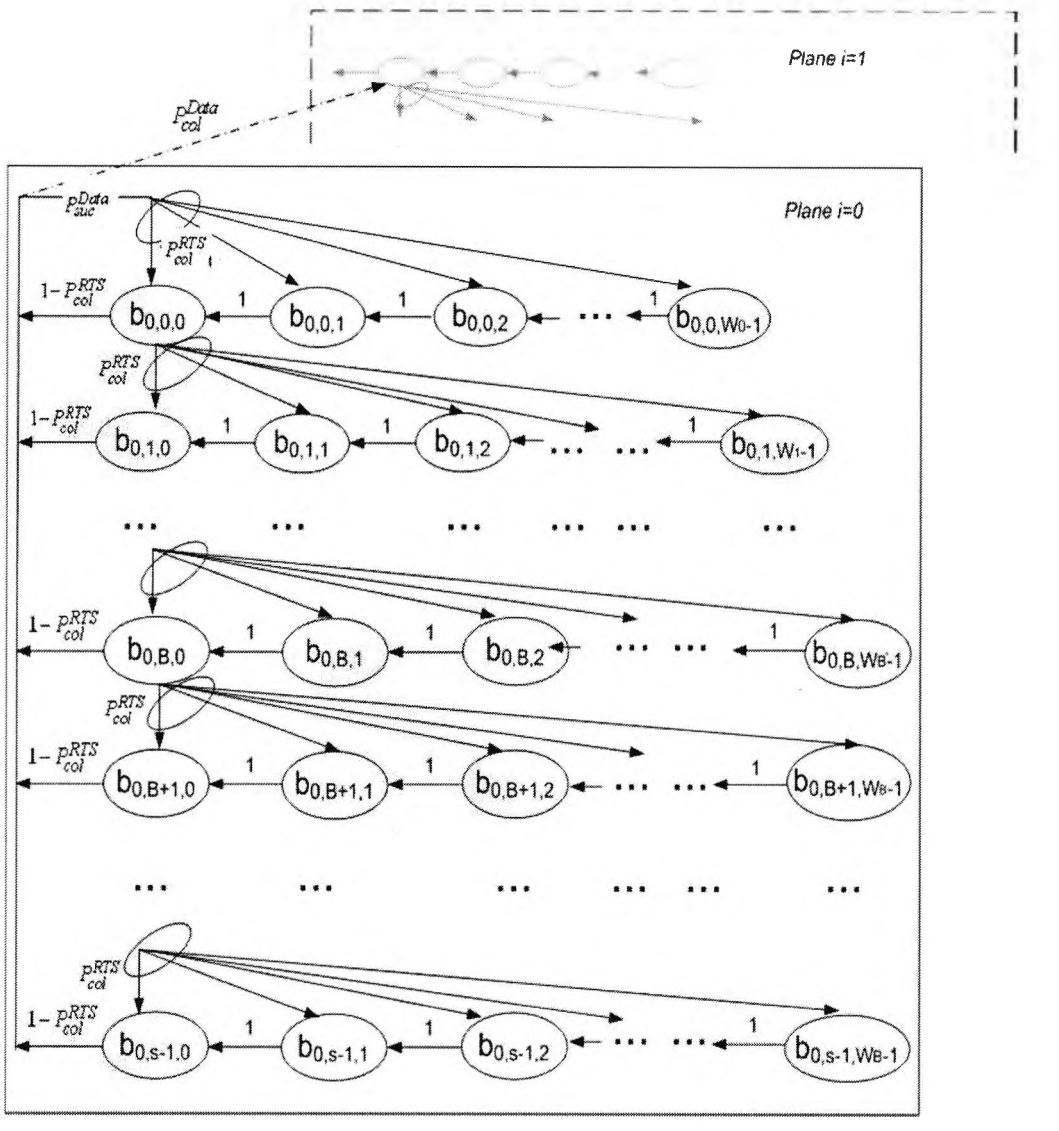


Figure 3.2: The states of a Markov chain in plane i=0

Although the structure of the other planes are similar to figure 3.2, it is important to note that different planes do not share exactly the same number of states. For instance, apart from the first plane (i=0), the minimum value of the current number of backoff stage (j) should be greater than or equal to 1. In other words, the first row of the plane $i = 0$ cannot exist in the following planes. This is because when the station is in plane $i \neq 0$, it means there has been at least one unsuccessful transmission prior to the transition. Therefore, $j \neq 0$.
To obtain a better understanding of the differences between states across multiple planes, figure 3.3 depicts the first state of the first and last row across multiple planes.

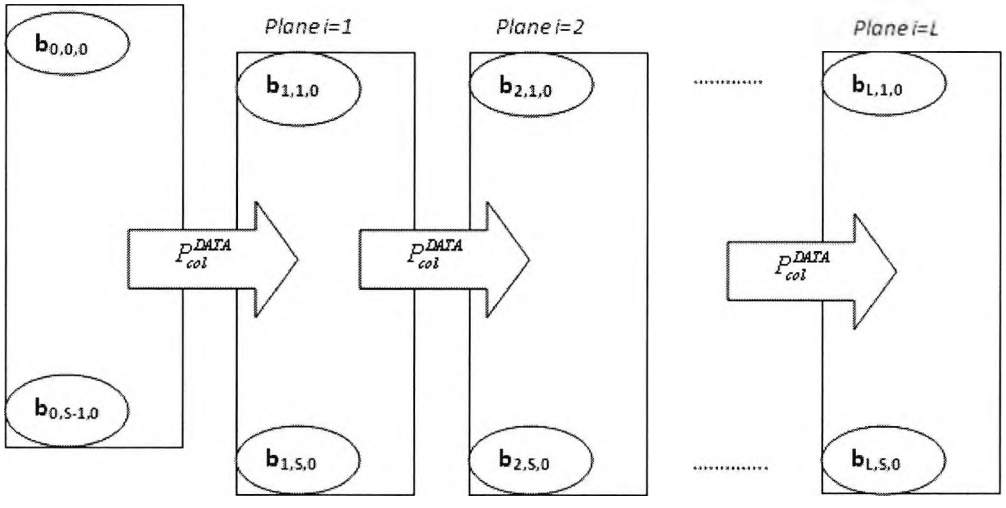


Figure 3.3: The key states of the Markov chain plane's transition

Throughout the rest of this section, the main objective is to write down all the states across different planes in terms of $b_{0,0,0}$. To be able to do so, let us first note that the state transition diagram shown in figure 3.2 is governed by transition probabilities given in equations 3.3 to 3.6. Also, along with each transition probability, its transition duration is specified to enable us to calculate

the average time that a station stays in one state.

- The backoff counter decrements and the station makes a transition from state (i,j,k) to state (i,j,k-1):

$$\begin{cases} P\{(i,j,k-1)|(i,j,k)\} = 1 \\ t\{(i,j,k-1)|(i,j,k)\} = \sigma \end{cases} \tag{3.3}$$

  where σ represents the system time slot.

- The station sends a RTS packet in state (i,j,0), but its RTS packet collides and the station reaches state (i,j+1,k).

$$\begin{cases} P\{(i,j+1,k)|(i,j,0)\} = \frac{P_{col}^{RTS}}{W_{j+1}} \\ t\{(i,j+1,k)|(i,j,0)\} = T_{col}^{RTS} \end{cases} \tag{3.4}$$

where P_{col}^{RTS} and T_{col}^{RTS} refer to the probability and average time of RTS collision, respectively.

In addition, the state transition diagram between different planes shown in figure 3.3 is governed by the following transition probabilities and durations:

- The station sends a RTS/CTS packet successfully in state (i,j,0), but the actual DATA is dropped (due to channel error, hidden terminal, etc.) and the station reaches state (i+1,j+1,k).

$$\begin{cases} P\{(i+1,j+1,k)|(i,j,0)\} = \frac{P_{suc}^{RTS} * P_{col}^{Data}}{W_{j+1}} \\ t\{(i+1,j+1,k)|(i,j,0)\} = T_{col}^{Data} \end{cases} \tag{3.5}$$

here P_{suc}^{RTS} and P_{col}^{Data} are the probability of RTS success and DATA packet collision, respectively. Also, T_{col}^{Data} refers to the average time the channel is occupied when a DATA packet collision occurs.

- The station sends a DATA packet (following a successful RTS/CTS exchange) successfully in state (i,j,0), and therefore reaches state (0,0,k).

$$\begin{cases} P\{(0,0,k)|(i,j,0)\} = \frac{P_{suc}^{RTS} * P_{suc}^{Data}}{W_{j+1}} \\ t\{(0,0,k)|(i,j,0)\} = T_{suc}^{Data} \end{cases} \tag{3.6}$$

where P_{suc}^{Data} and T_{suc}^{Data} refer to the probability and average time of DATA success event.

Considering the regularity of the Markov chain shown in figures 3.2 and 3.3, we have the following relations for each $k \in (0, W_j - 1)$:

$$\begin{aligned} b_{i,j,k} &= b_{i,j,0} * \frac{W_j - k}{W_j} \\ b_{0,j,0} &= b_{0,0,0} * (P_{coll}^{RTS})^j \qquad for(0 \leq j \leq S-1) \\ b_{i,j,0} &= b_{i,1,0} * (P_{coll}^{RTS})^{j-1} \qquad for(i \neq 0 \ \ and \ \ 1 \leq j \leq S) \\ b_{i,1,0} &= b_{0,0,0} * [P_{suc}^{RTS} * P_{col}^{Data})]^i \qquad for(i \neq 0) \end{aligned} \tag{3.7}$$

Using the set of equations in (??) and the normalization condition, we get:

$$
\begin{aligned}
1 &= \sum_{i=0}^{L}\sum_{j=0}^{S}\sum_{k=0}^{W_J-1} b_{i,j,k} \\
&= \sum_{j=0}^{S-1}\sum_{k=0}^{W_j-1} b_{0,j,k} + \sum_{i=1}^{L}\sum_{j=1}^{S-i+1}\sum_{k=0}^{W_j-1} b_{i,j,k} \\
&= \sum_{j=0}^{S-1}\sum_{k=0}^{W_J-1} (b_{0,j,0} * \frac{W_j-k}{W_j}) + \sum_{i=1}^{L}\sum_{j=1}^{S}\sum_{k=0}^{W_J-1} (b_{i,j,0} * \frac{W_j-k}{W_j}) \qquad (3.8) \\
&= b_{0,0,0} * \\
&\left[\sum_{j=0}^{S-1}\left((P_{col}^{RTS})^j * \frac{W_j+1}{2}\right) + \sum_{i=1}^{L}\sum_{j=1}^{S}\left(\left[(1-P_{col}^{RTS})\,(P_{col}^{Data})\right]^i \left[P_{col}^{RTS}\right]^{j-1} * \frac{W_j+1}{2}\right)\right]
\end{aligned}
$$

Although in the above equation, $b_{0,0,0}$ depends on both P_{col}^{Data} and P_{col}^{RTS}, the value of P_{col}^{Data} can statistically be estimated by each station. This is because $P_{col}^{Data} = 1 - \left(P_{suc}^{Data} | P_{suc}^{RTS}\right)$ and the value of $\left(P_{suc}^{Data} | P_{suc}^{RTS}\right)$ can be easily estimated by each station since each station may keep track of two parameters. First the number of occasions that it has transmitted a DATA packet (after a successful RTS/CTS reservation) and has received a MACK for that packet. Secondly the total number of transmitted DATA packets. Based on the above values, the node at each time can statistically calculate the probability of $\left(P_{suc}^{Data} | P_{suc}^{RTS}\right)$ as the ratio of the first to second parameter. Therefore, this implies that in equation (3.8), $b_{0,0,0}$ only depends on the value of P_{col}^{RTS}.

To find the value of P_{col}^{RTS}, it is sufficient to note that the probability that a transmitted RTS packet encounters a collision; This is the probability that in a time slot, at least one of the remaining contending stations (including the receiver

itself) transmits an RTS packet. Therefore, we have:

$$P_{col}^{RTS} = 1 - \left(n\tau(1-\tau)^{n-1} + (1-\tau)^{n}\right) \tag{3.9}$$

where τ is the probability of RTS transmission in a randomly chosen slot time. On the other hand, as RTS transmission occurs when the backoff counter reaches 0, we have:

$$\begin{aligned}\tau &= \sum_{i=0}^{L}\sum_{j=0}^{S} b_{i,j,0} = \sum_{j=0}^{S-1} b_{0,j,0} + \sum_{i=1}^{L}\sum_{j=1}^{S} b_{i,j,0} \\ &= b_{0,0,0}\left[\frac{1-\left(P_{col}^{RTS}\right)^{S}}{1-P_{col}^{RTS}} + \sum_{i=1}^{L}\sum_{j=1}^{S}\left(\left[\left(1-P_{col}^{RTS}\right)\left(P_{col}^{Data}\right)\right]^{i}\left[P_{col}^{RTS}\right]^{j-1}\right)\right]\end{aligned}$$

Therefore, equations (3.8), (3.9), and (??) represent a nonlinear system with two unknowns τ and P_{col}^{RTS} which can be obtained by numerical results in terms of different n and P_{col}^{Data}. Table 3.1 presents the value of τ (shown as τ_{def} or default τ) and its corresponding P_{col}^{RTS} for different values of n and P_{col}^{Data}.

Table 3.1: Default τ and its corresponding P_{col}^{RTS} under different n and P_{col}^{Data}

	$P_{col}^{Data} = 0.001$	$P_{col}^{Data} = 0.005$	$P_{col}^{Data} = 0.01$	$P_{col}^{Data} = 0.05$
$n = 5$	$P_{col}^{RTS} = 0.0306$ $\tau_{def} = 0.0587$	$P_{col}^{RTS} = 0.0304$ $\tau_{def} = 0.0585$	$P_{col}^{RTS} = 0.0301$ $\tau_{def} = 0.0582$	$P_{col}^{RTS} = 0.0282$ $\tau_{def} = 0.0562$
$n = 10$	$P_{col}^{RTS} = 0.0987$ $\tau_{def} = 0.0541$	$P_{col}^{RTS} = 0.0982$ $\tau_{def} = 0.0539$	$P_{col}^{RTS} = 0.0976$ $\tau_{def} = 0.0537$	$P_{col}^{RTS} = 0.0930$ $\tau_{def} = 0.0522$
$n = 20$	$P_{col}^{RTS} = 0.2197$ $\tau_{def} = 0.0441$	$P_{col}^{RTS} = 0.2192$ $\tau_{def} = 0.0440$	$P_{col}^{RTS} = 0.2184$ $\tau_{def} = 0.0439$	$P_{col}^{RTS} = 0.2126$ $\tau_{def} = 0.0431$
$n = 40$	$P_{col}^{RTS} = 0.3531$ $\tau_{def} = 0.0310$	$P_{col}^{RTS} = 0.3527$ $\tau_{def} = 0.0309$	$P_{col}^{RTS} = 0.3522$ $\tau_{def} = 0.0309$	$P_{col}^{RTS} = 0.3478$ $\tau_{def} = 0.0306$

Considering the results from table 3.1, it is interesting to note that the dependency of τ on P_{col}^{Data} is very marginal and almost negligible.

3.4 Throughput Analysis

Having derived the analytical model of 802.11, it is now straightforward to calculate the 802.11 analytical throughput. To this aim, let S be the normalized throughput, defined as the fraction of time the channel is used to successfully transmit DATA payload.[3].

$$S = \frac{\text{E[Data payload information in a slot time]}}{\text{E[Length of a slot time]}} \tag{3.10}$$

Assuming the average DATA payload size as E[P], the average amount of DATA payload information successfully transmitted in a slot time is $P_{tr}^{Data} P_{suc}^{Data} E[P]$. On the other hand, the average length of a slot time can be obtained by considering that a single slot time falls in one of the following four cases:

1. With probability $1 - (P_{suc}^{RTS} + P_{col}^{RTS})$, the slot time is empty
2. With probability P_{col}^{RTS} the time slot contains a unsuccessful RTS transmission
3. With probability $P_{tr}^{Data} P_{col}^{Data}$ the time slot contains a unsuccessful DATA transmission
4. With probability $P_{tr}^{Data} P_{suc}^{Data}$ the time slot contains a successful DATA transmission

[3]As explained earlier, in this chapter DATA payload refers to upper layer packets and does not include the control packets in link layer in contrary to [58] throughput model

Therefore, equation 3.10 can be rewritten as:

$$S = \frac{P_{tr}^{Data} P_{suc}^{Data} E[P]}{(1 - (P_{suc}^{RTS} + P_{col}^{RTS}))\,\sigma + (P_{tr}^{Data} P_{suc}^{Data})\,T_{suc}^{Data} + (P_{col}^{RTS})\,T_{col}^{RTS} + (P_{tr}^{Data} P_{col}^{Data})\,T_{col}^{Data}}$$

where σ is the slot duration.

Note that in equation (??), the terms P_{suc}^{Data} and P_{col}^{Data} indeed refer to $(P_{suc}^{Data}|P_{suc}^{RTS})$ and $(P_{col}^{Data}|P_{suc}^{RTS})$, respectively which as explained earlier can be statistically computed by each node.

In order to calculate the probability of DATA packet transmission (P_{tr}^{Data}), let us remember that DATA packets would be sent after a successful RTS/CTS handshake. In addition, regarding 802.11 MAC timing specification, the DCF protocol ensures that CTS frame transmission will be successfully received at its *destination* (the one who sent the RTS), if the CTS frame has been issued in response to the RTS. Therefore:

$$P_{suc}^{RTS} = P_{tr}^{RTS} * \left(\frac{n\tau(1-\tau)^{n-1}}{1-(1-\tau)^n}\right) = n\tau(1-\tau)^{n-1} \tag{3.11}$$

Also, we have:

$$P_{tr}^{Data} = P_{Suc}^{RTS} = n\tau(1-\tau)^{n-1} \tag{3.12}$$

Finally, using the timing specifications of 802.11 MAC [11] the corresponding values of transition times can be calculated as follows:

$$
\begin{aligned}
T_{suc}^{Data} &= DIFS + T_{rts} + SIFS + \delta + T_{cts} + SIFS + \delta + (H + E[P]) \\
&\quad + SIFS + \delta + T_{ack} + \delta \\
T_{col}^{RTS} &= DIFS + T_{rts} + SIFS + T_{cts\_timeout} \\
T_{col}^{Data} &= DIFS + T_{rts} + SIFS + \delta + T_{cts} + SIFS + T_{ack\_timeout}
\end{aligned}
$$

Here, T_{rts}, T_{cts}, and T_{ack} represent the time required to transmit RTS, CTS, and MACK over the channel, respectively. Also, $T_{cts\_timeout}$ (or $T_{ack\_timeout}$) refer to time intervals before a station assumes its RTS transmission (or Data transmission) has been unsuccessful and triggers a packet retransmission. In addition, $H = PHY_{hdr} + MAC_{hdr}$ is the packet header size and δ is the propagation delay.

3.5 Model Validation

To validate our model, we compare the analytical results of the saturation throughput when RTS/CTS scheme is used in our model, the Bianchi model [58], the Wu model [59] and the simulation performed in OPNET [61][4]. The values of the parameters used to obtain numerical results, for both the analytical model and the simulation are given in table 3.2. [5]

Table 3.2: System parameters for MAC and DSSS PHY Layer

Packet payload	11680 bits
MAC header	224 bits
PHY header	192 bits
ACK	112 bits + PHY header
RTS	160 bits + PHY header
CTS	112 bits + PHY header
Channel bit rate	2Mbps
Propagation delay	1 μs
Slot time	20 μs
$CTS_{Timeout}$	300 μs
$MACK_{Timeout}$	300 μs
SIFS	10 μs
DIFS	50 μs

The results, depicted in figure 3.4 clearly show that the proposed analytical model is more accurate than Bianchi [58] and Wu [59] model.

This is mainly because both models in paper [58] and [59] overestimate the results of 802.11 because they do not consider the impact of different retry limits in the Markov chain transitions as considered in this study. More specifically, [58] overestimates the throughput since it does take into account any of the retry limits and the impact of 802.11 timeout. While paper [59] improves the accuracy of the

[4]The details of the simulator and its parameters are given fully in chapter 6.

[5]The system values used in this chapter are those specified for the DSSS (Direct Sequence Spread Spectrum) physical layer

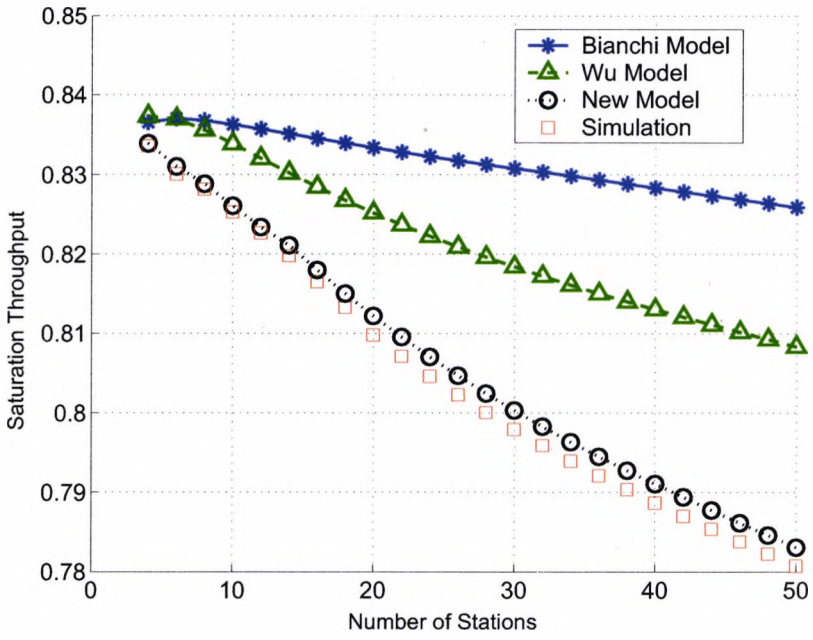


Figure 3.4: Analytical achieved throughput of 802.11 for different number of stations

analytical results, it still fails to closely follow the behavior of 802.11 as it only considers a single retry limit for all packets and uses a simplified throughput analysis. On the other hand, our model addresses the above shortcomings by including the impact of both short and long retry limits in the model and refining the throughput calculation by eliminating the effect of "fake" throughput (i.e. control packets throughput) from the total throughput.

3.6 Impact of 802.11 Parameters on Throughput

Having validated the developed 802.11 model, in this section we investigate the impact of different 802.11 parameters on the achieved throughput. In particular, the aim is to find out the optimum 802.11 parameters for which the analytical throughput calculated earlier would be maximum. Using the system parameters given in table (3.2) and based on equation ?? let us first consider figure 3.5 that depicts the theoretical saturated throughput of the DCF when RTS/CTS is enabled and $P_{col}^{Data} = 0.001$.

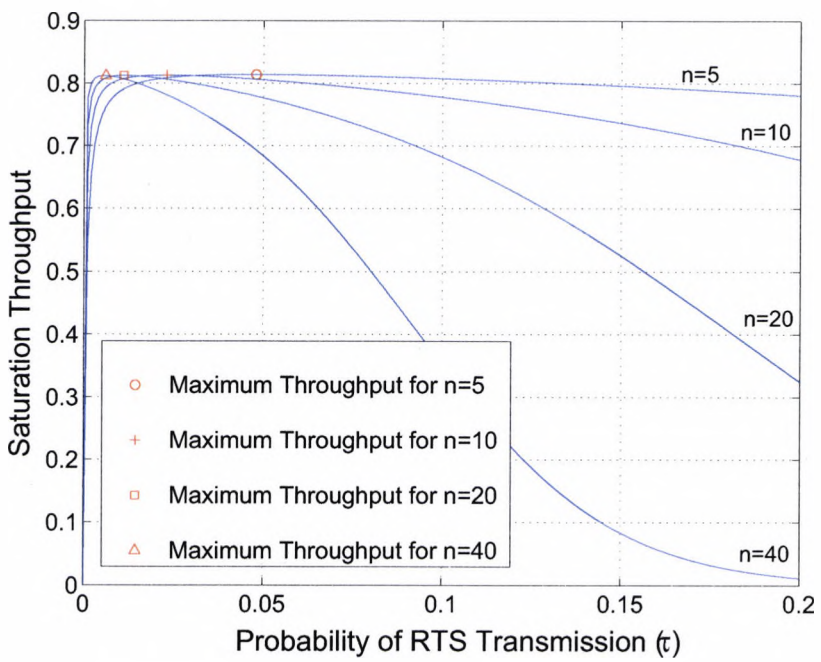


Figure 3.5: 802.11 analytical throughput versus τ

It is clear that as number of stations (n), increases the throughput becomes

very sensitive to the change of τ. To understand the importance of this observation, let us denote the value of τ at which the throughput is maximum (shown by different marks in figure 3.5) as *optimum* τ (τ_{opt}). Using this notation, figure 3.5 shows as the number of nodes increases, it becomes increasingly important for the network to operate very close to τ_{opt} as even a slight deviation in the value of τ_{opt} can severely degrade the system throughput.

It is also interesting to note that in figure 3.5, the value of maximum throughput remains almost constant under wide range of n and P_{col}^{Data}. To further support the above argument, table 3.3 presents the maximum throughput achieved under different values of P_{col}^{Data} and n.

Table 3.3: Maximum achievable throughput for different number of nodes under varying P_{col}^{Data}

	$n = 5$	$n = 10$	$n = 20$	$n = 40$
$\boldsymbol{P_{col}^{Data}}$=0.001	$Th_{max} = 0.81164$	$Th_{max} = 0.80972$	$Th_{max} = 0.80879$	$Th_{max} = 0.80832$
$\boldsymbol{P_{col}^{Data}}$=0.005	$Th_{max} = 0.8111$	$Th_{max} = 0.80918$	$Th_{max} = 0.80824$	$Th_{max} = 0.80777$
$\boldsymbol{P_{col}^{Data}}$=0.01	$Th_{max} = 0.81042$	$Th_{max} = 0.80849$	$Th_{max} = 0.80755$	$Th_{max} = 0.80708$
$\boldsymbol{P_{col}^{Data}}$=0.05	$Th_{max} = 0.80475$	$Th_{max} = 0.80277$	$Th_{max} = 0.80181$	$Th_{max} = 0.80132$
$\boldsymbol{P_{col}^{Data}}$=0.1	$Th_{max} = 0.79709$	$Th_{max} = 0.79503$	$Th_{max} = 0.79404$	$Th_{max} = 0.79354$

To find out the value of τ at which throughput is maximum (i.e. τ_{opt}), equation ?? can be rewritten as follow:

$$S = \frac{E[P]}{\left(\frac{\left(\frac{1-(P_{suc}^{RTS}+P_{col}^{RTS})}{P_{suc}^{RTS}}\right)+\left(\frac{P_{col}^{RTS}}{P_{suc}^{RTS}} * \frac{T_{col}^{RTS}}{\sigma}\right)+\left(P_{col}^{Data} * \frac{T_{col}^{Data}}{\sigma}\right)}{P_{suc}^{Data}} \right) + T_{suc}^{Data}} \tag{3.13}$$

As T_{suc}^{Data},P_{suc}^{Data}, and $E[P]$ are constant, the throughput S is maximized when

the following expression is maximized:

$$\left(\frac{P_{suc}^{RTS}}{1-(P_{suc}^{RTS}+P_{col}^{RTS})+P_{col}^{RTS}T_{col}^{*RTS}}\right) \tag{3.14}$$

where $T_{col}^{*RTS} = T_{col}^{RTS}/\sigma$

By substituting the P_{col}^{RTS} from equation (3.9), equation (3.14) becomes:

$$\begin{aligned} & \frac{n\tau(1-\tau)^{n-1}}{(1-\tau)^n + T_{col}^{*RTS}\left[1-\left(n\tau(1-\tau)^{n-1}+(1-\tau)^n\right)\right]} \\ = & \frac{n\tau}{(1-\tau)+\frac{T_{col}^{*RTS}}{(1-\tau)^{n-1}} - T_{col}^{*RTS}(1+n\tau-\tau)} \end{aligned} \tag{3.15}$$

Taking the derivative of equation (??) with respect to τ, and imposing it equal to 0, we obtain, after some simplifications, the following equation:

$$\left((1-\tau)+\frac{T_{col}^{*RTS}}{(1-\tau)^{n-1}} - T_{col}^{*RTS}(1+n\tau-\tau)\right)+\tau\left(1-T_{col}^{*RTS}\left(\frac{n-1}{(1-\tau)^n}-(n-1)\right)\right)=0 \tag{3.16}$$

It is very interesting to note that according to equation (3.16), the optimum τ at which maximum throughout occurs is solely dependent on number of contending stations (n). Table 3.4 presents the optimum τ for different number of n using equation (3.16.)

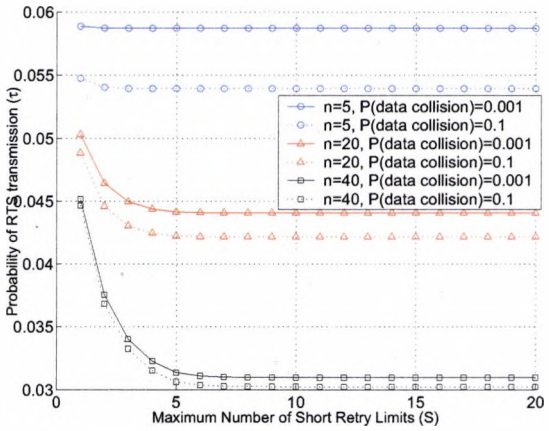
Table 3.4: Optimum τ at which 802.11 throughput is maximum

$n = 5$	$\tau_{opt} = 0.0480$
$n = 10$	$\tau_{opt} = 0.0229$
$n = 20$	$\tau_{opt} = 0.0112$
$n = 40$	$\tau_{opt} = 0.0055$

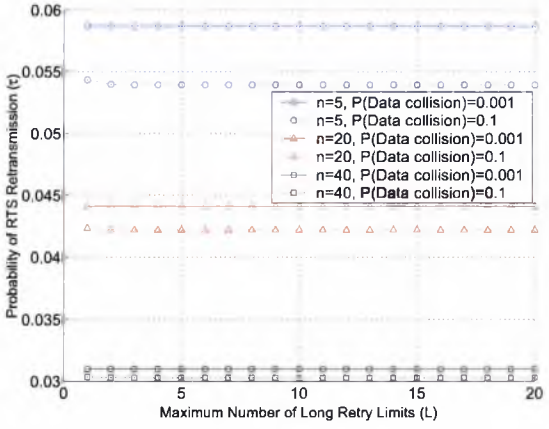
Comparing the default values of τ (τ_{def}) shown in table (3.1) and the optimum τ (τ_{opt}) shown in table (3.4), it is obvious the default values of 802.11 parameters

tend to overshoot the optimum τ in which the maximum throughput is achieved. In other words, while for a given number of contending stations, there exists an optimal value of packet transmission probability at which 802.11 achieves the highest throughput, the current version of 802.11 does not operate around this optimum packet transmission probability. As shown earlier in figure 3.5, this can severely degrade 802.11 performance as the throughput becomes extremely sensitive to the value of τ as the number of contending stations increases.

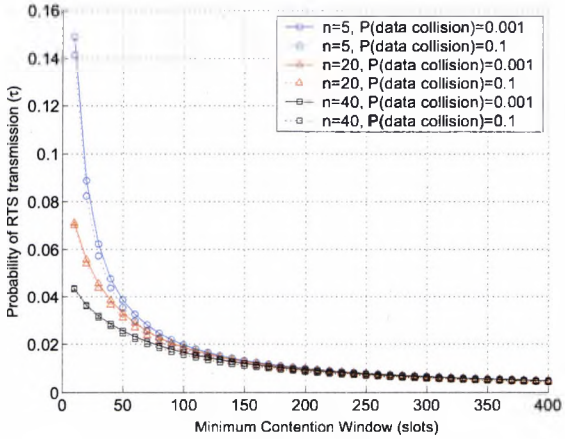
In order to tune the τ_{def} to τ_{opt}, let us remember that τ is the sum of $b_{i,j,0}$ stages in the Markov chain. Therefore, there are three parameters that can effect the value of τ: Number of Planes (i.e. Long Retry Limit or L), Number of rows in each plane (i.e. Short Retry Limit or S), and Contention window size (i.e W_0). Therefore, the issue of finding the optimal τ becomes finding the corresponding values of L, S, and W_0. However, before that, let us investigate the impact of S, L, and W_0 on the value of τ. This can be very useful since it can shed light on the main parameter that can substantially change τ. To this aim, figure 3.6 depicts the impact of changing different MAC parameters on τ under different number of stations and P_{col}^{Data}.



(a) Impact of maximum short retry limits on τ



(b) Impact of maximum long retry limits on τ



(c) Impact of minimum contention window size on τ

Figure 3.6: The impact of different 802.11 MAC parameters on τ

Surprisingly enough, it is obvious that the value of τ is highly sensitive to change of W_0 (CW_{min}) while the number of long retry limits almost has no impact on τ. Meanwhile, for S>5, the value of τ remains almost constant regardless of the change in maximum number of short retry limits. Therefore, the issue of changing the value of *default* τ towards *optimum* τ can be mainly achieved by adjusting the value of W_0.

Based on the above discussion, figure 3.7 presents the optimum W_0 for different number of n at which $\tau_{def} = \tau_{opt}$ and the 802.11 throughput is maximized.

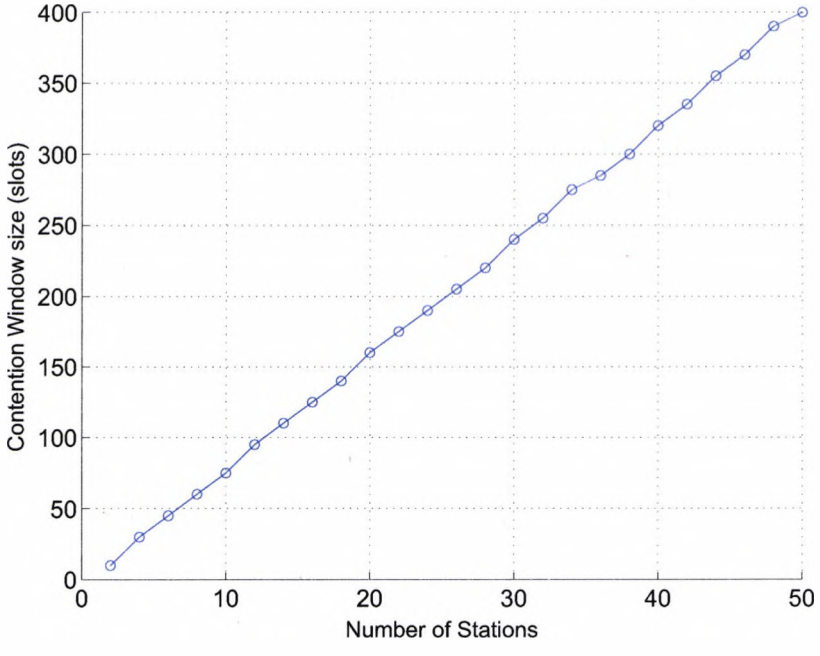


Figure 3.7: Optimum value of W_0 for different number of nodes

Unfortunately, in the 802.11 standard, the value W_0 is hardwired in the PHY layer details, and thus it cannot be made dependent on n. As a consequence of this lack of flexibility, the throughput specially in the large networks can be significantly lower than the maximum achievable.

In addition, it is very important to note that though the results in figure 3.6 suggest the negligible impact of S and L on 802.11 throughput, their values can dramatically change the TCP performance. More discussion of this issue will be given in the next chapter when we investigate the role of retry limits on TCP instability.

3.7 Summary

In this chapter, we proposed a 3-d Markov chain to accurately model the performance of 802.11 when the RTS/CTS mechanism is used. The importance of the proposed model was to examine the impact of different retry limits used by 802.11 MAC. Based on this modified model, an optimum value of packet transmission probability was calculated at which the achieved throughput is maximized. However, it was also shown that the current parameters of 802.11 do not operate around this optimum value and as a consequence this results in a dramatic throughput degradation. It was then determined that the value of W_0 is the main parameter to adjust the probability of packet transmission to its optimum value. However, due to the lack of flexibility to vary the value of W_0, the throughput specially in the large networks can be significantly lower than the maximum achievable.

Chapter 4

TCP Instability in Multihop Ad hoc Networks

4.1 Introduction

Connection instability refers to a situation where the receiver (data sink) does not receive any packets for a period of time and therefore the connection throughput drops to zero. Due to the nature of applications in multihop ad hoc networks (e.g. emergency operation and battlefield communication), connection instability in these networks has always attracted a considerable amount of research interest as disconnectivity or starvation even for a short period of time can have a devastating impact on the QoS and may not be acceptable for the end user. In other words, ad-hoc network users would prefer to receive a continuous and stable flow of data rather than sending/receiving large bulk of data instantly. In particular, as shown by a number of studies (e.g. [15,19,26,29,62,63]), the instability problem becomes more severe when applications use a reliable end-to-end delivery protocol such as

TCP as their end-to-end transport protocol in 802.11 multihop ad hoc networks.

To have a better understanding of the problem and therefore to be able to address it more appropriately, this chapter revisits the TCP instability by giving a number of simple yet important examples that will shed light on the roots of the problem. In particular, we use an in depth cross layer analysis of the chain of events occurring between different layers and reveal some interesting facts which will make the building blocks of the solutions in the next chapter.
To this aim, this chapter divides the TCP instability problem into two broad categories named as *intra-flow* and *inter-flow* instability, where the former is caused by the interaction of nodes belonging to the same TCP connection, while the latter happens when nodes belonging to different connections interact. In sections 4.3 and 4.3, the main causes of each of the above instabilities are explained and the major related work that have addressed them are reviewed, respectively.

4.2 Intra-flow Instability

4.2.1 Description

By definition, intra-flow instability refers to the situation where the successive transmissions of packets in a *single TCP flow* interfere with each other (link layer intra-flow interference) and result in large number of contention related packet drops and hence TCP instability in the network. Therefore, we begin our discussion of TCP intra-flow instability by reviewing different types of intra-flow interference and their impact on TCP instability.

4.2.2 Intra-flow Interference

As mentioned earlier, the intra-flow interference refers to situations where transmission interference in a single TCP flow causes packet drop in the network. When TCP runs over 802.11, the intra-flow interference can be broken down into the following categories:

1. Interference of TCP packets with each other
2. Interference between TCP packets and 802.11 control packets
3. Interference of 802.11 control packets with each other

Here, TCP packets refer to either TCP DATA or TCP ACK packets and 802.11 control packets include MACK (802.11 acknowledgments) and RTS/CTS if used.

To investigate and explain each category, in all subsequent analysis it is assumed one TCP flow is running on a 6 hop chain from node A (as data source) to node G (as data sink) and the transmission range of nodes is shown by a circle around them. Also, without the loss of generality, as an initial value all stations are assumed to have an NULL NAV field and RTS/CTS message is used only for TCP DATA packets.

I- TCP Packets Self Interference

In principle, the TCP packets self interference is caused by two effects. One is the interference caused between TCP DATA (TCP ACK) packets transmission with each other which prevents concurrent transmissions within a neighborhood area. For instance, as shown in figure 4.1, a transmission from node A interferes

with node C, which cannot simultaneously communicate with node D. Similarly, a transmission by node D may cause a collision at node B.

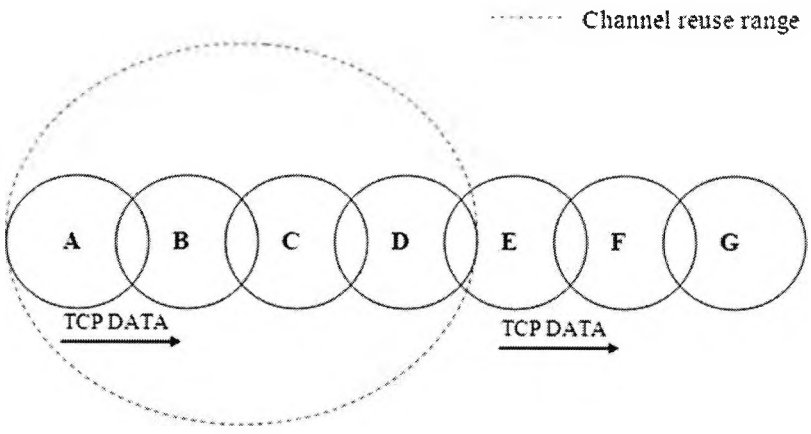


Figure 4.1: TCP packets self interference

This type of interference can harm TCP mainly in two ways. Firstly, it greatly decreases the TCP throughput in ad hoc networks since in most occasions very few simultaneous packet transmissions can occur in the network. For instance, in the above example, links A-B and E-F represent maximum possible concurrent channel usage while if link D-E is active, only one simultaneous transmission is possible. The other impact of such interference is on increasing the end-to-end delay. This is also because a successful transmission can occur only if nodes within the spatial channel reuse of that node are silent during the entire transmission. This means packets have to wait for a relatively long period of time in the node's buffer before the node can get a chance to access the channel. Therefore, the packets in multihop connections experience longer queuing delay and hence larger end-to-end delay.

The second part of the TCP self interference is caused by interference between

TCP DATA and TCP ACK packets along the forward and return paths, respectively. In essence, this interference can specially result in TCP ACK drop as there are larger number of TCP DATA frames on the forward route compared to the smaller number of the TCP ACK packets in the return path. So, the medium will be on average mostly accessed by TCP DATA frames and as a result significant amount of ACKs will be lost because of collisions while accessing the channel.

II- TCP and 802.11 Control Packets Interference

The other type of intra-flow interference in the link layer happens between the TCP packets (TCP DATA or TCP ACK) and one of the 802.11 control packets (RTS, CTS , or MACK). However, it is important to note that regarding 802.11 MAC timing specification, the DCF protocol ensures that CTS frame transmission will be successfully received at its destination (the one who sent the RTS), if the CTS frame has been issued in response to the RTS. This is because successful RTS frame transmission silences all the nodes in the neighborhood of the source either for a duration specified in the duration field of the RTS or for EIFS time (if collision occurs) which is large enough to transmit a CTS. Therefore, the CTS frame cannot collide with any frame at the source. Using a similar argument, it can be concluded that a successful TCP frame transmission ensures a successful MACK frame transmission. Thus, there cannot be CTS and MACK frames drop at the intended destination (the node who is waiting to receive the packet) because of medium contention. Figures 4.2 to 4.4 review the most common scenarios where a control packet and a TCP packet collide and are dropped as a result of this type of interference. Below each figure there is also a brief description of the sequence of events leading to the packet drop.

- **Scenario 1**

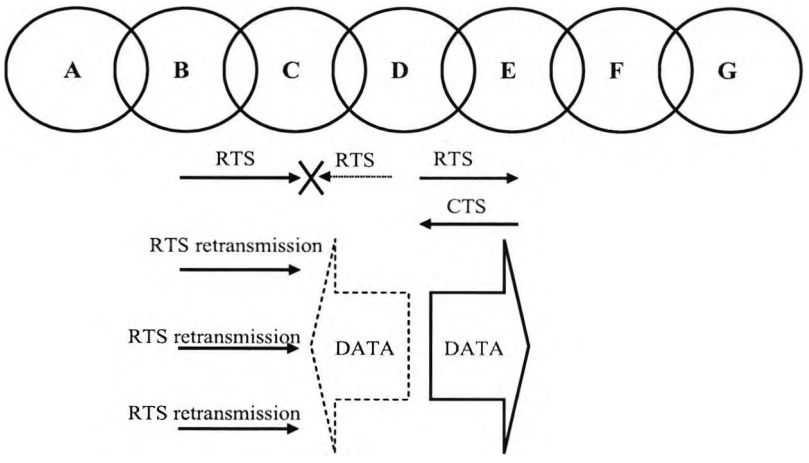


Figure 4.2: RTS & TCP DATA collision

1. Station D has TCP DATA to send to E. Therefore, it initiates RTS transmission towards node E

2. Station E transmits CTS back to D and consequently D starts transmitting TCP DATA to E

3. Meanwhile B has a TCP DATA to send to C, thus starting its own RTS handshake.

4. Due to ongoing TCP DATA transmission between D and E, the B's RTS is dropped at C.

5. B resends the RTS after an exponential backoff; however due to large size of TCP DATA, in most of the cases all RTS retransmissions (7 by default)

are collided at C, resulting in TCP DATA drop at B

- **Scenario 2**

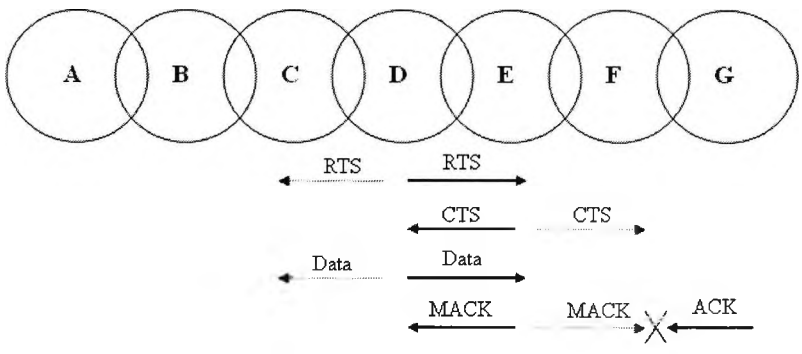


Figure 4.3: TCP ACK & MACK collision

1. Station D has TCP DATA to send to E. So, it starts initiating RTS transmission.

2. Station E transmits CTS back to D and consequently D starts transmitting TCP DATA to E

3. Station E sends an MACK at the end of successful reception of TCP DATA.

4. At the same time the TCP ACK is flowing back from G to F. where G is not aware of any ongoing transmission between E and D.

5. Collision happens at F between TCP ACK frame and unintended MACK received by F and TCP ACK is dropped.

- **Scenario 3**

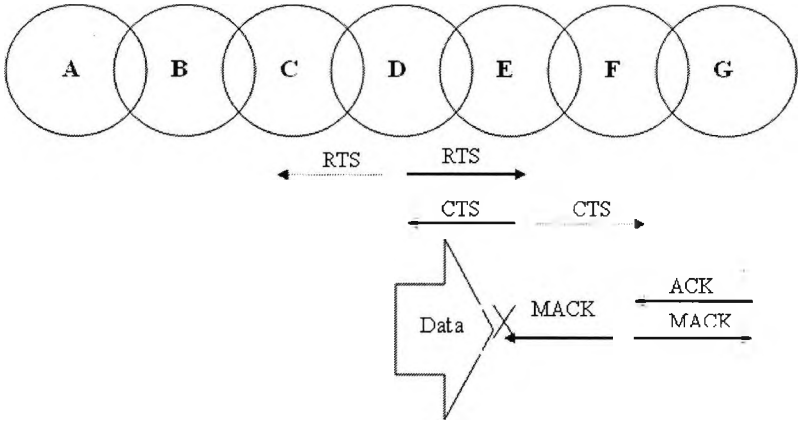


Figure 4.4: TCP DATA & MACK collision

1. Station D has TCP DATA to send to E. So, it starts initiating RTS transmission.

2. Station E transmits CTS back to D and consequently D starts transmitting TCP DATA to E

3. Meanwhile station G sends a TCP ACK to F.

4. Upon reception of TCP ACK, F triggers an MACK transmission to G. Note that in general F cannot send anything as it has updated its NAV by D's CTS. However, regarding the 802.11 standard, the MACK should be sent immediately irrespective of NAV duration.

5. Collision happens at E between the transmitted MACK and ongoing TCP DATA. So TCP DATA will be dropped despite of successful RTS/CTS transmission and needs to be retransmitted.

III- 802.11 Control Packets Self Interference

The last type of intra-flow interference we consider, happens between 802.11 RTS and CTS control packets if the RTS/CTS handshake is used prior to data transmission. We should note that collision of RTS and CTS packets are expected from time to time and indeed part of their natural design to avoid data packet collisions. However, their frequent loss can waste channel resources and have undesirable and negative impact on the performance of higher layers. On the other hand, the self interference between the RTS and CTS control packets can cause failure of the channel reservation scheme and lead to a loss of data packets as well. Figure 4.5 depicts a typical scenario of control packet self interference and loss of TCP packets as a result.

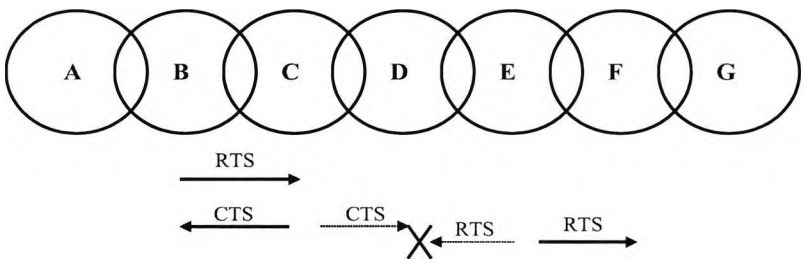


Figure 4.5: 802.11 control packets collisions

Here B starts an RTS-CTS handshake with C before transmitting a TCP packet. The CTS reply from C is received by B correctly, but it is not received by D, which is hidden from B, due to a collision with an RTS packet sent from E to F. This happens because E, being far away from both B and C, does not hear either the RTS or the CTS packet and is unaware of the communication between B and C. Node B assumes that the channel is successfully reserved and proceeds

with transmission of the data packet to C. Therefore, the TCP transmission from B is vulnerable to interference from D, which has not been able to set its NAV accordingly, and may initiate a transmission to any of its neighbours before the data transmission is over.

4.2.3 Intra-flow Interference and Intra-flow Instability

Having reviewed the three types of intra-flow interferences, this section explains how such interference (which happen at the link layer), can create TCP intra-flow instability. However, before that and to show the damaging impact of intra-flow interference on TCP stability, let us review figure 4.6 that shows the change of congestion window (cwnd) and the instances of TCP retransmissions in a static 6 hop 802.11 ad hoc chain topology similar to one shown in figure 4.1. Here, to confine the packet losses to contention drop, its is assumed the channel is error-free, no routing messages are exchanged between the nodes and all nodes have infinite buffers. Therefore, the packet losses and retransmissions are restricted to intra-flow interference related drops.

It is clear from the result in figure 4.6 that intra-flow interference can trigger a large number of TCP retransmissions, TCP congestion window fluctuation and therefore TCP instability.

To explain further how intra-flow interference can cause TCP instability, let us recall from section 2.2 that if a node cannot reach its adjacent node within the limited number of allowed retries (MAC-Retry-Limit), it will drop the packet. Such TCP packet (contention) drop can result in TCP instability in two ways; In the first situation, such contention drops trigger a route failure and hence

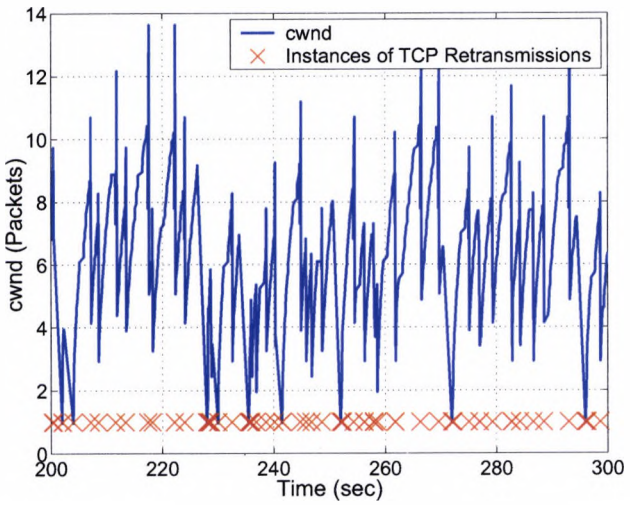


Figure 4.6: Illustration of TCP congestion window size in a 6 hop chain topology

route discovery process at the source[1]. During this process and while the new route is established, as no packet can be sent out at the TCP sender, the TCP receiver will experience a period of time where it does not receive any packet and hence experience instability [64]. This type of intra-flow instability can be therefore referred as *re-routing instability*. It is important to note that the TCP packet losses during re-routing instability are primarily recovered through the TCP timeout retransmission mechanism since the route discovery stage can be very long compared to the TCP RTO value.

The other impact of TCP intra-flow interference that may result in TCP instability can be traced back to frequent packet losses since such frequent losses can trigger multiple congestion window drops and TCP retransmissions during a short period of time (as shown in figure 4.6). Such frequent losses result in creating out of order packet delivery at the TCP receiver and therefore TCP instability. For that reason, we refer to this type of intra-flow instability as *out*

[1]Here source means data packet source which can be either the TCP sender or receiver

of order instability. It should be noted that unlike the re-routing instability, out of order instability generally triggers TCP fast retransmit at the sender due to duplicated ACKs resulting from out of order packet deliveries.

One interesting question that may arise here is which of the two TCP loss recovery algorithms (timeout and fast retransmit) are triggered more when the only source of packet loss is contention drops? The answer to this question is very important as it can help us to find out which of the above TCP intra-flow instabilities (rerouting instability or out of order instability) happen more often in the network. Then based on that, it is easier to find out the parameter(s) that can be tuned to decrease the number of false TCP retransmissions and therefore alleviate the TCP instability. Figure 4.7, depicts the average number of TCP retransmissions triggered by fast retransmit and TCP timeout individually over a different number of hops in a chain topology.

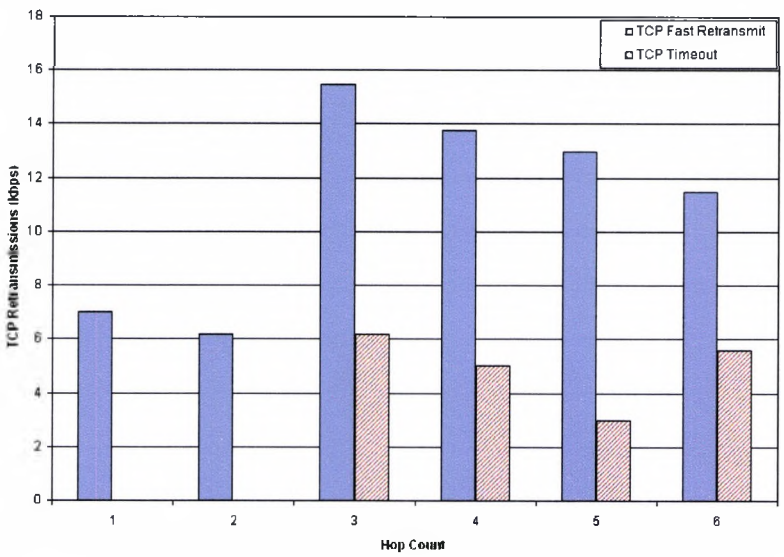


Figure 4.7: Cause of TCP retransmissions in a 6-hop chain topology

It is quite interesting that in all scenarios, a high percentage of the false TCP

retransmissions are triggered because of the receipt of duplicate ACKs. This clearly shows the high sensitivity of the default TCP to the current number of duplicate ACKs before retransmitting the packet and invoking congestion control in multihop ad hoc networks. In other words, the current threshold value of the fast retransmit duplicate ACK (3 as specified in RFC 2581 [39] in wired networks) seems to be very small in wireless multihop ad hoc networks as the packet reordering can happen very often due to frequent packet contention drops.

4.2.4 Discussion

Having shown the negative impact of intra-flow interference on TCP instability, the next question is how it is possible to minimize the intra-flow packet interference in multihop ad hoc networks? The answer to this question is not straightforward as each type of intra-flow interference discussed above are different in nature and therefore need to be addressed separately. For instance, TCP packets or 802.11 control packets self interference can be best eliminated by designing a smart decentralized link layer scheduler that coordinates the concurrent transmission between different pairs to maximize the channel utilization and minimize the number of collisions. On the other hand, TCP with 802.11 control packets interference is best to be addressed by reconsidering the link layer timing specifications, packet transmission coordination and prioritization. However, such schemes can be quite topology dependent and confined to specific scenarios. This is obviously hard to achieve in dynamic multihop ad hoc network environments where the topology of the network is changing rapidly and it is not feasible to propagate global topology information to individual nodes. More importantly, due to scarce channel resources, it is simply unrealistic to broadcast information

regarding the topology and the current activity of nodes across the network.

As a next alternative solution which indeed makes the building blocks of the next chapter, we propose to alleviate all types of intra-flow interferences discussed above by controlling the amount of outstanding data in the network. It should be noted that, though this approach does not fully solve the individual causes of intra-interference explained before, it addresses the problem by reducing unnecessary intra-flow interference and its adverse effects on TCP instability. To better understand this, let us recall from section 2.3 that the performance of TCP directly depends on the *swnd* which its optimal value should be proportional to bandwidth-delay product (BDP) of the entire path of the data flow [17, 65]. It is important to note that exceeding this threshold does not bring any additional performance enhancement, but only leads to increased buffer size in intermediate nodes along the connection. On the other hand, as shown in [15, 17, 26, 66], the BDP of a TCP connection over multihop 802.11 networks tends to be very small. This is mainly because in 802.11 MAC, the number of packets in flight is limited by the per-hop acknowledgements at the MAC layer. Such a property is clearly quite different from wire-line networks, where multiple packets can be pushed into a pipe back-to-back without waiting for the first packet to reach the other end of the link [67]. Therefore, as compared with that of wired networks, ad hoc networks running on top of 802.11 MAC, have much smaller BDP. However, as shown in [15, 26], TCP grows its congestion window far beyond its optimal value and overestimates the available BDP.

To get a better understanding of how TCP overestimates the available bandwidth-delay product in ad hoc networks, consider a simple scenario in figure 4.8 where all nodes can only access their direct neighbours.

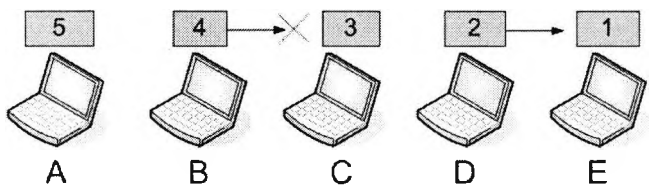


Figure 4.8: Initial stage of buffer queues in a 4 hop chain topology

Here a TCP connection is running from node A to E and all nodes have at least one packet to send in the forward direction. In addition, the congestion window size is also equal to 8 packets. Let us assume nodes B and D initially win the channel access and start to transmit their data (either a TCP packet or an RTS packet) into the network at the same time. Soon after both stations start transmitting their data, the packet from B→C collides with the interference caused by the D→E transmission. Following this case, node A is very likely to win access to the channel and starts transmitting several consecutive packets towards B before releasing the channel. Meanwhile, since B is unable to access the channel it buffers the new packets in addition to packet(s) already in its buffer and starts building up its queue (figure 4.9).

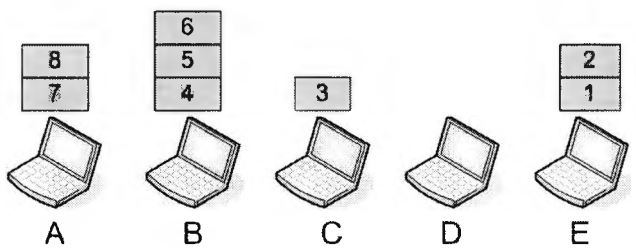


Figure 4.9: Illustration of buffer queue build up in a 4 hop chain topology

This results in an artificial increase of the RTT delay measured by the sender as node B now becomes the bottleneck of the path. Such situation leads to an overestimate of the length of available data pipe and therefore an increase

of the TCP congestion window and hence more network overload in the next RTT. Figure 4.10 summarizes the chain of actions that occur following a network overload and lead to TCP intra-flow instability (instability cycle).

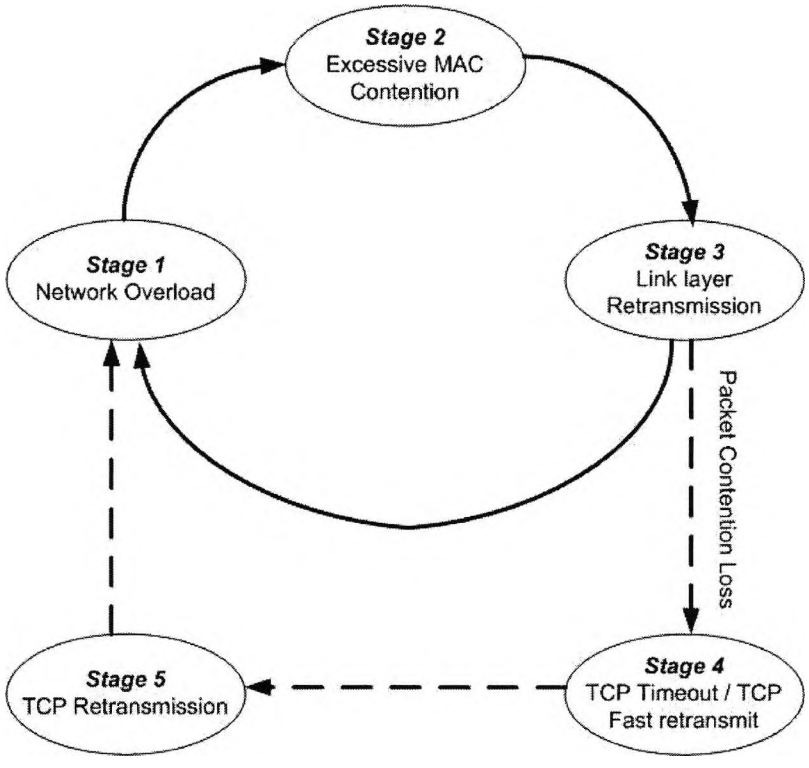


Figure 4.10: Intra-flow instability cycle

The instability cycle initially starts when increasing the network overload (stage 1) causes more contention among nodes as all of them try to access the channel (stage 2). On the other hand, when the level of contention goes up, more packets need to be retransmitted as the probability of collision increases with the increasing level of contention (stage 3). This in turn introduces extra network overload and therefore closing the inner part of the cycle (stage 1→stage 2→stage 3→stage 1). This cycle is continued until one or more nodes cannot reach its adjacent node within a limited number of tries (specified by the MAC Retry

Limit in 802.11 MAC standard) and drop the packet. This packet contention loss is then recovered by the TCP sender either through TCP fast retransmit or through TCP timeout (stage 4). In both cases, TCP drops its congestion window resulting in a sharp drop in number of newly injected packets to the network (stage 5) and therefore giving the network the opportunity to recover. However, soon after TCP restarts, it creates network overload again by overestimating the available BDP of the path, and the cycle repeats.

4.2.5 Related Work

During recent years, many studies have shown that limiting the amount of outstanding data in the network can greatly improve TCP performance. In an early paper by Gerla et.al. [14], the authors showed by simulations that TCP performance degrades for congestion window greater than 1 packet when the MAC layer offers no MACK protection; They further showed that, with the MACK protection, certain performance gain can be realized by allowing a slightly larger congestion window (2-3 packets).

To limit the amount of outstanding data in the network, [62] proposed to adjust the maximum window size parameter to 4 packets as this is the smallest value of window size for facilitating the fast retransmission scheme for TCP connections running over IEEE 802.11 based ad-hoc networks.

The authors in [19] showed that due to the spatial reuse and transmission interference property of the IEEE 802.11 MAC layer protocol in a chain topology, a sensible choice is to set TCP congestion window to $\frac{1}{4}h$, where h is the length of the chain. They further showed that TCP tends to overshoot this optimal value and operates at a larger window size as we explained earlier. In the same

paper, the authors proposed Link-layer Random Early Detection (LRED), which aims to control the TCP window size by tuning the link-layer dropping probability according to the perceived channel contentions. In essence, similar to the RED algorithm [68] with a linearly increasing drop curve as the queue size exceeds a minimum value, LRED increases its packet dropping probability when the link-layer contention level, measured by the retransmission counts, exceeds a minimum threshold. To this aim, in LRED the link layer maintains a moving average of the number of packet retransmissions and the head-of-line packet is dropped/marked with a probability based on this average retransmission count. In particular, at each node, if the average retransmission count is smaller than a minimum threshold, the head-of-line packets are transmitted as usual. When the average retransmission count becomes larger, the dropping/marking probability is set as the minimum of the computed dropping probability and an upper bound. It was shown that LRED can provide an early sign of network overload to the transport layer protocol and therefore force the TCP sender to reduce its transmission rate.

The authors in [69] propose Slow Congestion Avoidance to limit TCP's window growth rate to a level below the standard of one segment per RTT. This is intended to reduce the number of packets in the network without putting hard constraints on the maximum window size. In particular, the slow congestion avoidance modification increases the window size by one segment after a given number of round trip times with successful acknowledgment receptions.

In a different approach, [17] turned the problem of setting TCP's optimal congestion window into identifying the BDP of a path in ad hoc networks. They first showed that, independent of the MAC layer protocol being used, the upper

bound of the BDP of a path cannot exceed N×S where N is the number of round-trip hops and S is the size of the TCP data packet, assuming similar bottleneck bandwidth along the forward and return paths. In other words, they showed the number of outstanding data in the network in ad hoc networks cannot exceed the round-trip hop-count (RTHC) of the path. Then based on the 802.11 MAC layer protocol, they showed in a chain topology, a tighter upper bound exists which is approximately $\frac{1}{5}$ of the RTHC of the path. Based on this tighter bound, they proposed an adaptive congestion window setting strategy to dynamically adjust TCP's congestion window according to the current RTHC of its path.

To decrease the amount of channel contention in the network, the authors in [22] present a cross layer approach named Adaptive TCP that adaptively adjust the TCP maximum window size according to the number of RTS retransmissions at the MAC layer at the TCP sender to control the number of data packets in the network and thus decrease the channel contention.

In a similar effort, Dynamic Delayed ACK proposed in [70] aims to reduce the contention in the wireless channel, by decreasing the number of TCP ACKs transmitted.

The focus of [71] for alleviating self-contention is on contention between packets belonging to the same transport layer flow. The authors propose Quick Exchange and Fast Forward which are both extensions to 802.11's RTS/CTS mechanism, to reduce intra-flow interference. Quick Exchange allows to exchange two packets in opposite directions by using only one exchange of RTS/CTS information. By adding an extra duration header to the first data transmission the network allocation vector of all other nodes in range is extended appropriately. A packet in the opposite direction with a piggy-backed ACK can then be sent directly,

i. e. without a second RTS/CTS. After both transmissions the original sender completes the procedure with an additional ACK. Fast Forwarding speeds up the forwarding of a packet in the downstream direction. Like Quick Exchange there is only one exchange of RTS/CTS information. But here the ACK is piggy-backed with a new RTS packet for forwarding. This way, packets are forwarded faster over multiple hops to avoid self contention with other packets of the same flow. The authors in [72] propose to assign a higher priority for the medium access to a node that has just received a packet. This is to give "downstream" packet transmissions a higher priority and thus to alleviate intra-flow interference. In addition, a hop-by-hop backward-pressure scheme keeps upstream nodes from sending further packets until the previous ones have been forwarded. This mechanism is tightly coupled to the 802.11 RTS/CTS mechanism, allowing the receiving node to send a negative CTS (NCTS) in order to signal that it is not willing to receive another packet of a certain flow. The upstream node then has to wait until its next hop explicitly gives the permission to continue.

In [64], the authors studied the interaction between the transport and the on-demand routing protocols, and showed that the utility of TCP in 802.11 multihop networks lies in the ability to keep the routing information robust and stable. To fix the problem, they proposed a Fractional Window increment (FeW) scheme for TCP to prevent the over-reaction of the on-demand routing protocol by limiting TCPs aggressiveness. In essence, using FeW, the TCP congestion window grows by a fractional rate $1/\alpha$ (packets) at every round trip-time, where α is the probing rate.

To address the negative impact of the high probability of out of order packet

delivery in ad hoc networks, TCP-DOOR (Detection of Out-of-Order and Response) [73] attempts to detect and respond to out-of-order packet delivery events and thus avoiding invoking unnecessary congestion control. In order to detect out of order packets, ordering information is added to TCP ACKs and TCP data packets and out of order detection is carried out at both ends in the following way: the sender detects the out of order ACK packets and the receiver detects the out-of-order data packets. If the receiver detects out of order packets, it notifies the sender. Once the TCP sender knows of an out of order condition, it may take one of the two responsive actions: temporarily disabling congestion control or instant recovery during congestion avoidance. The first action means that, whenever an out of order condition is detected, the TCP sender will keep its state variables such as RTO and the congestion window size constant for a specific period. The second action means that, if during the specific last period, the TCP sender has already entered the state of congestion avoidance, it should recover immediately to the state prior to such congestion avoidance.

Finally, in a completely different approach, Wireless TCP (WTCP) [74] uses rate base scheme (that does not use ACKs for self-clocking) rather than the window-base transmission control to eliminate sending bursts of traffic into networks. WTCP uses the ratio of the inter-packet separation at the receiver and the inter-packet separation at the sender as the primary metric for rate control rather than packet loss and retransmit timeouts. As a result, WTCP reduces the effect of non-congestion related packet loss on the computation of transmission rate.

4.3 Inter-flow Instability

Unlike the intra-flow instability that is caused by the interaction of nodes belonging to the same connection, TCP inter-flow instability is mostly observed when multiple flows compete to access the shared channel. More precisely, inter-flow instability refers to the situation where for a period of time, one or more TCP connections is able to monopolize the channel resources at the expense of other contending connections. This implies when multiple connections are sharing the channel, some connections may experience a period of time where no packet can be received by their TCP receiver and hence experience an instability. Therefore, in a broad sense it can be claimed that TCP inter-flow instability is closely linked to the unfairness problem in multihop ad hoc networks. In the next subsection, we further verify the inter-flow instability and unfairness relationship and discuss in fine detail the main causes of unfairness in multihop ad hoc networks.

4.3.1 Fairness and Inter-flow Instability

To investigate the cause of inter-flow instability, we begin our discussion by analyzing a simple cross topology shown in figure 4.11 where connection 1 runs from node A to node G and connection 2 runs from nodes H to node M.

To start with, let us consider the case where nodes C and J are competing in their first attempt to access node D (which is shared between the 2 connections). As the contention windows (CW) at both stations are very small (e.g. less than 31) the transmission of RTSs of nodes C and J may very likely overlap partially, and as a result there will be a collision. The collision may occur several times until the CWs are large enough to allow either node to get control of the medium

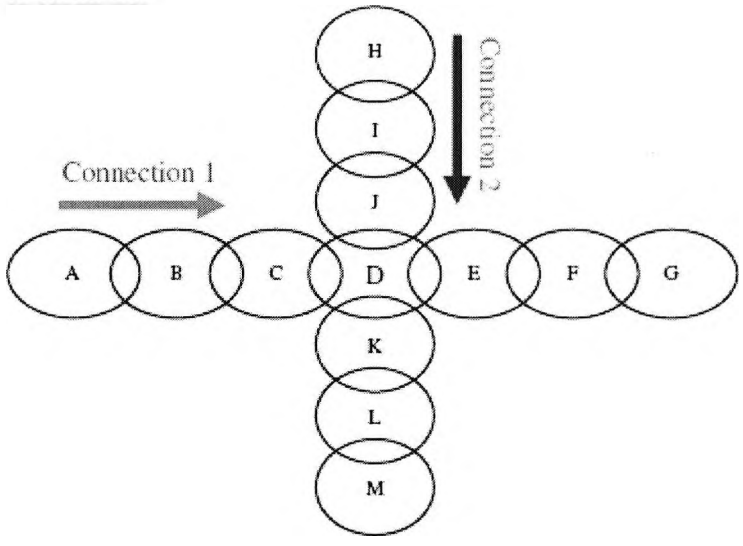


Figure 4.11: A 6 hop cross topology with two connections

[75]. In particular, one of the two nodes (let us say, node C) may select a small backoff time from its CW, while the other node (i.e., J) selects a large value resulting in letting the C-D RTS/CTS handshake to be successfully completed. Once the data transfer is completed, node C resets its CW and backoff before initiating another handshake. However, the remaining backoff timer at node J may be large compared to the backoff timer at node C, which is drawn from the range [0, CW_{min}]. In that case, nodes C and D may exchange several more frames (belonging to connection 1) before node J's back-off timer reduces to zero. Whenever the backoff timer at node J reduces to zero, node J starts transmitting to node D. However, as the CW at node C is equal to CW_{min} (because of previous successful transmission) the contention is most likely to result in a collision. After the collision, node C doubles its CW from CW_{min} whereas node J doubles its CW from a larger value ($CW >> CW_{min}$) as it could not succeed in its previous

transmission. Therefore, the *CW* at node J is very likely greater than that at C and node C is more likely to get control of the medium again!

This obviously brings connection 2 into TCP instability situation as connection 2 has been starved during this period and is unable to send its data to the destination. Even worse, this process (i.e., several packet transmissions from connection 1) may repeat several times if the J→D RTS/CTS handshake starts around the same time as B→C or E→F RTS/CTS handshakes; as in all cases node D will still be reserved by connection 1.

The simulation results shown in figure 4.12, fully support the above argument and confirm that the BEB algorithm used in 802.11 MAC can cause severe inter-flow instability for one or more connections in multihop ad hoc networks. Note that the results in figure 4.12 are obtained with the assumption that both connections are using UDP and no routing information is exchanged between the nodes. In this manner, the possible cause of inter-flow instability originated from the routing and TCP are eliminated and therefore it is possible to narrow down the problem to the operation of BEB in 802.11 MAC.

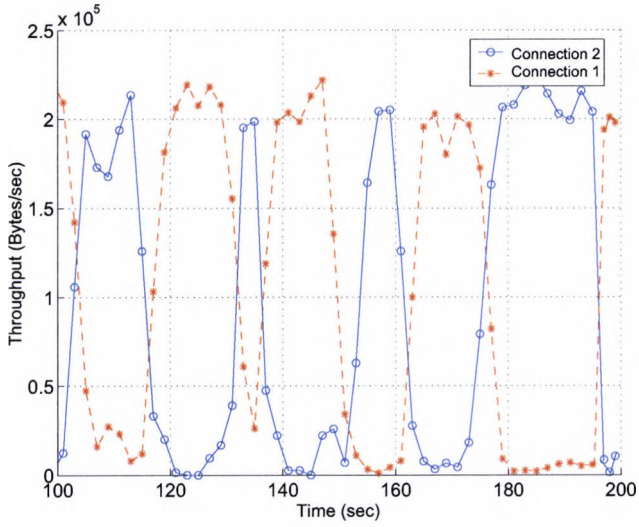


Figure 4.12: Inter-flow instability of two UDP connections over 802.11

Having analyzed the role of BEB on connection instability, let us consider figure 4.13 where both connections in figure 4.11 now run TCP and the DSR routing protocol is deployed over 802.11 MAC.

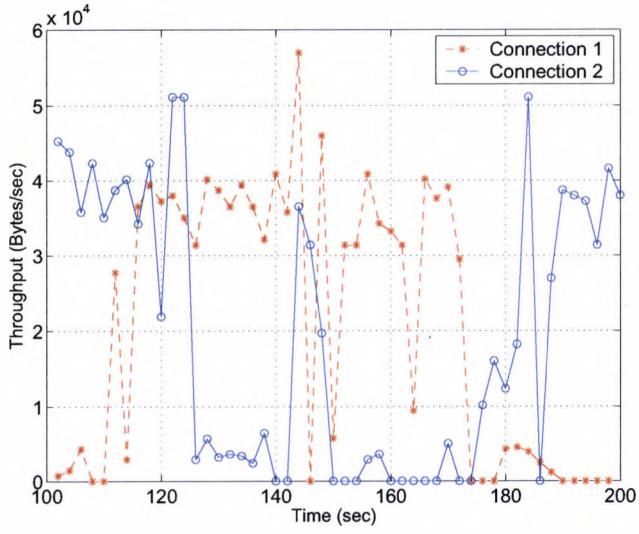


Figure 4.13: Inter-flow instability of two TCP connections over 802.11

It is clear that running TCP and also adding the routing scheme has deteriorated the connections stability. This can be because of couple of reasons; first of all the extra routing messages will create additional overhead in the network and therefore create higher inter-flow interference. In addition, the unfairness will trigger a number of false route failures and other problems discussed in section 4.2.3. On the other hand, the channel access unfairness created by BEB will also trigger a number of false TCP congestion control drops and therefore more instability as discussed in section 4.2.4. Therefore, from a comparison between figure 4.12 and 4.13, it can be concluded that the main cause of inter-flow instability lies in the operation of 802.11 BEB algorithm rather than the TCP itself (under the test conditions).

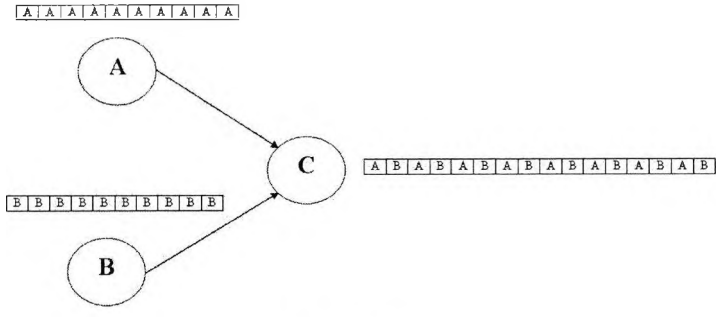
Having shown the close relationship between the inter-flow instability and the fairness aspect of the channel access, in the next subsection we review the key issues in analyzing fairness in contention based channel access schemes. This analysis can be very useful as it can help us to quantify the inter-flow instability in different scenarios and therefore measure the level of inter-flow instability between competing connections using the fairness metric as will be explained below.

4.3.2 Fairness Horizon

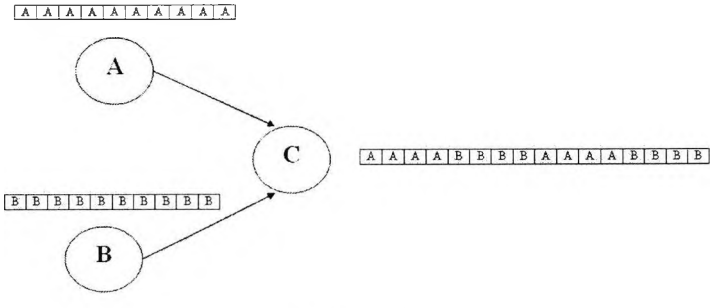
In its simplest form, the fairness of a contention-based channel access protocol refers to its ability to allocate the channel resources (e.g. bandwidth) equally between contending nodes. Based on the length of the time over which we observe the system, the fairness can be defined on a *short-term* or a *long-term* basis. Intuitively, short term fairness of an algorithm refers to its ability to provide equitable access to resources over short time scales while long term fairness, in

contrast, measures the amount of resources assigned over a longer time scale. For two main reasons we are more interested in the short-term fairness rather than long term fairness. First of all, short-term fairness implies long-term fairness but not vice versa. Secondly, as we will see later in this section, the long-term fairness does not guarantee connection stability while short-term fairness implies stability.

Since time scale is the key element in determining short-term or long-term fairness, the next step is to precisely define the time scale and its impact on fairness analysis. To this aim, let us consider two traces of packet arrivals shown in figure 4.14 where nodes A and B act as data source and both compete for access to the data sink at node C.



(a) Trace 1



(b) Trace 2

Figure 4.14: An example on traces of fairness horizon

For simplicity and without loss of generality, assume node C is receiving data either from node A or B every second. The trace of packet arrival at node C shown in figure 4.14(a) is an ideal TDMA-style trace, and we can easily argue that it exhibits perfect fairness independent of the time horizon one considers. In contrast, in the second trace (figure 4.14(b)) the fairness becomes dependent on the time horizon in which we look at the system. For instance, if we look at the entire sequence of packet arrival at node C, we realize both node A and B have sent the same amount of data (8 packets each) to node C and therefore the system is fair. Similarly, If the fairness horizon is set to 8, then in each contiguous sequence of 8 seconds, node C has indeed received an equal number of A's and B's packets and therefore the system is fair. However, if the fairness horizon is considered to be 4 seconds on times, then the trace of packet arrivals at node C lacks fairness, as either node A or B capture the channel for a period of 4 seconds causing the other node to starve during that time. Therefore, while over 8 seconds both connections can be considered stable, over the time horizon of 4, the connections appear unstable! To address this issue and measure the relative fairness in contention-based channels we use the sliding window method (SWM), proposed in [76]. SWM starts with a packet trace of channel accesses and slides a window of size W across it, as shown in figure 4.15 for a window of size 4 ($W = 4$).

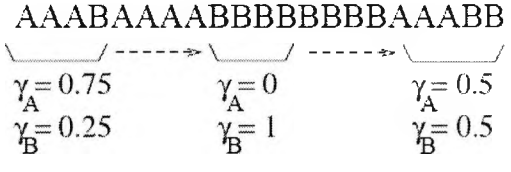


Figure 4.15: Illustration of the fairness sliding window

Here, the elements in the first window are the first 4 elements of the sequence, AAAB. We refer to the elements within a window as a snapshot. So as the window slides one element at a time, we obtain a series of snapshots, where for each snapshot the fractions of A's (γ_A) and B's (γ_B) is computed. Now, for each snapshot, the fairness within it can be measured using a per-snapshot Jain's fairness index [77] as defined below:

$$F_J = \frac{(\sum_{i=1}^{N} \gamma_i)^2}{N \sum_{i=1}^{N} (\gamma_i)^2} \tag{4.1}$$

After sliding the window through the entire sequence, we end up with a sequence of fairness values. We then calculate the average of all these values where this average corresponds to the fairness metric associated with window size W and the process is repeated with increasing window size. In this way, we will first have the big picture of the system fairness and also the relative fairness according to the wide range of time scales all in one graph.

4.3.3 Related Work

During recent years, many studies have addressed the TCP inter-flow instability and in particular the role of 802.11 backoff algorithm on TCP fairness in 802.11 based ad hoc networks. In an early work by Gerla et al [14], they investigated TCP fairness over different MAC protocols, namely FAMA [31], MACAW [32] and IEEE 802.11 [11]. In all the investigated scenarios, IEEE 802.11 always came top in terms of both throughput and fairness. In [78], the authors investigated TCP fairness over IEEE 802.11 MAC in ad hoc wireless networks and suggested the operation of 802.11 is the primary underlying reason that trigger TCP unfairness.

To improve 802.11 MAC fairness, [79] introduced Distributed Contention Control, where each station regularly computes a value called Slot Utilization, i.e. the ratio between the number of busy slots over the number of available slots during a period of time where this value reflects the level of usage of the radio channel. Whenever a backoff expires and a frame is ready to be transmitted, a transmission probability is computed according to the slot utilization value and to the number of unsuccessful previous transmission attempts. This additional contention level makes emitters restrain from transmitting whenever the medium becomes overloaded, preventing collisions and therefore enhancing the protocol performances.

By using a very similar concept, the study in [80] aims to improve fairness by releasing, in a probabilistic way, transmission opportunities granted by the standard backoff protocol. In particular, each station attempts to equalize the slot utilization it achieves with the slot utilization its neighbour stations get, such as to eliminate unfair channel allocation situations. To compensate for the reduction in total network throughout caused by refraining from transmitting, they have proposed a credit-based scheme that grants backoff-free transmission opportunities to the emitters as a reward for participating to the good operation of the network.

In [81], the authors proposed an algorithm called P-MAC where the contention window is determined dynamically, based on the time passed waiting for the channel, on the idle time and on the estimated number of stations in the contention area.

In Adaptive Transmission Control algorithm proposed in [82], each node estimates the number of active nodes within the contention range, as well as its

bandwidth share. The estimated number of active nodes is then used by a node to dynamically determine its fair share. Then based on the amount of deviation between actual and fair share, the node adjust its contention window. Similarly, in [83] each station continuously estimates its throughput and the aggregate throughput of the rest of stations it contends with. Then, the station calculates a fairness index, which determines by how much the station should adjust its contention window.

The authors in [84] propose Probabilistic NAV (PNAV), where according to a varying probability, a node waits for an extra duration after its NAV expires to access the channel after each transmission. This delay is a function of both the node and other nodes use of the medium and helps the hidden connection access the channel. It was shown this algorithm is very effective in order to give other nodes the possibility to gain access to the medium.

The authors in [85] propose a backoff scheme called Impatient Backoff Algorithm (IBA) where nodes decrease their contention window when their packet collide or are unable to send, thereby becoming more aggressive. On the other hand, nodes increase their contention window following each successful packet transmission. To stabilize the system due to frequent collisions, the algorithm is combined by resetting the mean contention window when it gets too low.

In [86], the authors proposed a scheme called Neighborhood RED, which is an extension of the RED [68] to ad hoc wireless networks. The authors show by detecting early congestion and dropping packets proportionally to a flows channel bandwidth usage, the NRED scheme is able to improve TCP fairness. In particular, in NRED each node keeps estimating the size of its neighborhood queue where a nodes neighbourhood consists of the node itself and the node's

which can interfere with this node's signals. Once the queue size exceeds a certain threshold, a drop probability is computed by using the algorithm from the original RED scheme. Thus, the NRED scheme is basically a distributed RED suitable for ad hoc wireless networks.

Finally using a different approach, the experiments discussed in [87], have illustrated the strong Signal to Noise Ratio (SNR) dependence of channel capture behavior with the IEEE 802.11 MAC protocol. A SNR differential as small as 5dB was shown to result in capture for the stronger connection.

4.4 Summary

In this chapter we considered the TCP instability by dividing the problem into two broad categories named as intra-flow and inter-flow instability.
It was first shown the intra-flow instability associated with the interaction of nodes belonging to the same TCP connection is primarily caused by the high number of contention drops in the link layer. The intra-flow instability was further divided into rerouting instability and out-of-order instability where the former is caused by the process of route failure and route discovery at the source while the latter happens due to frequent and random TCP packet losses in the network and the creation of out-of-order packets at the receiver. Using analysis, it was shown that the major contribution to intra-flow instability is created by out-of-order instability. In addition, the importance of controlling the amount of outstanding data in the network was highlighted as a simple and effective solution to alleviate the intra-flow problem.

In the second part of the chapter, we investigated the cause of inter-flow

instability in the presence of two or more connections that share the channel. It was first shown that channel capture serves as the primary reason behind TCP inter-flow instability and the operation of 802.11 binary exponential backoff algorithm was identified as the primary cause of channel access unfairness. To clarify the close link between fairness and inter-flow instability and therefore to quantify and compare the inter-flow instability in different scenarios, an analysis of fairness horizon was also given in this chapter.

Chapter 5

Cross Layer Contention Control

5.1 Introduction

Given the observations and explanations in chapter 4, it is clear that there are a number of different reasons behind TCP instability. In particular, it was concluded that the following events are the primary cause of TCP instability in 802.11 multihop ad hoc networks:

- Large numbers of packet contention drops caused by intra-flow interference
- TCP's early response to frequent packet reordering in the network
- Channel access unfairness caused by the 802.11 binary exponential backoff algorithm

This chapter presents detailed implementations of the proposed solutions to address each of the above issues separately. In essence, section 5.2 presents TCP ConTention Control (TCTC) that addresses TCP intra-flow instability by controlling the amount of outstanding data in the network. Section 5.3, introduces

Delayed Congestion Response with Extended Link-layer Retransmission (DCR-ELR) that aims to overcome the problem of TCP's high sensitivity to out of order packet delivery in multihop ad hoc networks. Finally, in section 5.4, the details of the Fair backoff Algorithm (FBA) that alleviates TCP inter-flow instability by modifying the backoff algorithm used in 802.11 MAC will be presented.

5.2 TCP Contention Control

5.2.1 Description

As discussed in 4.2.4, a high percentage of intra-flow interference and therefore contention drops can be eliminated by decreasing the amount of traffic in the network. However, this can be a very challenging task. On the one hand, if the amount of data in the network is reduced bellow the bandwidth delay product (BDP) of that flow, the channel resources are under-utilized. On the other hand, any increase above the BDP does not bring an additional performance enhancement, but only leads to increased buffering in intermediate nodes along the connection and other consequences as discussed earlier. Therefore, the main question in limiting the traffic load is how to properly set the amount of outstanding data in the network to achieve maximum throughput while minimizing the queueing delay experienced by individual packets.

In this section, this issue is addressed by introducing a cross layer solution called TCP ConTention Control (TCTC). In simple words, TCTC adjusts the TCP transmission rate to minimize the level of unnecessary contention in the intermediate nodes. To this aim, during fixed probe intervals, the TCP receiver monitors both the achieved throughput and the level of contention experienced by packets

during that interval[1]. Then, based on these observations, the receiver estimates the optimum amount of traffic maximize the throughput and minimize the contention delay for each connection. Finally, TCTC propagates the information back to the sender to adjust its transmission rate.
Using this information, the TCP sender now sets its transmission rate not merely based on the level of congestion in the network and the available buffer size at the receiver but also on the level of medium contention experienced by intermediate nodes. More precisely, while TCP congestion control adjusts the TCP transmission rate to avoid creating congestion in the intermediate network buffers, TCP contention control adjusts the TCP transmission rate to avoid creating queue buildup in the intermediate network buffers.

Since the key element in TCTC is optimum TCP flight size, the next subsection explains how TCTC estimates the optimum TCP flight size.

5.2.2 Optimum Load Estimation

To estimate the optimum amount of traffic that should be sent by the sender to get the maximum throughput while keeping the contention delay minimum, TCTC defines a new variable called *TCP Contention Window (ctwnd)*. The value of the *ctwnd* is determined according to the TCTC states as defined below:

- **Fast Probe**

  When a TCP connection is established, the TCTC enters the *Fast Probe* state where the *ctwnd* is increased exponentially. This is very similar to the TCP slow start phase in TCP congestion control where the TCP sender

[1] we will explain in section 5.2.3 how the contention delay is calculated by TCTC

probes the available bandwidth in a short time. The Fast Probe is also entered after the network is recovered back from Severe Contention state explained shortly.

- **Slow Probe**

  Slow probe is entered when the receiver realizes that both the achieved throughput and the packet contention delay have decreased compared to the last probe interval. In this situation, the receiver concludes the network is being under-utilized and tries to gradually increase the amount of newly injected data into the network by adding one MSS to *ctwnd* in every probe interval (additive increase)

- **Light Contention**

  If after changing the amount of injected data to the network, both the throughput and the level of packet contention delay are increased, the TCTC enters *Light Contention* state. In Light Contention state, the TCTC slowly decreases the *ctwnd* by one MSS per probe interval to control the amount of outstanding data in the network while avoiding unnecessary reduction in TCP throughput by implementing additive decrease. In other words, the Light contention state is entered when the network is in early states of overload.

- **Severe Contention**

  Severe Contention state is entered whenever the receiver sees an increase in the level of contention delay while the achieved throughput has been decreased. This situation is a clear sign of network overload since it shows the

push of more data into the network has just increased the amount of contention experienced by individual packets without increasing the throughput seen by the receiver. This situation can also happen if suddenly the level of contention in the network increases (e.g. a second connection starts using the intermediate nodes). To combat this, the TCTC sets its *ctwnd* to 2*MSS to force the sender to minimize its transmission rate.

The pseudo code in Algorithm 1, summarizes the calculation of *ctwnd* in these different phases.

Algorithm 1 PSEUDO CODE OF CALCULATING CTWND IN DIFFERENT STATES

```
1: if Δ_Throughput ≥ 1 then
2:     if Δ_Contention > 1 then
3:         TCP_Contention = TCP_Contention − MSS*MSS / TCP_Contention // Light Contention
4:     else
5:         TCP_Contention = TCP_Contention + MSS // Fast Probe
6:     end if
7: else
8:     if Δ_Contention > 1 then
9:         TCP_Contention = TCP_Contention + 2 * MSS // Severe Contention
10:    else
11:        TCP_Contention = TCP_Contention + MSS*MSS / TCP_Contention // Slow Probe
12:    end if
13: end if
14: if TCP_Contention ≤ 2 * MSS then
15:    TCP_Contention = 2 * MSS
16: end if
```

It is important to note that because of the TCP Delayed ACK algorithm [88], the minimum *ctwnd* in TCTC is set to 2*MSS to make sure at least 2 segments

are in the network and can trigger the transmission of a TCP ACK at the receiver without waiting for maximum ACK delay timer to expire.

As it can be seen in Algorithm 1, the condition for which the different states are entered is according to the value of two parameters named *Delta Throughput* ($\Delta_{Throughput}$) and *Delta Contention* $\Delta_{Contention}$.
$Delta_{Throughput}$, which is calculated according to formula 5.1, simply compares the amount of data received by the receiver (in Bytes) in the current probe interval (probe-new) and the last probe interval (probe-old)

$$\Delta_{Throughput} = \frac{(\text{data received})_{probe-new} * (probe - old)}{(\text{data received})_{probe-old} * (probe - new)} \tag{5.1}$$

$Delta_{Contention}$ on the other hand compares the average amount of contention delay experienced by all packets during the current probe interval with the average contention delay experienced by packets during the last probe interval. In the next subsection, we explain how $Delta_{Contention}$ acquires the required information on the contention delay values.

5.2.3 Contention Delay Measurement

As discussed earlier, the main objective of measuring contention delay is to reflect the current level of contention in the network. The first metric that is generally available and can be used to measure the level of contention is medium access delay. However, using medium access delay to measure contention delay can be misleading for two reasons:

1. Generally, medium access delay is recorded from the time a packet is inserted in the MAC layer buffer until the packet is sent to the physical layer for transmission. Therefore, it includes both the queueing and contention delays where the latter is from the time a node places the first fragment of that packet at the beginning of a buffer until the packet leaves the buffer for actual transmission on the physical layer. However, this can be problematic as the ratio between queueing and contention delay is unknown. To see how queueing delay can provide wrong information on current state of the contention, let us consider a snapshot (at different time) of a MAC buffer in an arbitrary node A shown in figure 5.1.

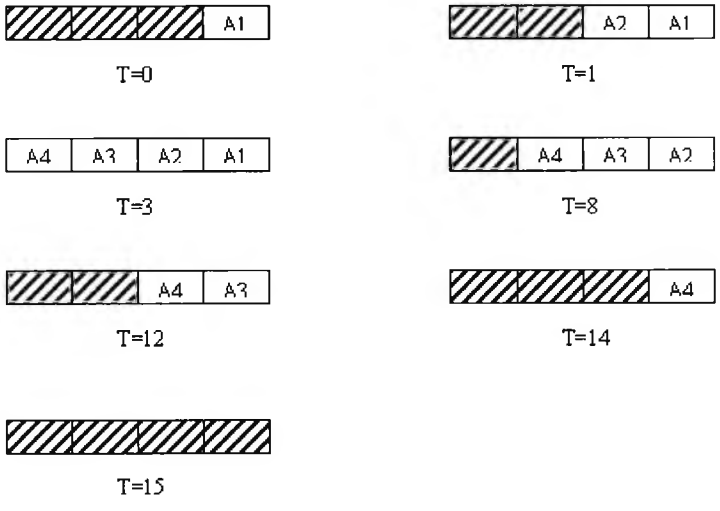


Figure 5.1: The impact of queueing and contention delay on medium access delay

At T=0, node A has one packet (A1) inside its buffer. At time T=1ms, packet A2 arrives in the buffer and at time T=3ms, packets A3 and A4 are added in the tail of node A's buffer. At T=8ms, node A finds the

chance to transmit packet A1 towards the data sink. This means A1 experienced a contention delay of 8ms (CD_{A1}=8ms) while its queueing delay is zero (QD_{A1}=0). At time T=12ms, A2 is also successfully transmitted which implies experiencing a contention delay of 4ms and a queueing delay of 7ms CD_{A2}=4ms, QD_{A2}=8-1=7ms. Using similar calculations, we get CD_{A3}=14-12=2ms, QD_{A3}=12-3=9ms, CD_{A4}=15-14=1ms, QD_{A4}=14-3=11ms. As shown in table 5.1, though the measured media access delay for all packets is more or less the same (8 to 11ms), the contention delay experienced by packet A1 is 8 times higher than A4 contention delay. In other words, while clearly the medium access contention around node A at time T=0 is much higher than at time T=14ms, the media access delay increases marginally at time T=14ms compared to time T=0.

Table 5.1: Summary of link layer delays in the example given in figure 5.1

Packet Number	Contention Delay (CD)	Queueing Delay (QD)	Media Access Delay
A1	8	0	8
A2	4	7	11
A3	2	9	11
A4	1	11	12

2. Medium access delay does not take into account the delay of retransmitted packets. This can provide misleading information on the level of contention since as in 802.11 MAC, acquiring the channel even after RTS/CTS reservation does not necessarily imply a successful transmission.

To address the above problems, TCTC uses contention delay as a primary measurement of the level of channel contention where the contention delay timer

inside each node only resets to zero when the node receives a MACK. In this manner the contention delay includes the period for the successful RTS/CTS exchange, if this exchange is used for the packet. In addition, the contention delay for a retransmitted packet will start from time the original packet was placed at the head of the buffer for the first time until the corresponding MACK is received. If after reaching the maximum retry limit, the packet cannot be transmitted, the value of contention delay is added to the contention delay of the next packet.

The value of measured contention delay is then inserted inside the Contention Delay Field (CDF) using the optional field in 802.11 MAC. More precisely, each packet records the contention delay it experienced in each node and adds the new contention delay to the CDF. In this manner, the Cumulative Contention Delay (CCD) experienced by each packet along the path is delivered to the final receiver (TCP receiver). The TCP receiver then calculates the value of Contention Delay per Hop (CDH) by dividing the CCD by total number of hops traversed by that specific packet. The main property of CDH is its independence from number of hops traversed by the packet. Finally the receiver derives the Mean Contention Delay per Hop (MCDH) by calculating the mean value of CDH received during each probe interval.

Having the value of MCDH, the $Delta_{Contention}$ as shown in equation 5.2 is derived by comparing the MCDH received by the receiver in current probe interval (probe-new) and the last probe interval (probe-old)

$$\Delta_{Contention} = \frac{MCDH_{probe-new}}{MCDH_{probe-old}} \tag{5.2}$$

5.2.4 Contention Delay Field

To carry the value of CDF inside a packet, we use the extended IEEE 802.11 MAC protocol packet format with optional fields inside the MAC header as suggested in [89]. In essence, a new field called "options" is proposed as a variable length field which extends standard MAC header. To perform separation of the data encapsulated into the frame from the MAC header, the option contains a Header Length field which specifies the entire length of the MAC header, including the list of options. In addition, each option consists of option type, length and data as shown in figure 5.2. The Length field is required to handle the case when a node does not support the corresponding option and therefore the knowledge of the option's length makes skipping the current option easier, jumping to the next one for processing.

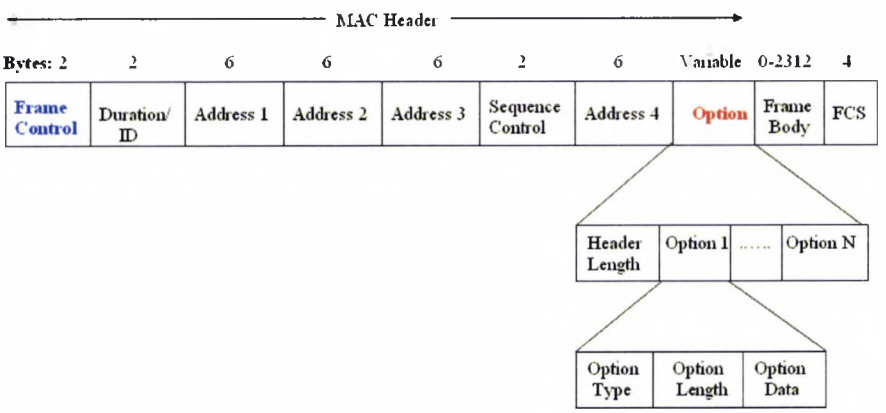


Figure 5.2: Options-enabled IEEE 802.11 data frame

Since in the proposed model, the only option being used is the CDF field, the 802.11 data packet supporting contention delay field is similar to figure 5.3 with the Frame Control value of 101000.

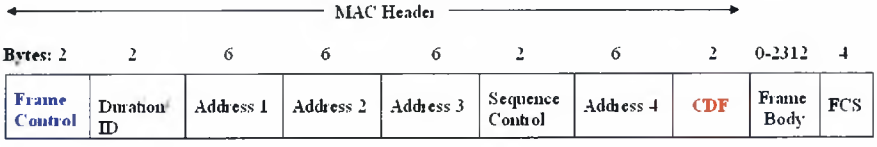


Figure 5.3: CDF-enabled IEEE 802.11 data frame

Note that here we have introduced a new type of data packet using the available reserved types in the Frame Control field of the MAC header of data frames. In particular, our frame "type" is equal to "10" (Data packet) and the "subtype" is set to "1000" as an indication of CDF-enabled data frame. This is to provide backward support within the existing IEEE 802.11 standard specification, since a CDF-enabled data frame should be of a different type with respect to a normal data frame.

5.2.5 Propagating the Contention Delay Information

Having calculated the optimum value of network overload over the next period of the probe interval, the next question is how to propagate this information (which is stored in *ctwnd*) back to the sender so the TCP sender can adjust its transmission rate accordingly. To answer that, let us recall from section 2.3.4 that the TCP sender cannot have a number of outstanding segments larger than the *rwnd* which is advertised by its own receiver. By default, the TCP receiver advertises its available receiving buffer size, in order to avoid saturation by a fast connection (flow control). We propose to extend the use of *rwnd* to accommodate the value of *ctwnd* in order to allow the receiver to limit the transmission rate of the TCP sender also when the path used by the connection exhibits a high contention and frame collision probability as well as its default flow control mechanism. In other

words, when TCTC is used, the new value of *rwnd* becomes the minimum of the available buffer size of the receiver (available receiver buffer) and the current value of *ctwnd* as shown below in equation 5.3.

$$rwnd = Min(\text{available receiver buffer}, ctwnd) \tag{5.3}$$

5.2.6 Choice of Probe Interval

It is clear from the above discussions that the choice of probe interval at the receiver can affect the performance of TCTC. Too large probe interval means that TCTC responds too slowly to contention changes in the network while too small probe interval will make TCTC sensitive to contention delay experienced by individual packets and therefore leads to *ctwnd* fluctuation. Heuristically, the probe interval of one RTT seems to be the natural and reasonable choice as it gives the receiver enough time to monitor the packets it received during one RTT and then sets its recommendation of sender's transmission rate (using *ctwnd*) for the next RTT. However, to provide the receiver with the correct and up to date information on the impact of the previous *ctwnd* on the level of network contention, the *ctwnd* should be updated in every other RTT. Therefore, we recommend the *ctwnd* gets updated every 2*RTT and remains fixed between two updates. In this way, the receiver can make sure the packets it has received are sent into the network after the sender has applied the changes imposed by the receiver in the last probe interval. Having the value of the TCTC probe interval, the other major issue is that the TCP receiver (which is running the TCTC algorithm) is unaware of connection's RTT[2]. This problem can be solved

[2]assuming the receiver is not sending any data packets towards TCP sender or there is a route asymmetry between sender and receiver

by using the fact that according to [90], in a small to medium size ad hoc networks the congestion window size (i.e. number of packets sent per RTT) does not grow beyond 5 packets. Therefore, considering this estimation together with the discussion given earlier, the probe interval in receiver can be defined as the period in which 2*5=10 packets are received. Note that although this is a rough estimation, it clearly solves our design objectives of probe interval which is merely to determine the frequency of updating TCTC. More details on the impact of probe interval on the performance of TCTC are given in chapter 7.

5.3 Delayed Congestion Response-Extended Link layer Retransmission

5.3.1 Description

As explained earlier in 4.2.3, due to the frequent packet contention losses in multihop ad hoc networks, packet reordering can happen quite often. Such frequent packet reordering can harm TCP performance in several ways [70,73,91]. First of all, it will cause the TCP sender to use fast retransmit to resend a data segment that was not lost, hence wasting bandwidth. Secondly, TCP assumes that loss is an indication of network congestion, and so a sender perceiving reordering as loss will also incorrectly reduce the data transmission rate. Thirdly, segment reordering causes interruptions to TCPs ACK clock, thereby causing its transmission to be more bursty. The above argument in addition with the results given in figure 4.7 lead us to two main conclusions:

- To avoid unnecessary packet retransmissions, the sensitivity of TCP to duplicate ACKs should be decreased in multihop ad hoc networks.
- The current 802.11 link layer retry limits appear to be too low in multihop ad hoc networks where nodes are not necessarily within the interference range of each other.

To first address the TCP sensitivity to duplicate ACKs, this section describes the Delayed Congestion Response (DCR) as was introduced in [92]. Then a modified version of DCR with Extra Link-layer Retransmission (DCR-ELR) is proposed to alleviate the TCP sensitivity to packet reordering while giving extra

opportunity for link layer local recovery.

5.3.2 Delayed Congestion Response

The basic idea behind Delayed Congestion Response (DCR) [92] is when both congestion and non congestion losses can occur, a simple solution would be to let the link layer mechanisms recover from the losses due to contention and the transport protocol to recover from the losses due to congestion. In other words, the DCR modifications aim to change the time at which the Fast Retransmit/Recovery algorithms are triggered. To this aim, the receipt of duplicate ACKs in TCP-DCR is assumed to be caused by non-congestion errors, for one RTT [92]. If the packet is recovered through the link layer retransmission mechanism before the end of this delay period (indicated by the receipt of a cumulative ACK acknowledging the lost packet), DCR proceeds as if the packet loss never occurred. However, if the packet is not recovered by link layer retransmission by the end of the delay period, DCR protocol triggers the congestion recovery algorithms of fast retransmission and recovery. By doing this, DCR effectively changes the paradigm that all losses are due to congestion to the paradigm that all losses are due to non-congestion related errors for a period of one RTT.

5.3.3 Extra Link layer Retransmission

The key factor in the efficiency of the DCR algorithm is to recover the non-congestion related data losses at the link layer while the congestion control is postponed. However, as shown in [56,93], the current 802.11 link layer retry limits

appear to be too low in multihop ad hoc networks and considerable number of non-congestion related losses are still recovered by TCP. To see the negative impact of TCP end-to-end recovery instead of 802.11 local recovery in ad hoc networks, consider figure 5.4 where node A is sending packets to node E. Let us assume,

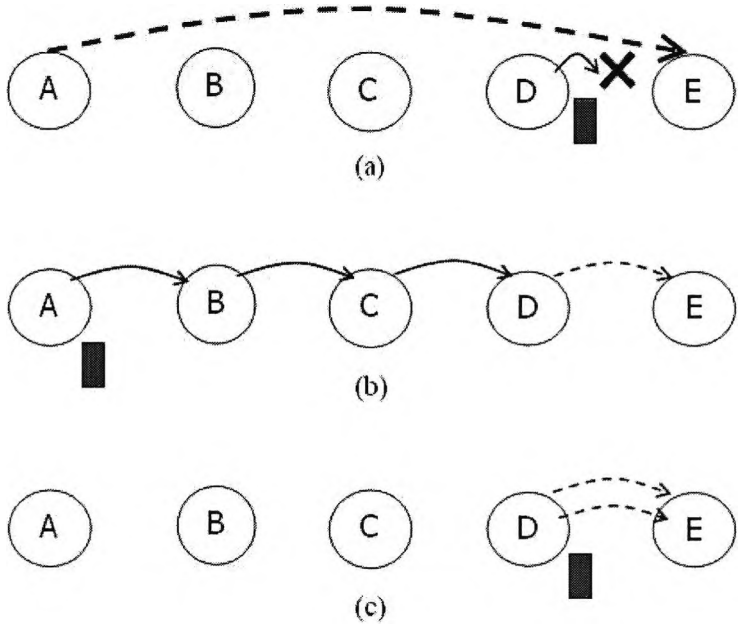


Figure 5.4: Comparison of the end-to-end and local recovery in ad hoc networks

node D cannot transmit its data after MAC-Retry-Limit (5.4(a)). In the default operation of 802.11, shown in 5.4(b), if the MAC cannot succeed in accessing the medium in limited tries at any hop, the packet should be retransmitted by TCP from the sender and start contending for the medium from the first hop again. This is obviously very inefficient as the packet has already used a considerable channel resources to get to node D and it is obviously a waste of channel resources to deliver the packet from node A to node D again. To tackle this problem, using the Extended Link layer Retransmission (ELR) approach shown in 5.4(c), after

node D runs out of its MAC_Retry_Limit, the node is given the chance to try for another MAC_Limit_Retry before giving up and dropping the packet. As it would be shown in chapter 7, the combination of DCR with ELR dramatically reduces the number of TCP's end-to-end recoveries and therefore improves the TCP stability.

5.4 Fair Backoff Algorithm

5.4.1 Description

In section 4.3.1 it was shown that channel access unfairness is the primary cause of TCP inter-flow instability. To tackle this issue, this section proposes a new scheme called Fair Backoff Algorithm (FBA) that aims to create a relatively simple and fair channel access algorithm between contenting nodes. To realize the design principles of FBA, let us recall that the main cause of channel access unfairness lies in the simple fact that the 802.11 backoff algorithm always acts in favour of the last successful node who accessed the channel. To address this problem, the binary exponential backoff algorithm is replaced with FBA that uses a dynamic access priority approach according to each node's perspective on the level of contention and its own channel access history. The main features of FBA are its simplicity, easy implementation and the fact that it does not require any control message exchange between competing nodes in 802.11 based ad hoc networks. Details of the FBA implementation are described further in the remainder of this section.

5.4.2 states

Since the Contention Window (CW) is the main parameter used in each node to access the channel, FBA changes the CW in each node according to three different states as shown in figure 5.5.

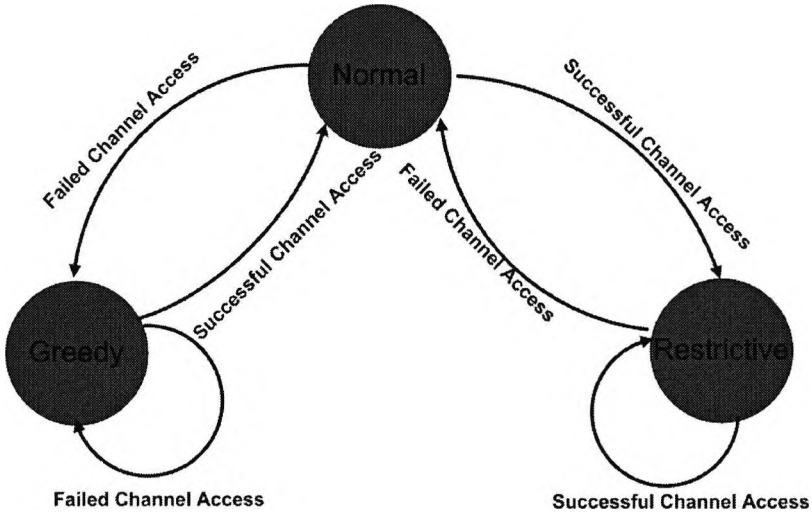


Figure 5.5: Different node states in the FBA scheme

- **Normal**

  This state is entered when the station

  - has data to send in its buffer
  - it has either recovered from a failed channel access or it has failed to gain the channel after a successful transmission.

  The main purpose of this state is to improve short-term fairness and thus stability by assigning relatively large CW to nodes in this state and therefore decreases the probability of a collision between contending nodes.

- **Restrictive**

  Following a successful channel access in the Normal state, the node enters a Restrictive state where the probability of node's channel access decreases according to the number of consecutive channel access events and increases with its buffer size. This state has been designed to force the successful

nodes to release the channel in favour of others while giving higher priority to the congested nodes compared to non-congested ones.

- **Greedy**

  If the node is unsuccessful after choosing its initial backoff within the Normal state, it will enter the Greedy state where the station takes high priority to access the channel by choosing relatively smaller contention window compared to successful stations. This state is included to prevent nodes from getting starved for a long period of time.

5.4.3 Contention Window Calculation

Having defined the possible states, now each node calculates its new CW according to the rule given in equation 5.4

$$CW = \begin{cases} CW_{min} \times Tradeoff\_co & Normal \\ CW_{min} \times [1 + Success \times (1 - Buffer\_co)] & Restrictive \\ CW_{min} \times Failed & Greedy \end{cases} \tag{5.4}$$

Here "Success" and "Failed" are the number of consecutive successful and consecutive failed channel access tries (and not necessarily failed transmission try) seen by each node, respectively. We have also defined two other variables called the "Tradeoff Coefficient" *(Tradeoff-co)* and "Buffer Coefficient" *(Buffer-co)* which are both critical to the performance of the algorithm.

Tradeoff-co is an integer number between 1 (i.e. $CW = CW_{min}$) and $\frac{CW_{max}}{CW_{min}}$ and it basically is designed to address the high probability of collisions between stations

that are in Normal state. To see the necessity of the Tradeoff-co, consider the situation when a node in restrictive state fails to access the channel and chooses at most $2*CW_{min}$ for its next retransmission. Meanwhile, a hidden node(s) in greedy state which has successfully gained the channel after unsuccessful transmissions chooses CW_{min} for its next packet transmission. As shown in [75], with such low contention window it is very likely that the transmission from these two nodes will collide and both stations will enter the Greedy state. Tradeoff-co aims to prevent such collisions in Normal state by assigning relatively large contention windows to such nodes. This will firstly result in smaller number of packet collisions, and secondly will improve short-term fairness as it gives immediate equal opportunity to nodes coming from different states regardless of their prior state. On the other hand, a large value of Tradeoff-co will result in a greater number of idle slots in the channel which obviously degrades the achieved throughput. Therefore, the value of Tradeoff-co can be seen as the design tradeoff between the achieved fairness and the throughput. More information on the impact of Tradeoff-co on system performance is given in Chapter 7 where the simulation results are presented and discussed.

The next parameter is Buffer-co which is a number between 0 and 1 and gives higher priority to nodes who are situated in a more congested area of the network. Another design goal behind Buffer-co is to control the unnecessary increase of the contention window in scenarios where only two nodes are communicating with each other. In this scenario, without the presence of Buffer-co, the successful node will continuously go to Restrictive state as it is the only node to send. However, Buffer-co will compensate by decreasing the contention window as number of packets inside the sender buffer increases, hence limiting the long idle period

between successive transmissions.

Therefore, the Buffer-co can be described as:

$$Buffer - co = \begin{cases} 0 & Buffer \leq Thresh_{min} \\ \frac{Buffer - Thresh_{min}}{Thresh_{max} - Thresh_{min}} & Thresh_{min} \leq Buffer \leq Thresh_{max} \\ 1 & Buffer \geq Thresh_{max} \end{cases} \tag{5.5}$$

where Buffer is the current number of packets inside the MAC buffer; Thresh_{max} and Thresh_{min} are chosen to be 20% and 80% of the maximum buffer size, respectively. Note that the value of Buffer-co parameters are chosen similar to the RED [68] active queue management scheme.

5.5 Summary

In this chapter, we proposed three separate solutions called TCTC, DCR-ELR, and FBA to address TCP instability problems as outlined in chapter 4. While TCTC and FBA were primarily designed to tackle TCP intra-flow and inter-flow instability, respectively, DCR-ELR aimed to reduce the number of TCP end-to-end packet loss recoveries.

The main idea behind TCTC was to adjust the TCP transmission rate to minimize the level of unnecessary contention in the intermediate nodes without degrading end-to-end throughput. This was done at the TCP receiver and by comparing the achieved throughput and the level of contention delay experienced by packets during consecutive probe intervals. After estimating the optimum transmission rate, the receiver propagates back the information to TCP sender through its receiver window option which is naturally used for flow control.

In the second proposal, DCR-ELR, the Delayed Congestion Response algorithm was combined with Extended Link layer Retransmission to first combat the TCP sensitivity to a large number of out of order packets (through DCR) and secondly to perform contention loss recovery in the link layer rather than in the transport layer (through ELR).

Finally, in last section, the FBA algorithm is proposed to replace the 802.11 BEB in order to tackle the channel access unfairness seen by contending nodes.

Chapter 6

Simulation Modelling

6.1 Introduction

To measure and evaluate the efficiency of the proposed algorithms, simulation was used in this research. This choice was done for two main reasons. Firstly, the distributed nature of ad hoc networks together with the fact that the behaviour of a node is dependent not only on its neighbours' behaviour, but also on the behaviour of other unseen nodes makes the analysis of multi-hop network extremely difficult [94, 95]. In particular, as shown in Chapter 3, deriving an analytical model with reasonable degree of accuracy (e.g. considering hidden terminal problem and interaction between TCP and 802.11) includes numerous variables and constraints whereby the level of complexity rises rapidly with respect to the number of variables taken into account. Secondly, the choice of a real testbed implementation was ruled out mainly because most of the off-the-shelf commercial wireless cards (including the wireless cards used in our testbed) do not provide the users with full control over the card configuration and some

options. On the other hand, in order to implement and validate the proposed algorithms discussed in chapter 5, there was a clear need to change the card settings. For all these reasons, simulation was the best available option remained to implement and evaluate the proposed schemes. On the other hand, to validate the accuracy of the simulator, a customized testbed was built and used in this study which would be discussed later in the chapter.

6.2 Simulation Environment

In order to conduct simulations, the OPNET (OPtimized Network Engineering Tool) [61] simulator has been used in this work. OPNET was chosen primarily because it was a proven simulation tool utilized in several previous studies in ad hoc networks (e.g. [18,29,96]) and provides a comprehensive set of simulation and modelling products for the development and performance analysis of TCP over 802.11 wireless ad hoc networks. In addition to providing a rich library of models for implementations, OPNET is an open model source code with integrated debugging and analysis which enabled us to trace the packets and events across different layers. In essence, by integrating custom codes into OPNET Debugger (ODB), special care could be taken to determine whether the implementation of the new algorithms would function as designed and that the system would not exhibit unwanted side-effects.

6.2.1 OPNET Modeler

OPNET Modeler[1] uses an object-oriented and discrete-event environment for the development of models and simulation scenarios. Simulation models in OPNET Modeler are organized in a hierarchy consisting of three main levels in which each level of hierarchy represents different aspects of the complete model being simulated: the project editor, node editor and process editors (Figure 6.1).

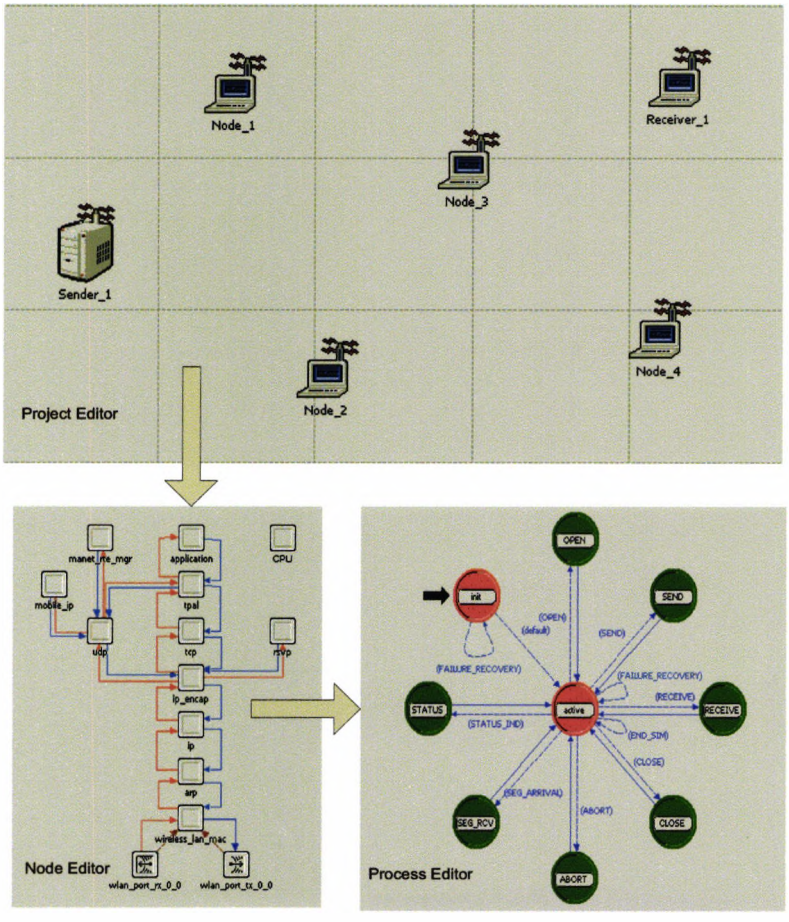


Figure 6.1: OPNET simulator hierarchy

[1]The OPNET Modeler 11.5 was used in this work

The project editor refers to the simulation scenario or simulation network. It defines the network layout, the nodes and the configuration of attributes of the nodes comprising the scenario. The node models are at the second level in the hierarchy and consist of an organized set of modules describing the various functions of the node. The modules in the nodes are implemented using process models, the lowest level in the hierarchy. Process models consist of finite state machines, definitions of model functions, and a process interface that defines the parameters for interfacing with other process models and configuring attributes.

6.2.2 Assumptions and Parameters

In the simulations performed throughout this dissertation, the following assumptions were used:

- Nodes have sufficient power to function and the number of nodes in a given topology remains constant throughout the simulation time. At no time does a node run out of power or malfunction because of lack of power. Equivalently, the wireless transceivers are active at all times.
- Transmissions are not affected by random errors. Transmissions may still interfere with each other; however a node will always successfully decode a transmission provided it is within transmission range of the source and there is no interfering transmission.
- All nodes are equipped with IEEE 802.11 transceivers and unless otherwise stated the RTS/CTS mechanism for packets larger than RTS-Threshold is employed on these wireless devices.

- The signal propagation model used is the Two-Ray Ground model [97] where signals propagate from sender to receiver in an open environment
- The simulation area is defined to be 1000x1000m where the transmission range of each node is confined to 100 meters based on the testbed implementation results presented in [90].
- Nodes are assumed to be static and the topology is fixed. The main reason behind this assumption is to isolate the packet losses and avoid unwanted routing phenomena such as network partitioning. Note that this assumption in any way does not invalidate the results shown in the thesis as the primary focus in this work is to investigate the impact of link layer parameters on transport layer and vice versa regardless of node mobility.

In addition to the above assumptions, the detailed default simulation parameters and settings for TCP, routing and wireless LAN are shown in figure 6.2, 6.3, and 6.4, respectively.

TCP Parameters	(...)
├Version/Flavor	Unspecified
├Maximum Segment Size (bytes)	1460
├Receive Buffer (bytes)	65535
├Receive Buffer Adjustment	None
├Receive Buffer Usage Threshold (of RCV BUFF)	0.0
├Delayed ACK Mechanism	Segment/Clock Based
├Maximum ACK Delay (sec)	0.200
├Maximum ACK Segments	2
├Slow-Start Initial Count (MSS)	1
├Fast Retransmit	Enabled
├Duplicate ACK Threshold	3
├Fast Recovery	New Reno
├Window Scaling	Disabled
├Selective ACK (SACK)	Enabled
├ECN Capability	Disabled
├Segment Send Threshold	Byte Boundary
├Active Connection Threshold	Unlimited
├Nagle Algorithm	Disabled
├Karn's Algorithm	Enabled
⊞ Timestamp	Disabled
├Initial Sequence Number	0
⊞ Retransmission Thresholds	Attempts Based
├Initial RTO (sec)	3.0
├Minimum RTO (sec)	1.0
├Maximum RTO (sec)	64
├RTT Gain	0.125
├Deviation Gain	0.25
├RTT Deviation Coefficient	4.0
├Timer Granularity (sec)	0.5
├Persistence Timeout (sec)	1.0

Figure 6.2: TCP settings used in the simulation

⊟ DSR Parameters	( )
⊟ Route Cache Parameters	(...)
├Max Cached Routes	Infinity
├Route Expiry Timer (seconds)	300
⊞ Route Cache Export	Do Not Export
⊟ Send Buffer Parameters	(...)
├Max Buffer Size (packets)	Infinity
└Expiry Timer (seconds)	30
⊟ Route Discovery Parameters	(...)
├Request Table Size (nodes)	64
├Maximum Request Table Identifiers (identifiers)	16
├Maximum Request Retransmissions (retransmissions)	16
├Maximum Request Period (seconds)	10
├Initial Request Period (seconds)	0.5
├Non Propagating Request Timer (seconds)	0.03
└Gratuitous Route Reply Timer (seconds)	1
⊟ Route Maintenance Parameters	(...)
├Maximum Buffer Size (packets)	50
├Maintenance Holdoff Time (seconds)	0.25
├Maximum Maintenance Retransmissions (retransmissions)	2
└Maintenance Acknowledgement Timer (seconds)	0.5
├DSR Routes Export	Do Not Export
├Route Replies using Cached Routes	Enabled
├Packet Salvaging	Enabled
├Non Propagating Request	Disabled
└Broadcast Jitter (seconds)	uniform (0, 0.01)

Figure 6.3: DSR settings used in the simulation

⊟ Wireless LAN Parameters	(...)
├BSS Identifier	Auto Assigned
├Access Point Functionality	Disabled
├Physical Characteristics	Direct Sequence
├Data Rate (bps)	2 Mbps
⊟ Channel Settings	(...)
├Bandwidth (MHz)	Physical Technology Dependent
└Min Frequency (MHz)	BSS Based
├Transmit Power (W)	0.005
├Packet Reception-Power Thre...	-95
├Rts Threshold (bytes)	512
├Fragmentation Threshold (bytes)	None
├CTS-to-self Option	Disabled
├Short Retry Limit	7
├Long Retry Limit	4
├AP Beacon Interval (secs)	0.02
├Max Receive Lifetime (secs)	0.5
├Buffer Size (bits)	256000
├Roaming Capability	Disabled
├Large Packet Processing	Drop
⊞ PCF Parameters	Disabled

Figure 6.4: 802.11 settings used in the simulation

For the sake of clarity, some of the simulation parameters are discussed in the following text.

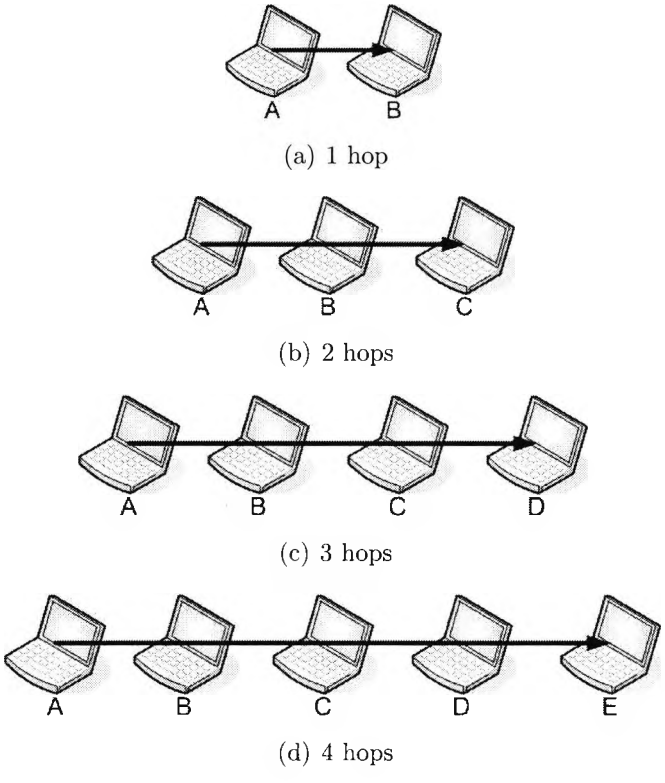
To minimize the possibility of simulation stochastic nature errors, each simulation set is repeated four times, each with a different random seed [2]. For scenarios where the system instantaneous behaviour is of interest (e.g. packet delay, queue size, etc.), the simulation set with the minimum TCP throughput deviation is chosen and presented as a result. Otherwise, the simulation results are averaged over the four sets and presented.

In each set, a TCP connection is set between two selected nodes to facilitate a FTP transfer session for the duration of the simulation. The overall simulation time is set to 600 seconds where in the first 50 seconds, the results are not taken into account to eliminate the initial bias effects (e.g. route setup, connection establishment, etc.). The simulation time, set to 10 minutes, is chosen in order to examine TCP performance over bulk file transfers application. This is mainly because the focus of this thesis is the behaviour of TCP when the network is close to its saturation and the full spectrum of TCP congestion control mechanisms is utilized over substantial time periods. Transferring small files such as web pages is too short that may only activate the slow start mechanism and is not adequate to trigger TCP instability.

[2] the repetition of 4 experiments were chosen as the difference in data results variation was negligible

6.3 Simulation Validation

In order to validate the simulation tool used and to ensure that the simulation results are fairly representative of real world systems, we setup a small multihop ad hoc network testbed consisting of 5 nodes. Figure 6.5 shows a logical view of the different scenarios implemented in the testbed.



(a) 1 hop

(b) 2 hops

(c) 3 hops

(d) 4 hops

Figure 6.5: Testbed topologies

The nodes were positioned so that they have only one upstream and one downstream neighbour. For instance, in figure 6.5, node A can only transmit and listen to node B. Node B has nodes A and C within its radio range and so on. This setup is necessary to ensure that routes would be using similar links in every test. The wireless connection between the nodes were verified by monitoring the

ping results. A good connection was identified when there were few or no packet losses. To verify that multihop communications are possible, a ping operation with -R option was performed from the TCP sender to the TCP receiver and vice versa.

6.3.1 System Components

In the testbed, five different nodes were used in which three of them were laptops and the two other were desktop computers. To make sure the hardware differences (e.g. CPU speed, memory size, USB speed, etc.) between the nodes do not impose any restriction on system performance, extra attention was oaid. In particular, all nodes were using the same wireless interface (802.11g US Robotic USR5422 USB key [98]) and all nodes in the testbed were running the Linux-Suse 10.1 with kernel 2.6.17. The wireless cards were using the DSSS physical layer operating at the nominal bit rate of 11Mbps. To support the wireless card drivers in Linux, NDISwrapper version 1.23 was used [99]. The benefit of NDISwrapper is that it uses the Windows NDIS (Network Driver Interface Specification) driver for wireless cards and allow the Windows driver of wireless card driver to be used under Linux.

6.3.2 Testbed Environment

All experiments were performed in an indoor environment. To be able to compare the simulation and testbed results, the same parameters discussed in 6.2.2 were applied with the exception that the data rate 11 Mbps was chosen for the simulation to match the nominal speed of the wireless cards.

One major difference that existed between the settings in our testbed and simulation was the use of static routing instead of DSR reactive routing in the testbed. This choice was made due to enormous number of link breakages and instabilities encountered when the DSR-UU [100] implementation was used in the testbed as a routing algorithm candidate.

The antenna of the wireless cards were covered with foil shielding tape to reduce their transmission and communication range. This works since copper changes the effective antenna impedance, thus inducing an impedance mismatch with the card circuitry resulting in less power being delivered on the air interface. This configuration was mainly done to accomplish repeatable tests by placing the nodes in a close geographical locations where all nodes were subjected to similar source of errors. It is important to note that according to the results conducted in [101], covering the cards does not have any observable effect on the cards other than reducing their range (and causing them to heat up a little faster).
The method used for the experimentations, was a file transfer between two selected nodes over a certain amount of time (200 seconds), on network topologies shown earlier in figure 6.5.
Finally, to minimize transmission errors caused by interference from other IEEE 802.11 devices (from other offices that cannot be turned off) and people movement around the building, the tests were performed during late evening hours. We repeated each experiment four times and averaged the collected results.

6.3.3 Experimental Results and Data Gathering

In order to validate the simulation results, several experiments were performed and a number of measurements were taken using TCPdump [102], TCPtrace [103],

Web100 [104], and Ethereal [105]. In particular, the main parameter of interest was accumulated TCP goodput with and without the use of RTS/CTS. Figure 6.6 depicts the overall goodput achieved in both simulation and testbed over a series of multihop chain topology.

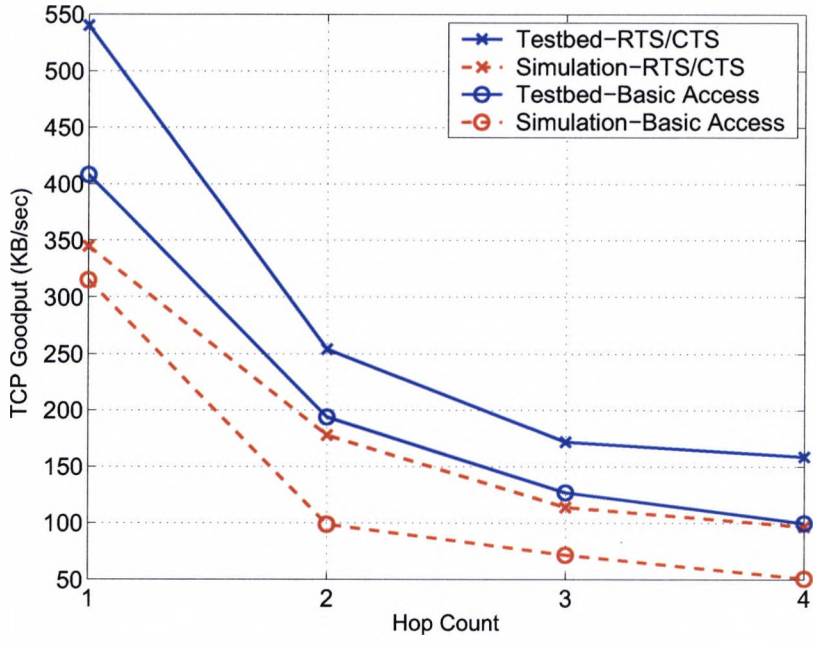


Figure 6.6: Comparison of testbed (with static routing) and simulation (with dynamic routing) results

Despite the results in figure 6.6 suggesting there is a non-negligible gap between simulation and testbed results, the goodput drop pattern in both scenarios closely match. In particular, in both scenarios the use of RTS/CTS improves the TCP goodput and the main TCP goodput drop occurs between the 1-hop and 2-hops topologies. It is very important to note that such a non-negligible gap between testbed and simulation results can be easily explained by recalling the fact that our testbed benefits from static routing while DSR routing protocol has been used in deriving the simulation results. To better understand the impact

of routing protocol on system performance, figure 6.7 shows the TCP aggregated goodput when static routing is used in both the simulation and the testbed.

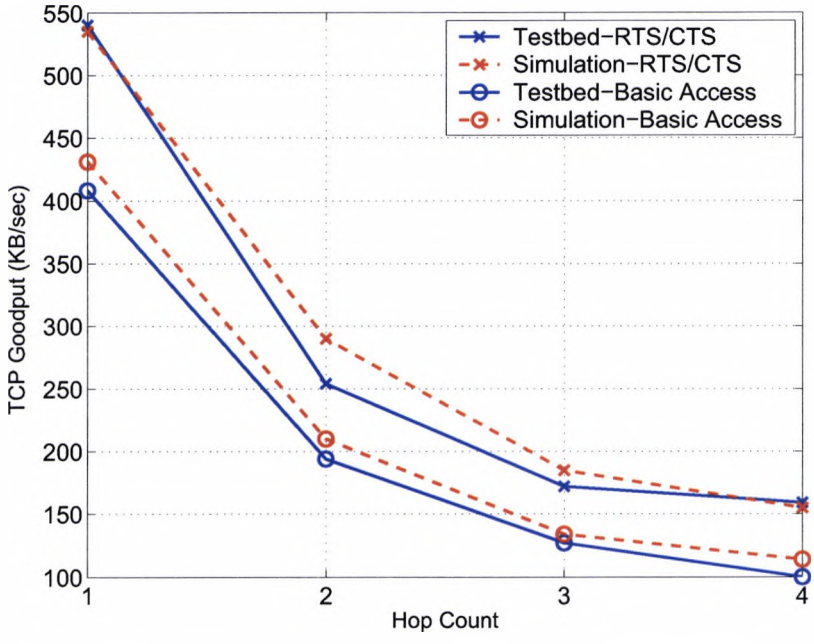


Figure 6.7: Comparison of testbed and simulation results both under static routing

The results are in fact quite promising as they show only a marginal gap between the results obtained from simulation and testbed when the same routing settings are applied.

Therefore, we can summarize the simulations performed in the OPNET can closely imitate the behaviour of TCP and 802.11 protocols in real implementations and thus can be used as a reasonably trusted means of performance investigation in the next chapter.

6.4 Summary

In this chapter, we mainly described the methodology used to evaluate and validate the work conducted in the thesis. We briefly explained why OPNET was chosen as the simulation tool. Then the accuracy of the simulator was compared against the results obtained from a real testbed implementation. Although some differences between the numerical results gathered from these two tools were observed in different scenarios, the pattern in measured information matched quite closely with each other. In other words, despite the different channel characteristics and assumptions in the simulation with the real world implementation, such that the results did not exactly match, the OPNET simulator closely followed the standard protocol behaviour implemented in existing off-the-shelf wireless cards. Therefore, this observation in addition to information obtained from ODB provides evidence for the reliability of the results obtained from simulations in the next chapter.

Chapter 7

Simulation Results and Analysis

7.1 Introduction

After defining the set of cross layer solutions in Chapter 5 and the simulation model in Chapter 6, this chapter presents a study of the simulation results. In order to fully study the proposed schemes[1], a number of simulations with different topologies and parameters have been conducted in this study. In particular, to evaluate the efficiency of the proposed algorithms on intra-flow instability, a set of chain topologies with varying number of hops and network conditions has been used in section 7.2. In section 7.3, a set of cross topologies with different number of hops has been used to measure the effectiveness of the proposed schemes in tackling the inter-flow instability problem. Finally, to investigate the impact of the proposed algorithm on both intra-flow and inter-flow instability, section 7.4 presents the results in various mesh topologies.

[1] In this chapter, the term "default algorithms" or for short "default" refers to the case where the standard version of 802.11 MAC and TCP protocols with parameters given in section 6.2.2 are used. On the other hand, the term "proposed algorithms" or for short "proposed" refers to modified 802.11 MAC and TCP protocols with changes proposed in chapter 5

Before starting to explain the simulation results, it is important to first discuss the metrics that have been used to analyze the performance of the proposed schemes against the default algorithms. To this aim and to fully investigate different aspects of system performance, a wide range of metrics across different layers have been chosen to accurately evaluate the performance of the proposed algorithms. In particular, the following metrics have been considered:

- Goodput: The goodput of each TCP connections is defined as the number of bytes of the TCP data connection correctly delivered to the receiver, such that byte and all previous bytes of the stream were delivered with no missing TCP segments. The goodput result is in of particular interest as it is one of the primary metrics to evaluate TCP performance.

- TCP segment delay: TCP segment delay is measured from the time a TCP segment is submitted to the IP layer in the sender until its corresponding ACK is received. To see the importance of this metric, let us recall that in general, TCP instability refers to the situation where a TCP receiver does not receive the TCP packets from the sender in a timely manner. Therefore in non-varying channel conditions, TCP segment delay can be used as a measurement of TCP stability across the network

- Number of TCP retransmissions: In an error-free channel, TCP retransmissions (caused either by fast retransmit or timeout) can be used as an indication of TCP ability to control congestion in the network and eliminate unnecessary TCP retransmission due to contention related packet loss.

- Fairness index: The fairness index is calculated using the fairness window technique described in section 4.3.2. More precisely, the fairness window

mechanism starts with a trace of channel accesses and slides a window of size w seconds across the packets arrived from different connections and compute the fairness for every window. After sliding the window through the entire sequence we end up with a sequence of fairness values. Then, we calculate its average where this average corresponds to the fairness metric associated with window size of w seconds. The process is repeated with increasing window sizes, and then the average fairness values versus the window size are plotted. A sliding window method is useful because in this way both short term and long term fairness can be illustrated together. To calculate the fairness of a given window size, w, the ratios of packet arrivals from each connection over that window are calculated. Let γ_i be the fraction of packets from connection i that arrived during the window and N be the total number of connections competing for network resources. Finally, by using the Jain's fairness index [77], we have:

$$F_J = \frac{(\sum_{i=1}^{N} \gamma_i)^2}{N \sum_{i=1}^{N} (\gamma_i)^2} \tag{7.1}$$

where absolute fairness is achieved when $F_J = 1$ and absolute unfairness is achieved when $F_J = 1/N$.

- Average number of buffered packets and queueing delay: These two link layer metrics can be used interchangeably to indicate the level of congestion in the network when are compared across different scenarios.

- Average backoff slots: This statistic is the average time a node has to wait before accessing the channel and it can be used as a metric to indicate the amount of time a channel is "*wasted*" in an idle state.

- Link layer packet drop: The link layer packet drop is basically the amount of higher layer packets that have been dropped in the link layer either because of buffer overflow or exceeding the 802.11 retry limits. This metric can be quite useful as it can shed light on the actual cause of higher layer packet drops in the link layer.

- Average Link layer Attempt (ALA): As the name suggests, Average Link layer Attempt shows the average number of attempts to transmit a link layer packet successfully to next hop and is calculated as follows:

$$ALA = \frac{\sum_{i=1}^{N}\sum_{j=1}^{C}(\text{TransmittedPackets})_{i,j}}{\sum_{i=1}^{N}\sum_{j=1}^{C}(\text{SuccessfullyTransmittedPackets})_{i,j}} \tag{7.2}$$

  Here, C is the total number of connections and N is the total number of nodes in each connection.

  There are a number of situations where ALA can be useful. For instance, it can be used to measure the effectiveness of the medium access algorithm in collision avoidance. Also, it can be used as a power consumption metric where a smaller ALA suggests less amount of energy is consumed to transmit one packet.

To eliminate the repetition of similar results, in the rest of this chapter and according to the specific scenarios, only some of the above metrics would be used to measure the system performance across the network.

7.2 Chain Topology

In this section, the results under varying number of hops in a single flow chain topology are presented. The importance of analyzing the results in a chain topology is that it enables us to investigate different aspects of intra-flow instability in isolation from inter-flow instability.

First, let us review figure 7.1 which shows the underlying cause of TCP packet drops in the link layer under varying number of hops in a chain topology. The importance of this figure is to determine the roots of TCP retransmissions triggered from the link layer.

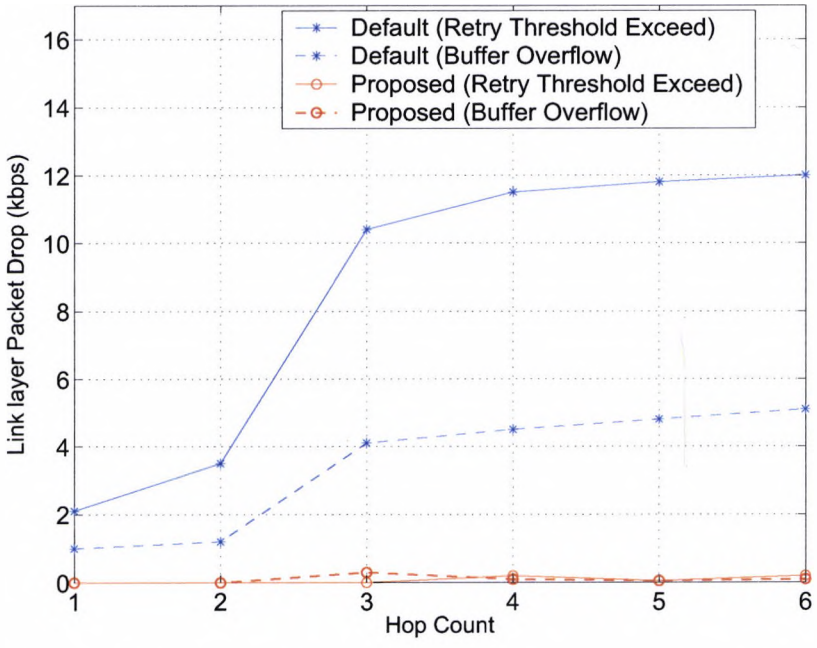


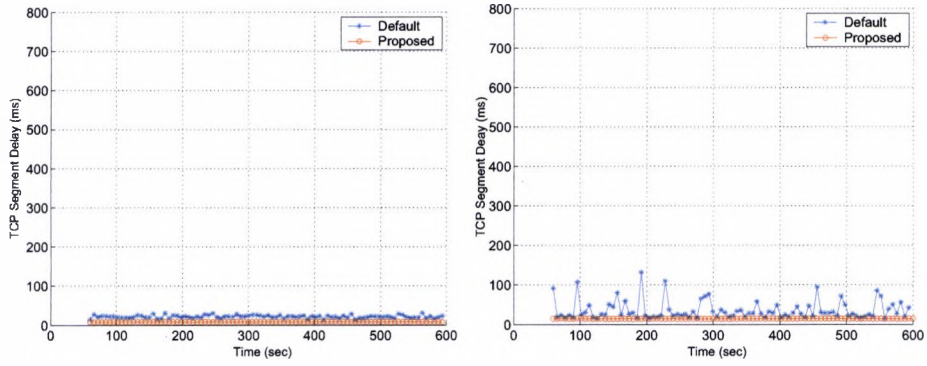
Figure 7.1: The nature of higher layer drops in a chain topology

As it can be seen, a majority of higher layer packets are dropped because of excessive contention between competing nodes in the default algorithm. This

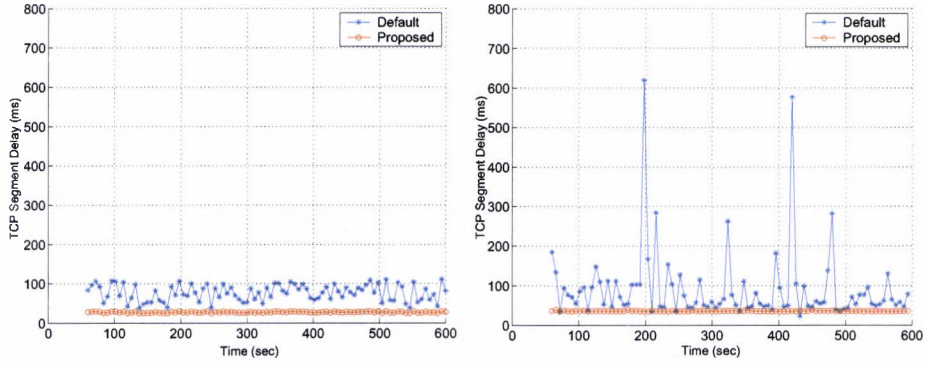
clearly confirms the discussion given in subsection 4.2.2 on the impact of intra-flow packet interference on TCP performance. Meanwhile, figure 7.1 suggests that still a considerable number of packets are dropped because of congestion in the network as a result of large number of outstanding data in the network. These two issues are clearly addressed using the proposed algorithm where the TCTC almost reduces the TCP packet drops to zero by limiting TCP flight size and therefore controlling both congestion and contention in the network. In addition, most of the packet drops are recovered locally in the link layer through the use of the DCR-ELR algorithm.

One issue that needs to be addressed in figure 7.1 is the sharp increase of packet drops due to contention and buffer overflow in a 3-hop chain. The increase in the contention related drops can be explained by the introduction of hidden terminals in the forward path in a 3-hop chain in contrast to 1-hop and 2-hops scenarios. For the increase in buffer overflow related packet drops, we believe the increase is mainly due to the channel contention reuse feature of the 802.11 which is 3 hops as described in section 4.2.2. In particular, since in a 3-hop chain topology, there can be only one transmission at a time, packets are kept in buffers for longer period of time and therefore there is a higher possibility of packet drop due to buffer overflow.

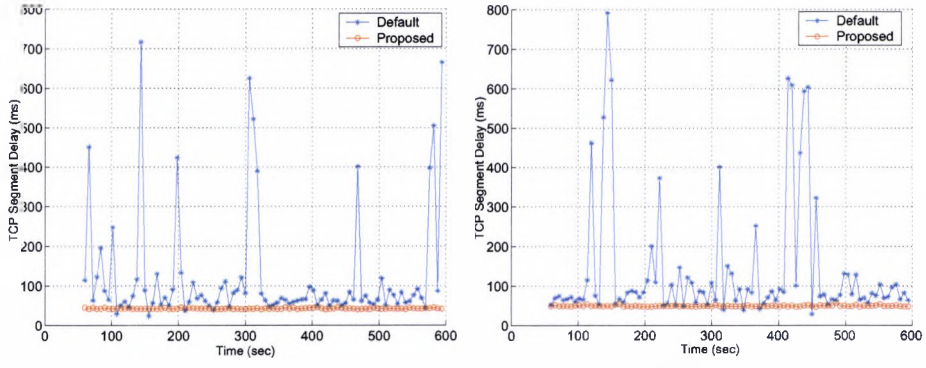
To evaluate the TCP stability, let us consider the results of the TCP segment delay shown in 7.2.



(a) 1 hop (b) 2 hop



(c) 3 hop (d) 4 hop



(e) 5 hop (f) 6 hop

Figure 7.2: TCP segment delay in a chain topology

As it is obvious from the results shown in figure 7.2, TCP segments in the proposed scheme experience a lower and less fluctuating delay in contrast to the default algorithm. More specifically, while using the proposed scheme the TCP segments delay fluctuation is on average 3% in each scenario, the figure rises up to 44% when the default algorithm is employed. Furthermore, while the TCP segment delay is bounded between 8 to 40 ms depending on number of hops traversed by the packet in the proposed algorithm, the TCP packets segment delay can be as high as 800ms in the default algorithm which implies a high sensitivity of packet delay to hop counts. It is also interesting to note that the TCP instability in the default algorithm becomes more serious, as the number of hops increases.

While the above results confirm the improvement of the TCP stability when the proposed schemes are deployed, it is very important to make sure that such stability has not been achieved in the cost of compromising TCP goodput. To this aim, figure 7.3 presents the aggregated TCP goodput achieved across different number of hops.

The results are very promising as it shows there is on average 34% goodput improvement in all scenarios when the proposed schemes are applied in comparison to using default protocols.

Finally, to evaluate the efficiency of the proposed algorithm in controlling the level of congestion and unnecessary contention in the network, let us consider the number of TCP retransmissions for both schemes as shown in figure 7.4.

The results show that in comparison to the default algorithm, the proposed scheme has dramatically reduced the number of TCP retransmissions by controlling the amount of outstanding data in the network. In addition, the dramatic

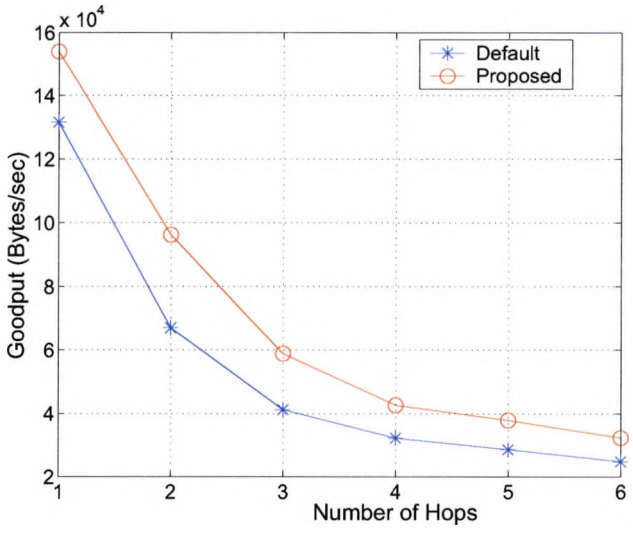


Figure 7.3: TCP goodput in a chain topology

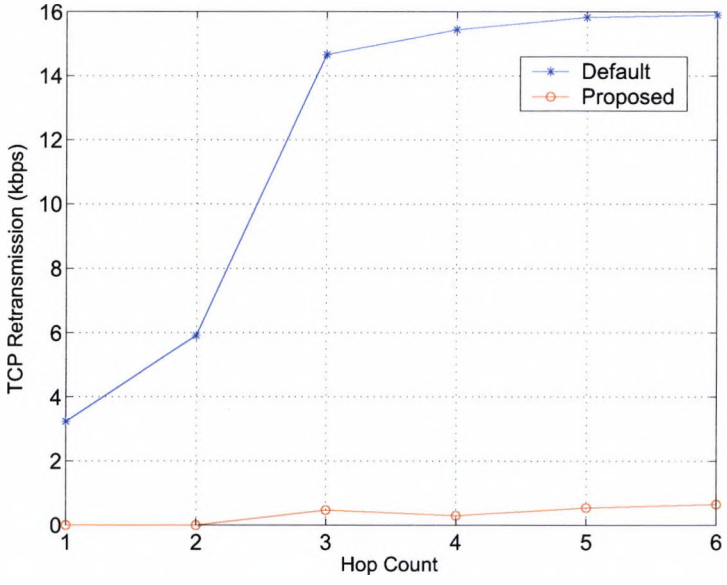
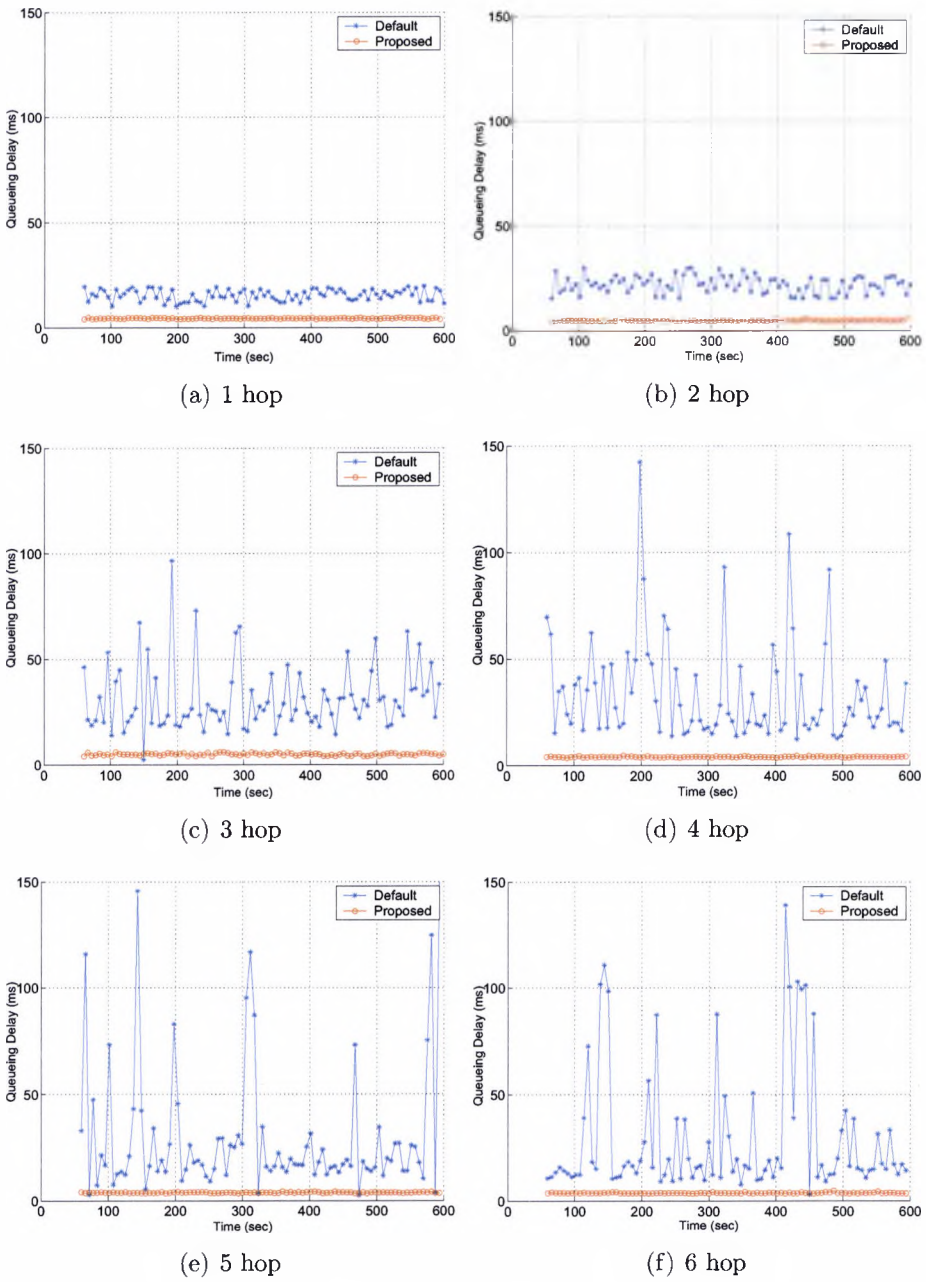


Figure 7.4: Average number of TCP retransmissions in a chain topology

cut in the number of TCP retransmissions can be traced back to DCR-ELR. This is because as DCR postpones TCP's reaction to large number of out-of-order packet deliveries, ELR recovers most of the failed link layer packet drops locally

and therefore only a very small number of end-to-end recoveries are triggered from TCP.
It is important to note that the reduction in TCP retransmissions without any reduction in TCP goodput clearly demonstrates the efficiency of the proposed algorithm in utilizing channel resources and tuning TCP sender transmission rate close to its optimum value.

In addition to transport layer metrics, a number of measurements have also been performed in the link layer to evaluate the performance of the proposed algorithms in particular from the power efficiency point of view. As the first link layer metric, figure 7.5 shows the average time in milliseconds that packets are kept in queues before being transmitted into the network. The results from this figure can be used to measure the level of congestion as well as the relative amount of time wasted in queues in different scenarios.



(a) 1 hop

(b) 2 hop

(c) 3 hop

(d) 4 hop

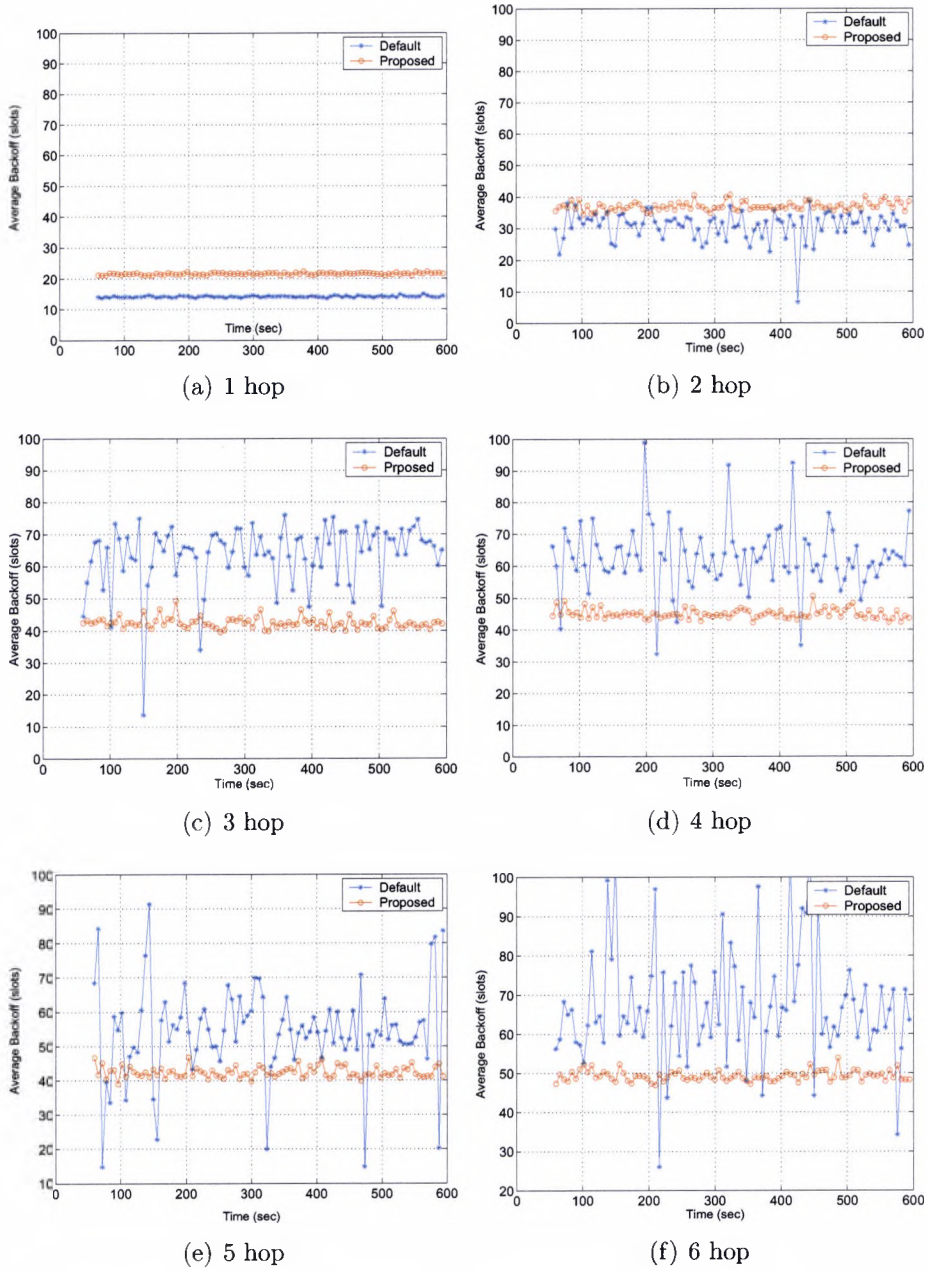
(e) 5 hop

(f) 6 hop

Figure 7.5: Queueing delay in a chain topology

The results suggest the proposed algorithm keeps the queueing delay and hence the number of packets in the buffer very low and almost fixed during the simulation time and across all number of hops. It is worth mentioning again that such a decrease in the queueing delay has been achieved without compromising the TCP goodput. In other words, the new algorithm has merely reduced the unnecessary buffering of packets in the network.

As a second link layer metric, figure 7.6 presents the average number of back-off slots before a station accesses the channel. This metric is of particular interest to us in a single flow chain topology since it provides a way to monitor any possible impact of replacing the 802.11 backoff algorithm (BEB) with FBA. Especially, let us recall from section 5.4 that FBA was designed to provide fairness between contending stations by the introduction of a Restrictive state where the probability of a node's channel access decreases according to the number of consecutive channel access events and increases with its buffer size. However, with this approach there is a danger of introducing higher delays in the case of single flow where none of the nodes are shared by any other connection. In addition, the introduction of Tradeoff-co that was added to reduce the probability of packet collision in the Normal state can have a negative impact on the number idle slots in the network.



(a) 1 hop (b) 2 hop

(c) 3 hop (d) 4 hop

(e) 5 hop (f) 6 hop

Figure 7.6: Link layer contention window size in a chain topology

The results from figure 7.6 rule out such scenarios and confirm in the case of no inter-flow contention from other connections, the new algorithm does not introduce a considerable amount of channel idle slots in comparison to the default operation of 802.11 BEB. Indeed, apart from the 1-hop and 2-hops scenarios, the channel idle slots have been decreased using the proposed algorithm. This observation is mainly due to unnecessary contention in the default algorithm and therefore choosing large contention window size by nodes after consecutive collisions. Note that the negligible change of backoff slots in the proposed scheme during the observation period, further verifies the more stable and predictable behavior of the system compared to default algorithm. The higher number of backoff slots in the 1-hop and 2-hops topologies can be explained by FBA's conservative approach in the Normal state.

Finally, figure 7.7 compares the average number of link layer attempts (ALA) before a packet is successfully transmitted to its next hop. This metric as explained earlier can essentially be used as an indication of relative power consumption efficiency of different algorithms.

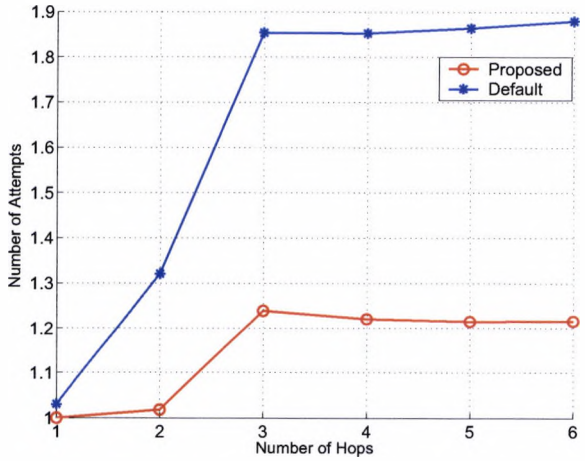


Figure 7.7: Average link layer attempts in a chain topology

From the results presented, it can be claimed that on average 32% less transmissions are required in the proposed scheme compared to the default one to deliver the same amount of link layer data traffic. In other words, the proposed algorithm can reduce the number of unnecessary packet transmissions by a factor of one third and therefore can save considerable amount of energy.

7.2.1 Impact of Probe Interval

In subsection 5.2.6, the probe interval was introduced as the number of samples in which the receiver updates its contention delay window (*ctwnd*). As discussed there, the main importance of the probe interval is to determine the sensitivity of the receiver to the contention delay information received from individual packets. Note that the choice of the probe interval becomes important when for any reason (e.g. start of new connection in the interference range of one or more of the nodes along that connection), the level of contention experienced by packets changes during the connection period. To simulate such a scenario, we used a topology shown in figure 7.8 where connection 1 (from node A o B) starts at time 0 and runs until the end of simulation while connection 2 (from node F to G) starts at time 250 seconds and lasts for 100 seconds.

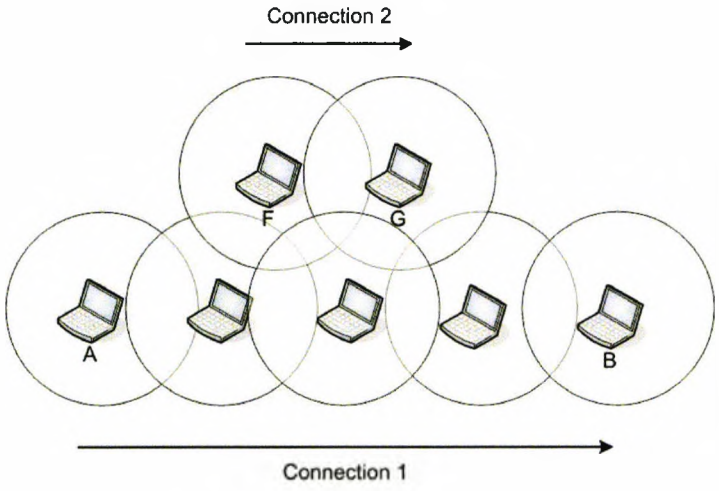


Figure 7.8: 4 hop chain topology with interference

To show the impact of different probe intervals on system performance, the average number of buffered packets across all nodes has been used as shown in figure 7.9.

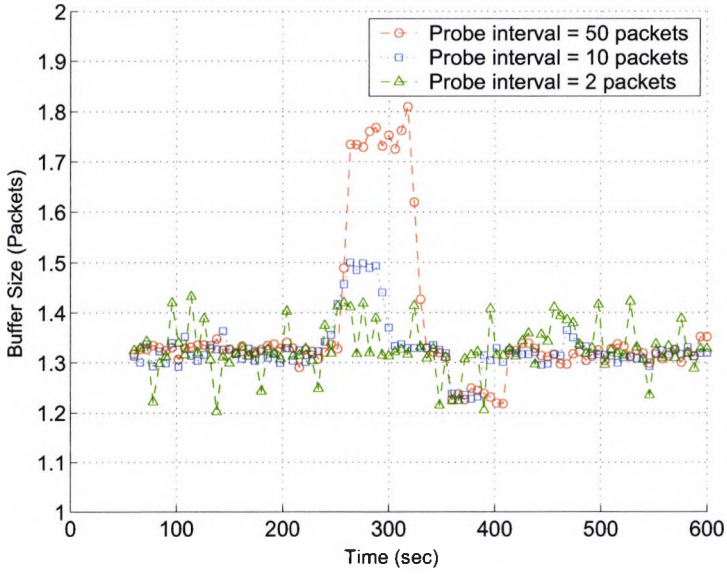
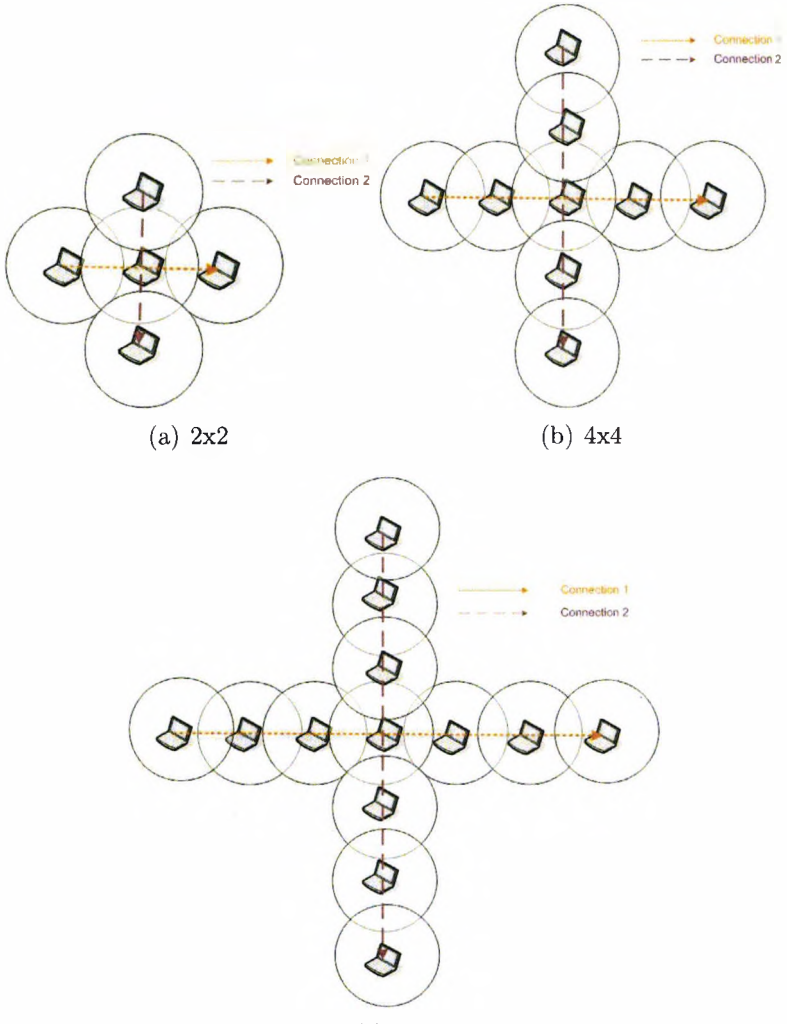


Figure 7.9: Impact of probe interval choice on buffer size

The results confirm first of all a short probe interval (e.g. 2 packets) can lead to fluctuation and therefore instability in the network as the calculation of *ctwnd* becomes very sensitive to the contention delay experienced by individual packets. On the other hand, if a large probe interval such as 50 packets is chosen, the convergence time (i.e. the time it takes for the algorithm to adjusts itself to new situation) can be considerable (as large as 60 seconds). This is definitely unacceptable since if during the probe interval time, the contention level increases (e.g. time 250 seconds in figure 7.9 when connection 2 starts), connection 1 will experience a severe delay and packet retransmissions as node B does not update its ctwnd before the end of current probe interval. On the other hand, if during the probe interval time, the contention level decreases (e.g. time 350 seconds in figure 7.9 when connection 2 stops), the channel resources would be underutilized as node B restricts node A from accessing the newly available bandwidth. Although the exact value of the optimum probe interval depends on individual situations, the simulation results suggest values close to 10 packets satisfy the objectives of introducing the probe interval.

7.3 Cross Topology

The main objective of this section is to evaluate the effectiveness of the FBA scheme in tackling the channel access unfairness as described in 4.3.1. To this aim, as shown in figure 7.10 a number of cross topologies with 2 connections are used.



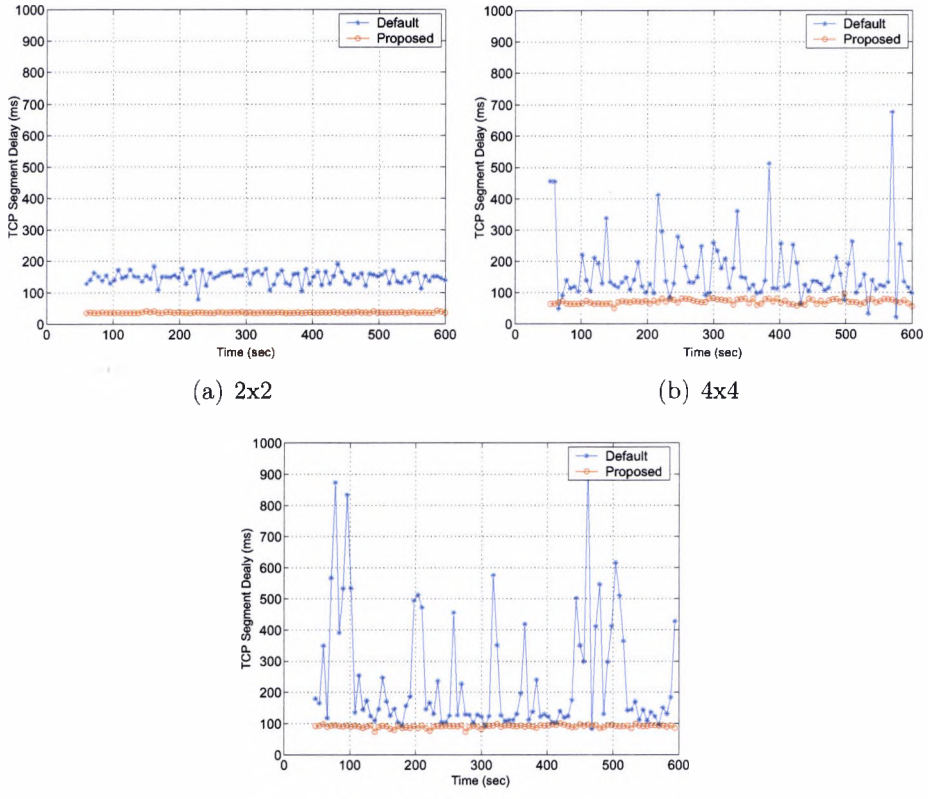
(a) 2x2 (b) 4x4

(c) 6x6

Figure 7.10: Cross topologies with different hop count

The different hop counts in figure 7.10 was considered to make sure the operation of FBA does not depend on the length of the connection. However, in order to be able to compare the results between two connections, both contending connections in each scenario will run over the same number of hops.

To evaluate the TCP stability in the cross topology, let us investigate figure 7.11 which depicts the average TCP segment delay across both connections.



(a) 2x2

(b) 4x4

(c) 6x6

Figure 7.11: TCP segment delay in a cross topology

Similar to the results presented earlier in the chain topology, the TCP packets in all cross topologies experience lower and less fluctuating delay on their way to receiver. We should note that the peaks in figure 7.11 correspond to situations

where one of the connections capture the channel and therefore the packets for other connection starves.

Figure 7.12 compares the average number of TCP retransmissions in the default and proposed algorithms.

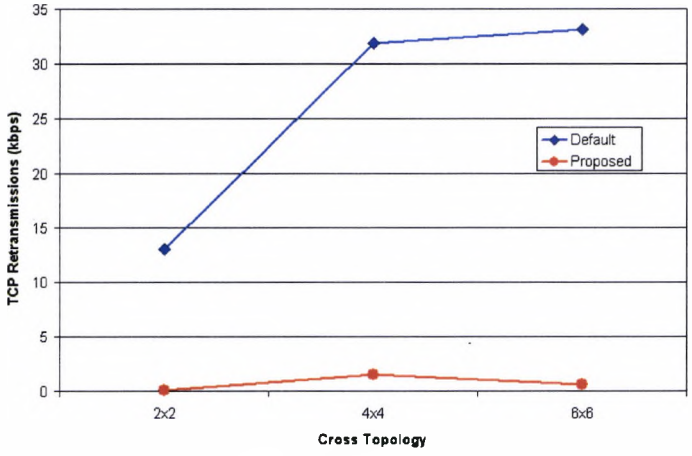


Figure 7.12: TCP retransmissions in a cross topology

It is observed that the proposed algorithms have kept the number of TCP retransmissions very low by controlling the amount of outstanding data in the network (because of TCTC), more local recovery rather than end-to-end recovery (because of DCR-ELR) and avoiding the channel capture by one of the connections (because of FBA).

Before comparing the connection's goodput and further verifying the fairness in accessing the channel resources, let us recall from section 5.4 that fairness among multiple connections was provided using the FBA algorithm that consisted of several states namely Normal, Restrictive, and Greedy. While the maximum value of contention window was deterministic in the Restrictive and Greedy state, its value was dependent on a new variable called Tradeoff-co in the Normal state. As mentioned there, this coefficient was a tradeoff between fairness and achieved

goodput. The next section investigates the impact of Tradeoff-co on system performance in more details.

7.3.1 Impact of the Tradeoff Coefficient

To investigate the impact of Tradeoff-co on the TCP goodput, let us consider the 4x4 cross topology shown in figure 7.10b. First, figure 7.13 compares the fairness index between 2 connections using the default and proposed scheme (with different value of Tradeoff-co).

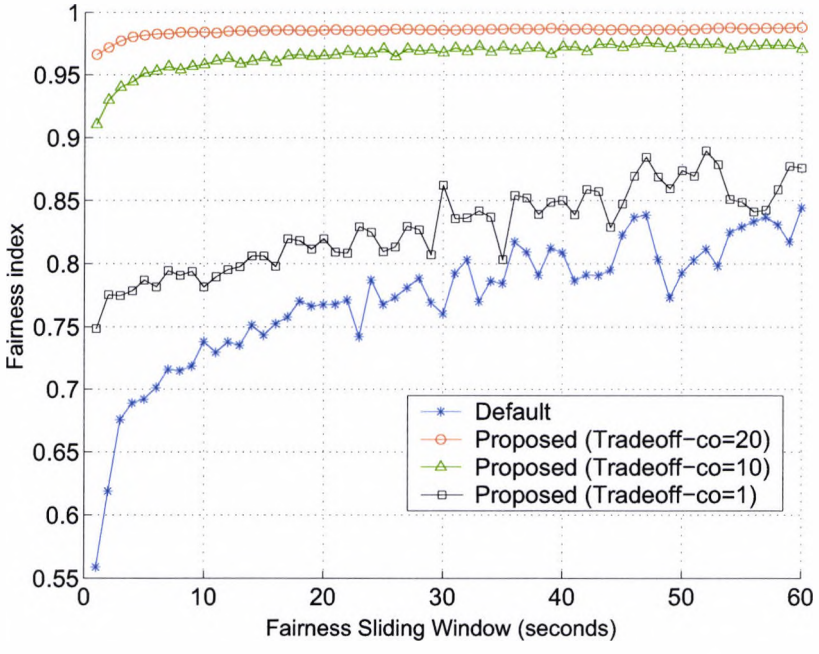
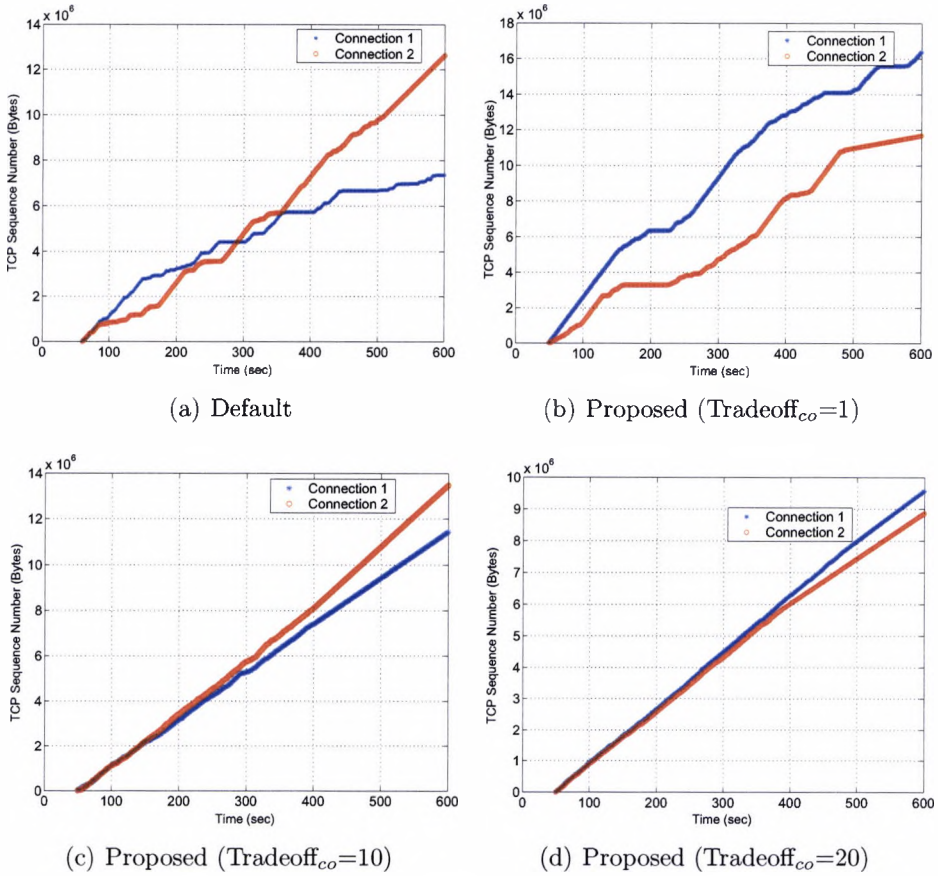


Figure 7.13: Fairness index in a 4x4 cross topology

It is clear that the introduction of new backoff algorithm has resulted in higher fairness (both short and long term) improvement over the default algorithm between contending connections. More specifically, the fairness improvement becomes considerable for Tradeoff-co larger than 1. This is expected since as

explained in section 5.4, the Tradeoff-co was introduced to provide equal opportunities for contending stations by increasing the minimum contention window. On the other hand, a larger contention window means providing better fairness (especially short-term fairness) as the probability of collisions is reduced between contending nodes. However, as the Tradeoff-co increases so does the contention window size as more idle channel slots are introduced in the system. This implies we should expect a degradation of the TCP goodput as Tradeoff-co is increased. The results depicted in figure 7.14, confirm the above statement and show the inverse impact of Tradeoff-co on TCP goodput.



(a) Default

(b) Proposed (Tradeoff$_{co}$=1)

(c) Proposed (Tradeoff$_{co}$=10)

(d) Proposed (Tradeoff$_{co}$=20)

Figure 7.14: TCP goodput in a 4x4 cross topology

Note that the change of channel fairness and TCP goodput stability versus Tradeoff-co can also clearly be extracted by comparing the change of transmitted TCP sequence number in both connections shown in figure 7.14.
To be able to compare the aggregated TCP goodput, table 7.1 presents the average TCP goodput achieved by connection 1 and 2.

Table 7.1: TCP goodput in a 4x4 cross topology

	Goodput (Connection 1)	Goodput (Connection 2)	Goodput (Average)
Default	186.18kbps	113.45kbps	149.81kbps
Proposed (Tradeoff-co=1)	237.09kbps	174.10kbps	205.6kbps
Proposed (Tradeoff-co=10)	196.36kbps	171.63kbps	184.01kbps
Proposed (Tradeoff-co=20)	138.18kbps	130.90kbps	134.54kbps

As the results from figure 7.13 and table 7.1 suggest, despite the 35% increase of aggregated TCP goodput using Tradeoff-co=1, the fairness index has improved by small margin. On the other hand, while the fairness has been dramatically improved using the Tradeoff-co=20, there is a 10% reduction in TCP goodput. Therefore, the choice of Tradeoff-co is simply a tradeoff between goodput and fairness. Based on simulation results in different scenarios, our recommendation for Tradeoff-co value is a number between 5 to 10 as it delivers an acceptable combination of fairness and TCP goodput.

7.4 Mesh Topology

To further verify the effectiveness of the proposed schemes in more realistic scenarios, two different sets of experiments are performed in a mesh topology as explained in the following section.

7.4.1 Identical Flows

The main objective of this section is to measure the effectiveness of the TCTC, FBA and DCR-ELR scheme in alleviating both intra-flow and inter-flow instability when multiple identical connections share the channel resources. To this aim, in the first mesh topology scenario shown in figure 7.15, there are in total 6 TCP sender/receiver pairs all running over 6 hops. The connections have been

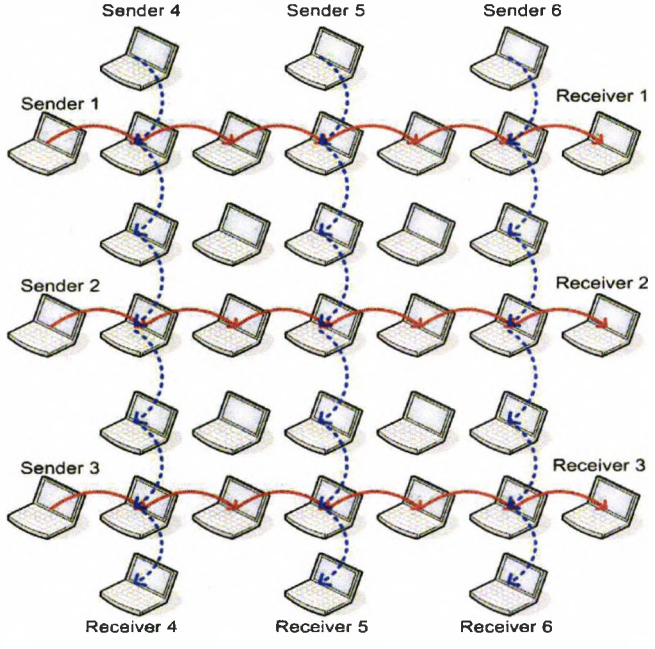
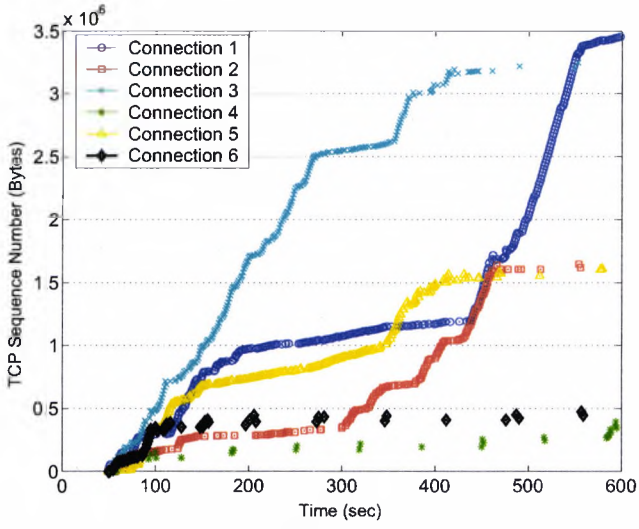


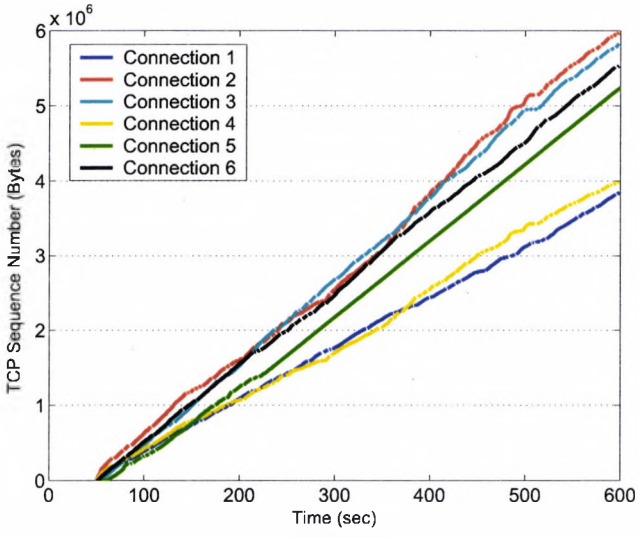
Figure 7.15: 6x6 mesh topology with identical flows

setup in a way that all connections run over 6 hops and the adjacent connections are 2 hops away from each other to avoid the hidden connection problem and therefore become identical. To evaluate the TCP instability, the change of TCP sequence number during simulation time has been used as the main parameter in this section. This is because while TCP sequence number can be used for stability comparison, it also includes the information on aggregated goodput and fairness and therefore it eliminates having multiple graphs to compare different aspects of the TCP stability using the default and proposed algorithms.

Figure 7.16 illustrates the change of TCP sequence number in mesh topology given in figure 7.15 with identical flows.



(a) Default



(b) Proposed

Figure 7.16: Illustration of TCP instability in a mesh topology with identical flows

There are couple of interesting points that can be extracted from figure 7.16. First of all, the introduction of proposed scheme considerably improves the stability of all TCP connections. We believe such stability has been mainly achieved thanks to TCTC approach in controlling the amount of outstanding data in the network. This can be more clear if we notice that in connections using the default algorithm, the TCP goodput starts dropping sharply a short time after the connections start. This clearly validates the existence of TCP instability loop explained in subsection 4.2.4 where soon after the connection starts, the TCP sender transmits more data than the network can handle resulting in excessive contention and packet drops.

The other interesting point in figure 7.16 is the lower goodput of connection 1 and 4. To investigate the cause of this observation, a detailed packet trace was performed in OPNET debugger (ODB). The main finding and explanation was that the level of contention is always higher in the first few nodes at the beginning of the connection. Since Sender 1 and 4 in figure 7.15 share the same next hop, the probability of collision in that shared node is higher than other nodes. This higher probability of collision results in extra recovery time and therefore the TCP goodput is less in comparison to the other connections.

To have a better perspective on fairness between different TCP connections in our mesh topology, figure 7.17 compares the TCP short and long term fairness between all 6 connections under default and proposed algorithms.

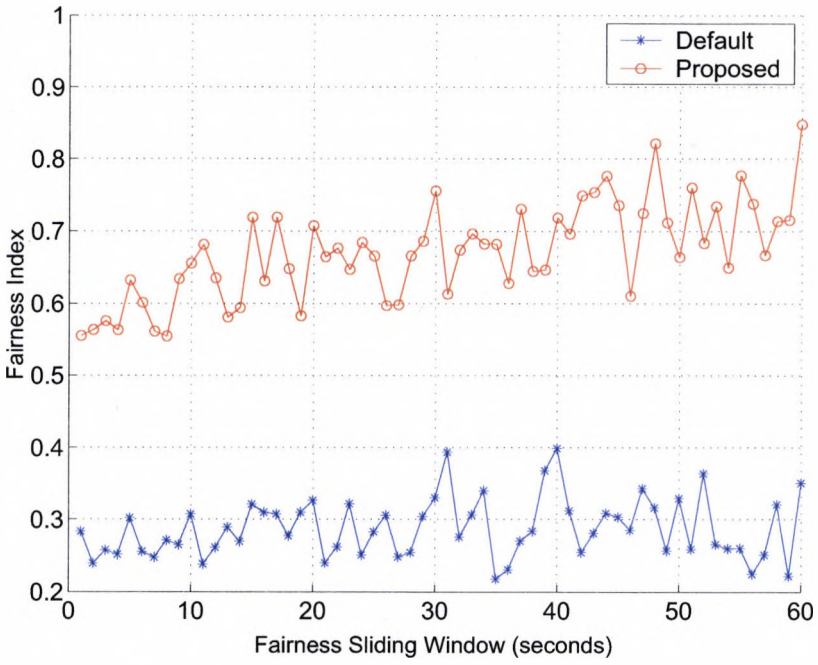


Figure 7.17: Fairness index in a mesh topology

Note that here the absolute fairness and unfairness is achieved when fairness index is equal to 1 and 0.16, respectively. The figure clearly supports the improvement of fairness between all connections using the proposed scheme mainly thanks to FBA.

7.4.2 Random Flows

In the second set of experiments, we run simulations in a mesh topology where 50 nodes are distributed randomly in a area of 1000m x 1000m and 6 random TCP source/receiver pairs are chosen as shown in figure 7.18.

Note that the main difference between the random and identical flows in the used mesh topology is twofold: Firstly, connections can have a different number

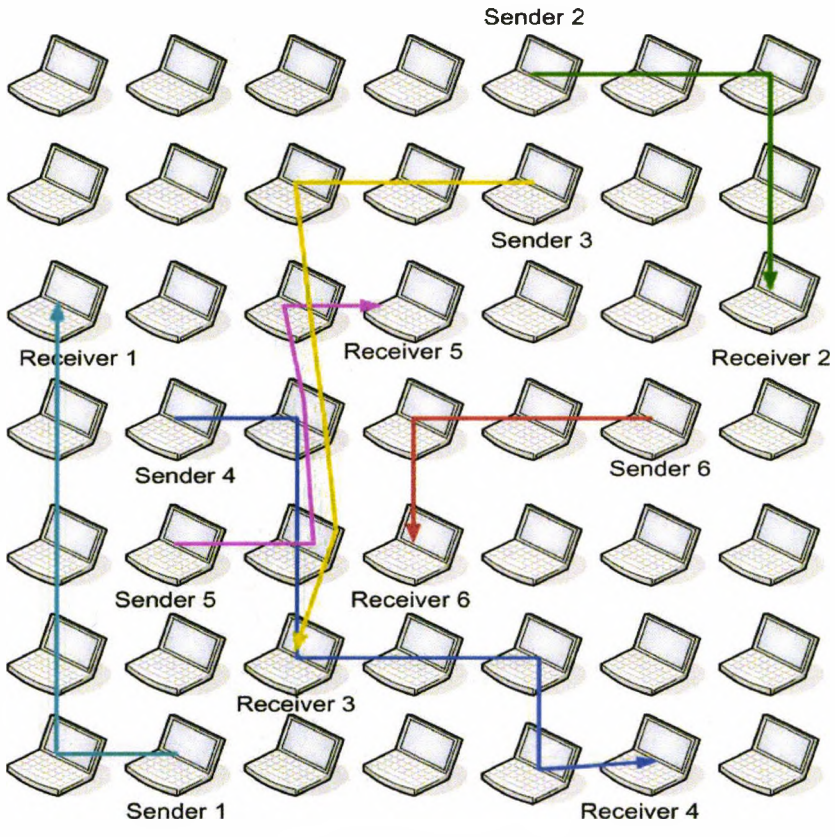
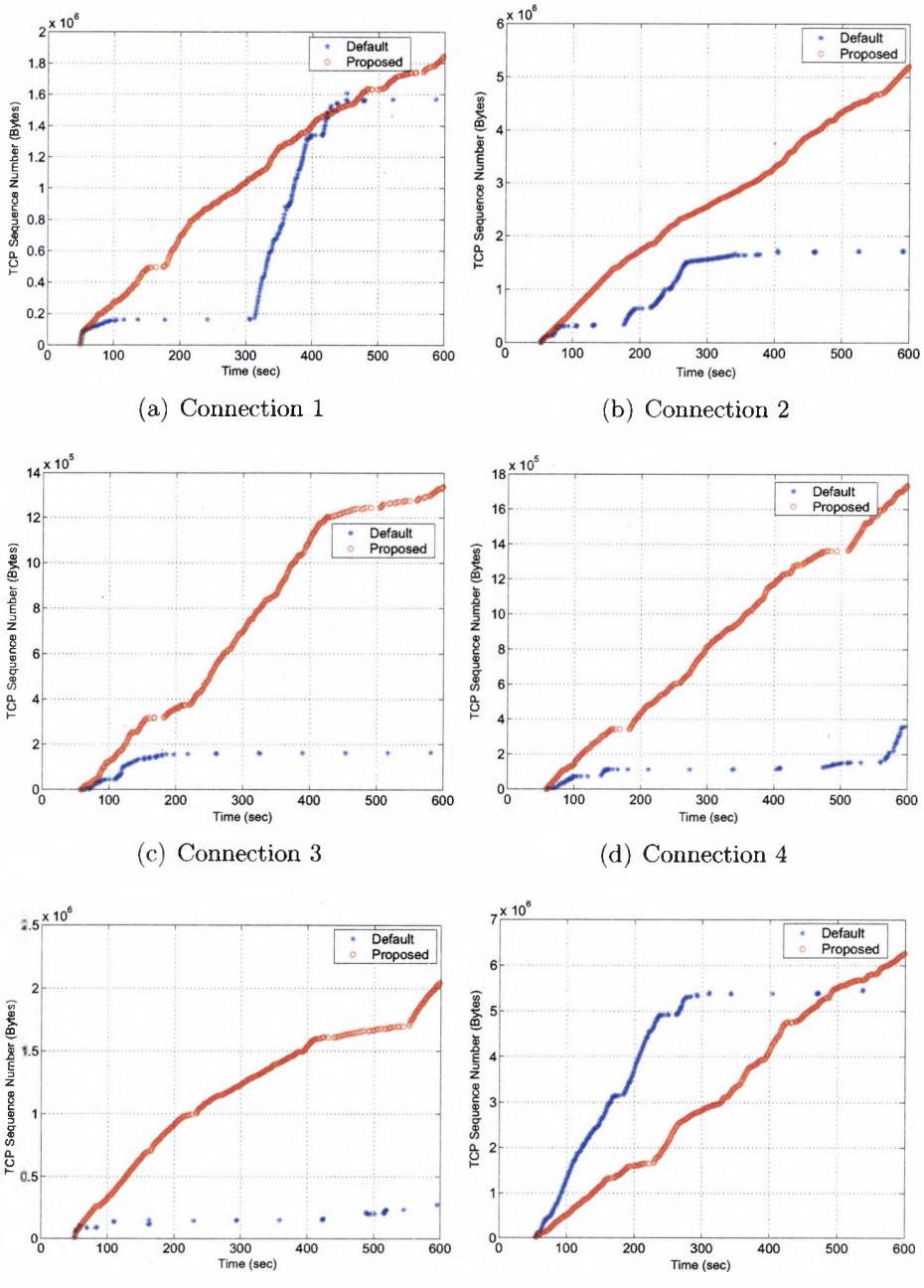


Figure 7.18: 6x6 mesh topology with random flows

of hops with respect to each other. Secondly, nodes can be shared between more than 2 connections.

Similar to section 7.4.1, the TCP sent sequence number is used to compare stability and aggregated goodput. However, since connections can run over different number of hops, the TCP sequence number of different connections cannot be compared against each other. In addition, the fairness measurements described earlier in this chapter cannot be used to evaluate the fairness between different connections as the connections are not identical. Therefore, the results presented here can only be used for TCP stability measurements.

Figure 7.19 shows the TCP sent segment sequence number across all connections.



(a) Connection 1

(b) Connection 2

(c) Connection 3

(d) Connection 4

(e) Connection 5

(f) Connection 6

Figure 7.19: TCP goodput in a random topology

The results suggest the overall improvement of TCP stability and goodput for all connections using the proposed scheme. However, in some connections (e.g. Connection 3 and 5), the TCP goodput fluctuations is not negligible and there are occasions where the connection cannot send any packet to the network for a period of time. Through detailed investigation of packet traces in the ODB, it has been realized that such a problem mainly occurs in connections where one or more of the intermediate nodes are shared between many connections (usually more than 3). In particular, the results of the packet trace shows the FBA algorithm in such situations cannot function appropriately as the FBA restricts the frequent channel access by the shared node (by forcing the node to stay in the Restrictive state). However, by putting the node in the Restrictive mode, the shared node cannot relay all the packets it receives from multiple connections which can lead to unfairness and instability.

7.5 Summary

This chapter presented the simulation results of the effectiveness of the new algorithms on TCP and 802.11 performance. To be able to investigate the effectiveness of proposed algorithms on intra-flow and inter-flow instability separately and also together, a variety of chain, cross, and mesh topologies were chosen. Also, due to cross layer impact of 802.11 and TCP on each other, a wide range of metrics across both layers were selected to have a fuller picture of the impact of the proposed scheme on the performance of both layers. The results were promising as it validated the arguments made in Chapter 4 regarding the cause of intra-flow and inter-flow instability. In addition, it was shown the introduction of TCTC,

DCR-ELR, and FBA in tackling the individual causes of TCP instability without any compromise in achieved throughput. On the other hand, it was also realized that the proposed backoff algorithm cannot be as effective as expected in scenarios where one or more of the intermediate nodes are shared among many connections. Nevertheless, the proposed package will always outperform the default algorithm and therefore it is worth being deployed in the network.

Chapter 8

Conclusion and Future Directions

8.1 Summary of Thesis

Ad hoc networks have enjoyed significant research attention in the last few years thanks to their easy deployment, maintenance and application variety. To enable seamless integration of ad hoc networks with the Internet and to provide reliable end-to-end connections, TCP seems to be the natural choice for users of ad hoc networks that want to communicate reliably with each other and with the Internet. However, the TCP protocol and its parameters were devised for wired networks. Based on assumptions that are challenged in an ad hoc networks environment, mainly due to the wireless nature of the shared medium, random channel errors and route failure. Similarly, the IEEE 802.11 standard was mostly evolved and optimized for infrastructure-based WLANs rather than ad hoc networks. As shown , such discrepancies may lead to unpredictable behaviour and performance degradation of TCP in ad hoc networks. Of particular interest in this thesis was to investigate the TCP instability issue seen in ad hoc networks.

Based on various simulation observations and analysis, the TCP instability was broken down into two main categories namely intra-flow and inter-flow instability. First it was shown a critical source of TCP intra-flow instability lies in the TCP window mechanism which controls the amount of traffic sent into the network. To tackle the problem, a cross layer algorithm called TCP Contention Control (TCTC) was proposed to adjust the amount of outstanding data in the network based on the level of contention experienced by packets as well as the throughput achieved by connections. The main features of this algorithm were its flexibility and adaptation to the network conditions.

It was also realized that channel capture serves as the primary reason behind TCP inter-flow instability. Through simple examples, it was then shown that the operation of 802.11 binary exponential backoff algorithm plays an important role in channel access unfairness. To overcome TCP inter-flow instability, a modified backoff algorithm called Fair Backoff Algorithm (FBA) was introduced to provide fair channel access between contending stations. The main design issue of FBA was to adjust the contention window size by taking into account the previous history of channel access in addition to number of packets waiting in the node's buffer.

In parallel to TCTC and FBA, a modified version of Delayed Congestion Response (DCR) called DCR-Extended Link layer Retransmission (DCR-ELR) was introduced to alleviate the TCP sensitivity to high numbers of packet reorderings due to frequent contention losses in ad hoc networks. The main feature of the proposed algorithm was to provide the link layer with extra chances of local loss recovery due to high cost of the TCP end-to-end recovery procedure.

To evaluate the impact of the proposed algorithm, OPNET simulations were used. To validate the accuracy of the simulator, a series of simple experiments

were performed on a real testbed and it was shown the simulation results closely match with testbed results. Finally through an extensive set of simulations, it was shown that the combination of the proposed schemes can dramatically improve TCP stability without compromising other network metrics especially the TCP goodput.

In addition to addressing TCP instability in ad hoc networks, a new and accurate 3 dimensional Markov model of the operation of 802.11 DCF was also presented. The objective of developing this model was to analyze the impact of 802.11 parameters on link layer throughput. Using mathematical analysis, it was shown the default parameters of 802.11 DCF result in substantial throughput degradation. It was also realized, to optimize the link layer throughput, the minimum contention window size should be adjusted according to the number of contending stations. However, in the 802.11 standard, the value of minimum contention window is hardwired in the 802.11 physical layer details, and thus it cannot be made dependent on the number of stations. As a consequence of this lack of flexibility, it was shown that the throughput especially in the large networks, can be significantly lower than the maximum achievable.

8.2 Directions for Future Work

In the course of this research, several prospects for future work have become evident and some issues may be the subject for further study. These are summarized below.

- The evaluation presented in this thesis assumes a homogeneous context, i.e. where all nodes in the network implement the proposed schemes and are all

well functioning. In reality, however any proposed alterations are deployed gradually in a network and communicating clients are expected to function adequately in a mixed environment. one research prospect along these lines would involve examining existing solutions in such heterogeneous settings and determining whether a gradual adoption is a viable option.

- During the analytical analysis of the operation of 802.11 DCF, the impact of hidden terminals on control packets was ignored to simplify the model. In addition, the network was assumed to operate in a saturated traffic. To get a more realistic model, these assumptions should be removed.

- Throughout this thesis, different metrics such as segment delay, TCP goodput, and fairness index were used to analyze and compare TCP instability in different scenarios. However, there is a clear need to be able to quantify the instability parameter by developing a metric that takes into account different instability perspectives such as delay variation, instantaneous throughput, and aggregated goodput.

- One area that requires extra attention is the concept of the fair allocation of channel resources. In particular, as shown in this thesis, the approach of providing per node fairness can be sometimes problematic in ad hoc networks as nodes which are shared between more connections should get more channel resources. However, the per node fairness approach cannot address this issue without the exchange of information packets between neighboring nodes in ad hoc networks which does not seem to be realistic or at least recommended. Therefore, more study is required to combined the concept of per node fairness with per flow fairness.

- Improving the performance of TCP in multi-hop ad hoc networks is truly a cross-layer problem. In other words, it is essential to consider the interaction of different layers when optimizing the TCP performance. One of the major areas that can dramatically improve the operation of TCP in ad hoc networks is the role that the routing protocols play. In particular, a further study is required to analyze the interaction of TCP, the routing protocol and the 802.11 MAC protocol.

- One of the main features of the TCP Contention Control proposed in this chapter was its compatibility with all TCP versions as it does not require any changes in TCP congestion control algorithm. This can be very useful in heterogeneous networks (wire + wireless) where the same TCP can be used in both networks. Therefore, comprehensive simulations should be conducted to evaluate the capability of the proposed schemes when the connection is expanding from wire to wireless networks or vice versa.

8.3 Conclusion

In general, to fully benefit from the flexibility and advantages of multihop ad hoc networks, there is a clear need for improvement of the way TCP operates in such networks. In particular, the main objective of this research was to investigate thoroughly the problem of TCP instability in multihop ad hoc networks.

The main contributions of this work are twofold. Firstly, a comprehensive analytical model of IEEE 802.11 DCF in ad hoc networks was developed. Based on the analytical results, it was realized that for a given number of stations, there exists an optimal value of packet transmission probability at which 802.11

achieves the highest throughput. However, the actual probability of packet transmission in current 802.11 standard is much higher than its optimum value hence causing dramatic 802.11 throughput degradation.
Secondly, by addressing the TCP instability problem by separating it into intra-flow and inter-flow instability, it was shown the large number of outstanding TCP packets in the network, the TCP high sensitivity to duplicate acknowledgments, TCP recovery instead of link layer recovery for contention drop packets, and the unfair 802.11 backoff algorithm are the primary causes of TCP instability. For each of the above issues, a solution was proposed and through extensive simulations, it was shown the combination of the proposed schemes can dramatically improve TCP stability without compromising other network metrics especially TCP goodput.

As discussed earlier in this chapter, there are still a variety of interesting research issues that can be the subject of future investigations. One important area is to consider solutions for intra-flow instability by improving both per node and per flow fairness. Furthermore, more simulations and testbed experiments are required to be conducted in order to fully evaluate the scalability and effectiveness of the proposed algorithms in heterogenous networks.

List of Author's Publications

1. V. Rakocevic, E. Hamadani, Wireless Ad Hoc Networks: Design Principles and Low Power Operation, Book Chapter, Ubiquitous Computing: Design, Implementation and Usability, Idea Group, 2007

2. E. Hamadani, V. Rakocevic, TCP Contention Control: A Cross Layer Approach to Improve TCP Performance in Multihop Ad hoc Networks, LNCS in Wired/Wireless Internet Communications, Springer, Volume 4517, pages 1-16, May 2007

3. E. Hamadani, V. Rakocevic, A Cross layer Analysis of TCP Instability in Multihop Ad hoc Networks, European Wireless 2007, Paris, April 2007

4. E. Hamadani, V. Rakocevic, Evaluating and Improving TCP Performance against Contention Losses in Multihop Ad Hoc Networks, IFIP International Conference on Mobile and Wireless Communication Networks , Marrakech, Morocco, September 2005

5. E. Hamadani, V. Rakocevic, Impact of RTS/CTS Handshake on the Performance of TCP Connections in Multihop Ad-Hoc Networks , PREP 2005 Conference, Lancaster, UK, April 2005

Bibliography

[1] M. Rahman and F. Mir, "Fourth Generation (4G) Mobile Networks - Features, Technologies and Issues," *IEE Conference Publication*, no. 2005-11182, pp. 255 – 259, 2005.

[2] K. Ray and S. Misra, "Fourth Generation (4G) Networks: Roadmap- Migration to the Future," *IETE Technical Review (Institution of Electronics and Telecommunication Engineers, India)*, vol. 23, no. 4, pp. 253 – 265, 2006.

[3] I. Chlamtac, M. Conti, and J. Liu, "Mobile ad hoc networking: Imperatives and challenges," *Ad Hoc Networks*, vol. 1, no. 1, pp. 13 – 64, 2003.

[4] N. Golmie, R. Van Dyck, and A. Soltanian, "Interference of Bluetooth and IEEE 802.11: Simulation Modeling and Performance Evaluation," *Proceedings of the 4th ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 11 – 18, 2001.

[5] K. Pahlavan and A. Levesque, *Wireless Information Networks*. John Wiley and Sons, 1995.

[6] H. Balakrishnan and V. Padmanabhan, "How Network Asymmetry Affects TCP," *IEEE Communications Magazine*, vol. 39, no. 4, pp. 60 – 67, 2001.

[7] K. Xu, S. Bae, S. Lee, and M. Gerla, "TCP Behavior Across Multihop Wireless Networks and the Wired Internet," *Proceedings of the 5th ACM International Workshop on Wireless Mobile Multimedia*, pp. 41 – 48, 2002.

[8] E. Jung and N. Vaidya, "A Power Control MAC Protocol for Ad hoc Networks," *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM*, pp. 36 – 47, 2002.

[9] F. A. Tobagi and L. Kleinrock, "Packet Switching in Radio Channels Dash 2. The Hidden Terminal Problem in Carrier Sense Multiple-Access And The Busy-Tone Solution," *IEEE Transactions on Communications*, vol. CM-23, no. 12, pp. 1417–1433, 1975.

[10] L. Kleinrock and F. Tobagi, "Packet Switching in Radio Channels Dash 1. Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics.," *IEEE Transactions on Communications*, vol. CM-23, no. 12, pp. 1400 – 1416, 1975.

[11] "IEEE Standards for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY),Part 11:Technical Specifications," 1999.

[12] "RFC 793 - Transmission Control Protocol," Sep 1999. http://www.ietf.org/rfc/rfc793.txt.

[13] R. Caceres and L. Iftode, "Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 5, pp. 850 – 857, 1995.

[14] M. Gerla, R. Bagrodia, L. Zhang, K. Tang, and L. Wang, "TCP over Wireless Multi-Hop Protocols: Simulation and Experiments," *Proceedings - IEEE ICC*, vol. Vol 2, pp. 1089–1094, 1999.

[15] Z. Fu, X. Meng, and S. Lu, "How Bad TCP Can Perform in Mobile Ad Hoc Networks," *Proceedings- IEEE International Symposium on Computers and Communications*, pp. 298–303, 2002.

[16] X. Li, P.-Y. Kong, and K.-C. Chua, "Analysis of TCP Throughput in IEEE 802.11 Based Multi-hop Ad hoc Networks," *Proceedings - International Conference on Computer Communications and Networks, ICCCN*, vol. 2005, pp. 297 – 302, 2005.

[17] X. Chen, H. Zhai, J. Wang, and Y. Fang, "TCP Performance over Mobile Ad Hoc Networks," *Canadian Journal of Electrical and Computer Engineering*, vol. 29, no. 1, pp. 129 – 134, 2004.

[18] M. D. Baba, N. Ahmad, M. Ibrahim, R. A. Rahman, and N. H. Eshak, "Performance Evaluation of TCP/IP Protocol for Mobile Ad hoc Network," *WSEAS Transactions on Computers*, vol. 5, no. 7, pp. 1481 – 1486, 2006.

[19] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, "The Impact of Multihop Wireless Channel on TCP Throughput and Loss," *Proceedings - IEEE INFOCOM*, vol. 3, pp. 1744 – 1753, 2003.

[20] Y. Wang, K. Yu, Y. Liu, and H. Zhang, "Analysis and Improvement of TCP Performance in Mobile Ad Hoc Networks," *International Conference on Communication Technology Proceedings*, vol. 2, pp. 1301 – 1304, 2003.

[21] W. Xu and T. Wu, "TCP issues in mobile ad hoc networks: Challenges and solutions," *Journal of Computer Science and Technology*, vol. 21, no. 1, pp. 72 – 81, 2006.

[22] Y. R. Y. Xiao, X. Shan, "Cross-Layer Design Improves TCP Performance in Multihop Ad Hoc Networks," *IEICE Transactions on Communications*, vol. E88-B, no. 8, pp. 3375 – 3381, 2005.

[23] H. Lim, K. Xu, and M. Gerla, "TCP Performance over Multipath Routing in Mobile Ad Hoc Networks," *IEEE International Conference on Communications*, vol. 2, pp. 1064 – 1068, 2003.

[24] H. Elaarag, "Improving TCP Performance over Mobile Networks," *ACM Computing Surveys*, vol. 34, no. 3, pp. 357 – 374, 2002.

[25] A. Ahuja, S. Agarwal, J. P. Singh, and R. Shorey, "Performance of TCP over different routing protocols in mobile ad-hoc networks," *IEEE Vehicular Technology Conference*, vol. 3, pp. 2315 – 2319, 2000.

[26] Z. Fu, H. Luo, P. Zerfos, S. Lu, L. Zhang, and M. Gerla, "The Impact of Multihop Wireless Channel on TCP Performance," *IEEE Transactions on Mobile Computing*, vol. 4, no. 2, pp. 209 – 221, 2005.

[27] L. Zhang, X. Wang, and W. Dou, "Analyzing and Improving the TCP Flow Fairness in Wireless Ad Hoc Networks," *Ruan Jian Xue Bao/Journal of Software*, vol. 17, no. 5, pp. 1078 – 1088, 2006.

[28] H. Zhai, X. Chen, and Y. Fang, "Improving Transport Layer Performance In Multihop Ad Hoc Networks by Exploiting MAC Layer Information,"

IEEE Transactions on Wireless Communications, vol. 6, no. 5, pp. 1692 – 1701, 2007.

[29] S. Xu and T. Saadawi, "Revealing the Problems with 802.11 Medium Access Control Protocol in Multi-Hop Wireless Ad Hoc Networks," *Computer Networks*, vol. 38, no. 4, pp. 531 – 548, 2002.

[30] S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, *Mobile Ad Hoc Networking*. John Wiley and Sons, 2004.

[31] J. Garcia-Luna-Aceves and C. L. Fullmer, "Floor Acquisition Multiple Access (FAMA) in Single-Channel Wireless Networks," *Mobile Networks and Applications*, vol. 4, no. 3, pp. 157 – 174, 1999.

[32] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A Media Access Protocol for Wireless LANs," *Computer Communications Review*, vol. 24, no. 4, pp. 212 –, 1994.

[33] P. Karn, "MACA: A New Channel Access Method for Packet Radio," *Proceeding of 9th ARRL/CRRL Amateur radio computer networking*, 1990.

[34] D. Comer, *Internetworking With TCP/IP Volume 1: Principles Protocols, and Architecture*. 5 ed., 2005.

[35] V. Jacobson and M. Karels, "Congestion Avoidance and Control," *ACM SIGCOMM Computer Communication Review*, vol. 18, no. 4, 1988.

[36] V. Paxson and M. Allman, "RFC 2988 - Computing TCP's Retransmission Timer," November 2000. http://www.ietf.org/rfc/rfc2988.txt.

[37] J. Nagle, "RFC 896 - Congestion control in IP/TCP internetworks," Jan 1984. http://www.ietf.org/rfc/rfc893.txt.

[38] W. Stevens, "RFC 2001 - TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," Jan 1997. http://www.ietf.org/rfc/rfc2001.txt.

[39] M. Allman, V. Paxson, and W. Stevens, "RFC 2581 - TCP Congestion Control," April 1999. http://www.ietf.org/rfc/rfc2581.txt.

[40] S. Bradner, "RFC 2119 - Key words for use in RFCs to Indicate Requirement Levels," March 1997.

[41] S. Floyd, T. Henderson, and A. Gurtov, "RFC3782 - The NewReno Modification to TCP's Fast Recovery Algorithm.," April 2004. http://www.ietf.org/rfc/rfc3782.txt,.

[42] L. Brakmo, S. O'Malley, and L. Peterson, "TCP Vegas: New Techniques for Congestion Detection and Avoidance," *Computer Communications Review*, vol. 24, no. 4, pp. 24 –, 1994.

[43] A. Medina, M. Allman, and S. Floyd, "Measuring the Evolution of Transport Protocols in the Internet," *Computer Communication Review*, vol. 35, no. 2, pp. 37 – 51, 2005.

[44] E. Blanton, M. Allman, K. Fall, and L. Wang, "RFC 3517 - A Conservative Selective Acknowledgment (SACK)-based Loss Recovery Algorithm for TCP," April 2003. http://www.ietf.org/rfc/rfc3517.txt.

[45] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "RFC 2018 - TCP Selective Acknowledgment Options," October 1996.

[46] T. Clausen and P. Jacquet, "RFC 3561 - Optimized Link State Routing Protocol (OLSR)," July Oct 2003. http://www.ietf.org/rfc/rfc3626.txt.

[47] E. M. Royer and C. K. Toh, "Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks," *IEEE Personal.Communications*, vol. 6, no. 2, pp. 46–55, 1999.

[48] D. Johnson, Y.-C. Hu, and D. Maltz, "RFC 4728 - The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4," Feb 2007. http://www.ietf.org/rfc/rfc4728.txt.

[49] M. Lakshmi and P. Sankaranarayanan, "Performance Analysis of Three Routing Protocols in Wireless Mobile Ad Hoc Networks," *Information Technology Journal*, vol. 5, no. 1, pp. 114 – 120, 2006.

[50] A. Boukerche, "Performance Evaluation of Routing Protocols for Ad Hoc Wireless Networks," *Mobile Networks and Applications*, vol. 9, no. 4, pp. 333 – 342, 2004.

[51] C. E. Perkins, E. M. Royer, S. R. Das, and M. K. Marina, "Performance comparison of two on-demand routing protocols for ad hoc networks," *IEEE Personal Communications*, vol. 8, no. 1, pp. 16 – 28, 2001.

[52] S. Das, C. Perkins, and E. Royer, "Performance comparison of two on-demand routing protocols for ad hoc networks," *Proceedings - IEEE INFOCOM*, vol. 1, pp. 3 – 12, 2000.

[53] O. Tickoo and B. Sikdar, "Queueing Analysis and Delay Mitigation in IEEE 802.11 Random Access MAC Based Wireless Networks," *Proceedings - IEEE INFOCOM*, vol. 2, pp. 1404 – 1413, 2004.

[54] M. Carvalho and J. Garcia-Luna-Aceves, "Delay Analysis of IEEE 802.11 in Single-Hop Networks," *11th IEEE International Conference on Network Protocols (ICNP03)*, 2003.

[55] Y. Barowski, S. Biaz, and P. Agrawal, "Towards the performance analysis of IEEE 802.11 in multi-hop ad-hoc networks," *IEEE Wireless Communications and Networking Conference, WCNC*, vol. 1, pp. 100 – 106, 2005.

[56] R. Jiang, V. Gupta, and C. V. Ravishankar, "Interactions between TCP and the IEEE 802.11 MAC Protocol," *discex*, vol. 01, p. 273, 2003.

[57] F. Cali, M. Conti, and E. Gregori, "IEEE 802.11 Wireless LAN: Capacity Analysis and Protocol Enhancement," *Proceedings - IEEE INFOCOM*, vol. 1, pp. 142 – 149, 1998.

[58] G. Bianchi, "Performance analysis of the ieee 802.11 distributed coordination function," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, pp. 535 – 547, 2000.

[59] H. Wu, Y. Peng, K. Long, S. Cheng, and J. Ma, "Performance of Reliable Transport Protocol over IEEE 802.11 Wireless Lan: Analysis and Enhancement," *Proceedings - IEEE INFOCOM*, vol. 2, pp. 599 – 607, 2002.

[60] J. He, D. Kaleshi, A. Munro, Y. Wang, A. Doufexi, J. McGeehan, and Z. Fan, "Performance Investigation of IEEE 802.11 MAC in Multihop Wireless Networks," *Proceedings of the Eighth ACM Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, vol. 2006, pp. 242 – 249, 2006.

[61] "OPNET Simulattor." http://www.opnet.com.

[62] S. Xu and T. Saadawi, "Revealing and Solving The TCP Instability Problem in 802.11 Based Multi-Hop Mobile Ad Hoc Networks," *IEEE Vehicular Technology Conference*, vol. 1, no. 54D, pp. 257 – 261, 2001.

[63] Y. Li, Q. Chen, K. Long, and S. Wu, "Analyzing and Improving the TCP Stability in Wireless Ad Hoc Networks," *Journal of Software*, vol. 14, no. 6, pp. 1178 – 1186, 2003.

[64] K. Nahm and C. Helmy, A. Jay Kuo, "TCP over Multihop 802.11 Networks: Issues and Performance Enhancement," *MOBIHOC*, pp. 277–287, 2005.

[65] K. Chen, Y. Xue, and K. Nahrstedt, "On Setting TCP's Congestion Window Limit in Mobile Ad Hoc Networks," *IEEE International Conference on Communications*, vol. 2, pp. 1080 – 1084, 2003.

[66] J. Song, K. Ahn, D. Cho, and K. Han, "A Congestion Window Adjustment Scheme for Improving TCP Performance Over Mobile Ad-Hoc Networks," *Lecture Notes in Computer Science*, vol. 4104 NCS, pp. 404 – 413, 2006.

[67] D. Katabi, M. Handley, and C. Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks," *Computer Communication Review*, vol. 32, no. 4, pp. 89 – 102, 2002.

[68] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, 1993.

[69] S. Papanastasiou and M. Ould-Khaoua, "TCP Congestion Window Evolution and Spatial Reuse in MANETs: Research Articles," *Wireless Communication and Mobile Computing*, vol. 4, no. 6, pp. 669–682, 2004.

[70] E. Altman and T. Jimenez, "Novel Delayed ACK Techniques for Improving TCP Performance in Multihop Wireless Networks," *Lecture Notes in Computer Science*, no. 2775, pp. 237–250, 2003.

[71] D. Berger, Z. Ye, P. Sinha, S. Krishnamurthyt, M. Faloutsos, and S. K. Tripathi, "TCP-friendly Medium Access Control for Ad-Hoc Wireless Networks: Alleviating Self-contention," *2004 IEEE International Conference on Mobile Ad-Hoc and Sensor Systems*, pp. 214 – 223, 2004.

[72] H. Zhai, X. Chen, and Y. Fang, "Alleviating Intra-flow and Inter-flow Contentions for Reliable Service in Mobile Ad Hoc Networks," *Proceedings - IEEE Military Communications Conference MILCOM*, vol. 3, pp. 1640 – 1646, 2004.

[73] Y. Z. F. Wang, "Improving TCP Performance over Mobile Ad-Hoc Networks with Out-Of-Order Detection and Response," *Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pp. 217–225, 2002.

[74] S. Cheng, Y. Cheng, and Y. Leu, "WTCP: An Efficient Transmission Control Protocol for Wired/Wireless Internetworking," *Proceedings of the National.Science Council., Republic.of China, Part A: Physical.Science and Engineering*, vol. 24, no. 3, pp. 176–185, 2000.

[75] Z. Li, S. Nandi, and A. K. Gupta, "Modeling the Short-Term Unfairness of IEEE 802.11 in Presence of Hidden Terminals," *Performance Evaluation*, vol. 63, no. 4-5, pp. 441 – 462, 2006.

[76] C. E. Koksal, H. Kassab, and H. Balakrishnan, "An Analysis of Short-Term Fairness in Wireless Media Access Protocols," *Proceedings ACM SIGMETRICS*, vol. 28, pp. 118–119, 2000.

[77] D. Chiu and R. Jain, "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks," *Computer Networks and ISDN Systems*, vol. 17, no. 1, pp. 1 – 14, 1989.

[78] S. Xu and T. Saadawi, "Does the IEEE 802.11MAC Protocol Work Well in Multihop Wireless Ad Hoc Networks," *IEEE Communications Magazine*, vol. 39, no. 6, pp. 130–137, 2001.

[79] L. Bononi, M. Conti, and L. Donatiello, "Design and Performance Evaluation of a Distributed Contention Control (DCC) Mechanism for IEEE 802.11 Wireless Local Area Networks," *Journal of Parallel and Distributed Computing*.

[80] R. Bruno, C. Chaudet, M. Conti, and E. Gregori, "A Novel Fair Medium Access Control for 802.11-Based Multi-Hop Ad Hoc Networks," *14th IEEE Workshop on Local and Metropolitan Area Networks, LANMAN 2005*, pp. 154–160, 2005.

[81] D. Qiao and K. Shin, "Achieving efficient channel utilization and weighted fairness for data communications in ieee 802.11 wlan under the dcf," *IEEE International Workshop on QoS*, pp. 227–236, 2002.

[82] Z. Li, S. Nandi, and A. K. Gupta, "Achieving MAC Fairness in Wireless Ad-Hoc Networks Using Adaptive Transmission Control," *Proceedings - International Symposium on Computers and Communications*, vol. 1, pp. 176 – 181, 2004.

[83] Z. Fang, B. Bensaou, and Y. Wang, "Performance Evaluation of a Fair Backoff Algorithm for IEEE 802.11 DFWMAC," *Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pp. 48 – 57, 2002.

[84] C. Chaudet, G. Chelius, H. Meunier, and R. Simplot, "Adaptive Probabilistic NAV to Increase Fairness in Ad Hoc 802.11 MAC," *Ad Hoc and Sensor Wireless Networks: an International Journal (AHSWN)*, 2005.

[85] J. Walrand and R. Gupta, "Achieving fairness in a distributed ad-hoc mac," *Systems and Control: Foundations and Applications, Springer-Birkhauser*, 2005.

[86] K. Xu, M. Gerla, L. Qi, and Y. Shu, "TCP Unfairness in Ad Hoc Wireless Networks and a Neighborhood RED Solution," *Wireless Networks*, vol. 11, no. 4, pp. 383 – 399, 2005.

[87] C. Ware, J. Judge, J. Chicharo, and E. Dutkiewicz, "Unfairness and Capture Behaviour in 802.11 Ad hoc Networks," *IEEE International Conference on Communications*, vol. 1, pp. 159 – 163, 2000.

[88] R. Braden, "RFC 1122 - Requirements for Internet Hosts - Communication Layers," Oct 1989. http://www.ietf.org/rfc/rfc1122.txt.

[89] D. Kliazovich and F. Granelli, "Cross-Layer Congestion Control in Ad Hoc Wireless Networks," *Ad Hoc Networks*, vol. 4, no. 6, 2006.

[90] G. Anastasi, E. Borgia, M. Conti, and E. Gregori, "IEEE 802.11b Ad Hoc Networks: Performance Measurements," *Cluster Computing*, vol. 8, no. 2-3, pp. 135–145, 2005.

[91] A. Kumar, E. Altman, M. Goyal, V. Kumar, D. Miorandi, M. Panda, V. Ramaiyan, and S. Srinivasan, "Analysis and Optimisation of IEEE 802.11 Wireless Local Area Networks," *Proceedings - WiOpt 2005: Third International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, vol. 2005, pp. 151 –, 2005.

[92] S. Bhandarkar, A. Reddy, M. Allman, and E. Blanton, "RFC4653 - Improving the Robustness of TCP to Non-Congestion Events," August 2006. http://www.ietf.org/rfc/rfc4653.txt,.

[93] M. Gunes, M. Hecker, and I. Bouazizi, "Influence of Adaptive RTS/CTS Retransmissions on TCP in Wireless and Ad-Hoc Networks," *Proceedings. Eighth IEEE International Symposium on Computers and Communication, 2003*, vol. 2, pp. 855–860, 2003.

[94] F. Alizadeh-Shabdiz and S. Subramaniam, "Analytical Models for Single-Hop and Multi-Hop ad Hoc Networks," *Mobile Network Application*, vol. 11, no. 1, pp. 75–90, 2006.

[95] F. Tobagi, "Modeling and Performance Analysis of Multihop Packet Radio Networks," *Proceedings of the IEEE*, vol. 75, no. 1, pp. 135 – 155, 1987.

[96] H. Hallani and S. A. Shahrestani, "Performance Evaluation and Simulation Verification for Wireless Ad-Hoc Networks," *WSEAS Transactions and Communications*, vol. 4, no. 7, pp. 355 – 362, 2005.

[97] H. Westman, *Reference Data for Radio Engineers*. Howard W. Sams Co., 6th ed., 1997.

[98] "802.11g US Robotic USR5422 USB key." http://www.usr.com/support/5422/5422-ug,.

[99] "Ndiswrapper version 1.23." http://ndiswrapper.sourceforge.net.

[100] "DSR-UU version 0.2." http://core.it.uu.se/core/index.php/DSR-UU.

[101] V. Kawadia and P. Kumar, "Experimental Investigations into TCP Performance over Wireless Multihop Networks," *Proceedings of ACM SIGCOMM 2005 Workshops: Conference on Computer Communications*, pp. 29 – 34, 2005.

[102] "TCPdump." http://www.tcpdump.org.

[103] "TCPtrace." http://jarok.cs.ohiou.edu/software/tcptrace/.

[104] "Web100." http://www.web100.org.

[105] "Ethereal." http://www.ethereal.com/.