THE STUDY OF TCP PERFORMANCE IN IEEE 802.11 BASED MOBILE AD HOC NETWORKS

LI XIA

NATIONAL UNIVERSITY OF SINGAPORE

2007

THE STUDY OF TCP PERFORMANCE IN IEEE 802.11 BASED MOBILE AD HOC NETWORKS

LI XIA

(*B. Sc., Nan Jing University, PRC*)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2007

To my father

Li DingNan

Acknowledgements

Firstly of all, I would like to express my deepest gratitude and appreciation to my supervisors, Professor Chua Kee Chaing and Dr. Kong Peng Yong for their support, encouragement, advice, and friendship during my educational stay. It is a pleasant time to work with them during the past four years and they have made my research experience at the National University of Singapore (NUS) and Institue for Infocomm Research (I2R) an invaluable treasure for my whole life.

My thanks also go to all my friends in NUS and I2R, for their help and support in solving various technical and analytical problems. The friendship with them makes my study and life fruitful and unforgettable.

Finally, I must thank my family. This work is dedicated to you.

Contents

List of Figures

List of Tables

Summary

Transmission Control Protocol (TCP) is a transport protocol that guarantees reliable ordered delivery of data packets over wired networks. Although it is well tuned for wired networks, TCP performs poorly in Mobile Ad Hoc NETworks (MANETs). This is because TCP's implicit assumption that all packet losses are due to congestion is invalid in mobile ad hoc networks where wireless channel errors, link contention, mobility and multi-path routing may significantly corrupt or disorder packet delivery. If TCP misinterprets such losses as congestion and consequently invokes congestion control procedures, it will suffer from performance degradation and unfairness. To understand TCP behavior and improve the TCP performance over multi-hop wireless networks, considerable research has been carried out. The research in this area is still active and many problems are still widely open. To the best of our knowledge, we find that most researchers identify the interaction of TCP layer with the underlying routing layers as a key factor for the poor TCP performance, and that there is little analytical study which aims to model TCP behavior over MANETs. In the thesis, we focus on the interaction between TCP and IEEE 802.11 Medium Access Control (MAC) protocol, and investigate 802.11's inadequacy in handling multiple packet losses which seriously deteriorate the TCP performance. We have carried out a mathematical analysis to TCP protocol over 802.11 based ad hoc networks rather than just conducting simulations or experiments. Based on the study, two schemes are proposed to improve network performance.

It is known that IEEE 802.11 MAC layer may wrongly assume that a route is broken due

to temporary losses in connectivity arising from medium contention or wireless channel error, which we term as a false route breakage and can degrade TCP performance significantly. In our study, it is found that false route breakages due to Request-To-Send (RTS) transmission failures are mainly attributed to the hidden terminal effect caused by MAC layer medium contention. In contrast, wireless channel error is the dominant factor which leads to TCP data packet transmission failures caused false route breakages. To investigate these two kinds of false route breakages, we first present a unique quantitative study of a single TCP flow over an n-hop static string topology ad hoc network using IEEE 802.11 protocol. The analysis results in formulae to compute the throughput of a single TCP flow for an n-hop string topology under the ideal situation where there is no packet loss in the network. As such, the derived TCP throughput is the upper bound throughput and can be used as a guideline for the best case performance. Our analysis shows that the likelihood of false route breakages due to RTS transmission failures is dependent on the path length of a source-destination pair, and in particular, is proportional to the size of TCP segments in a network. Hence, we propose a simple enhancement to IEEE 802.11 MAC protocol which increases the reliability of wireless links by reducing false route breakages due to RTS transmission failures.

Subsequently, a packet level model is proposed to investigate the impact of wireless channel error on TCP performance during persistent data transmission over IEEE 802.11 based multi-hop wireless networks. We investigate and compare two TCP flavors which are Reno and Impatient NewReno. We use a Markov renewal approach to analyze the behaviors of these two TCP flavors. Compared to the previous works, besides the modelling of multiple lossy links, our model investigates the interactions among TCP, IP and MAC protocol layers, specifically the impact of 802.11 MAC protocol and Dynamic Source Routing (DSR) routing protocol on TCP throughput performance. Considering the spatial reuse property of the wireless channel, the model takes into account the different proportions between the interference range and transmission range. Moreover, the model adopts more accurate and realistic analysis to

fast-recovery process, and shows the dependency of throughput and the risk of experiencing successive fast-retransmits and timeouts on the packet error probability. The results show that the impact of wireless channel error is reduced significantly due to the packet retransmissions on a per-hop basis and small values of Bandwidth Delay Product (BDP) over ad hoc networks. The TCP throughput always deteriorates less than $\sim$10% with a packet error rate ranging from 0 to 0.1. It is found that the TCP performance for different path length varies with different values of the *long retry limit*, and the default value of four does not always provide the best TCP performance for packet error rate $q > 0.1$. Our model provides us with a theoretical basis for the design of an optimum *long retry limit* for IEEE 802.11 MAC protocol to eliminate false route breakages caused by wireless channel error.

Finally, we propose a new transport protocol for MANETs instead of making modifications to the original TCP. This is because, provided that the BDP is very small in 802.11 based ad hoc networks, any Additive Increase Multiplicative Decrease (AIMD)-style congestion control in TCP is costly and hence not necessary. On the contrary, a technique to guarantee reliable transmission and to recover packet losses plays a more critical role in the design of a transport protocol over ad hoc networks. With this basis, we propose a novel and effective Datagram-oriented end-to-end reliable Transport Protocol in Ad hoc networks, which we call DTPA. The proposed scheme incorporates a fixed window based flow control and a cumulative bit-vector based selective ACK strategy. A mathematical model is developed to evaluate the performance of DTPA. Based on this model, an optimum transmission window is determined for a n-hop chain and is the value of BDP plus 3.

In this thesis, all analytical results and proposals are verified and validated using simulator GloMoSim. Further study in the research areas of reliable transport protocols, MAC and routing protocols over mobile ad hoc networks are quite promising. Several possible extensions of our research are addressed at the end of this thesis.

Abbreviations

MANET	Mobile Ad Hoc NETwork
TCP	Transmission Control Protocol
IP	Internet Protocol
ACK	ACKnowledgement
PACK	Positive ACK
NACK	Negative ACK
SACK	Selective ACK
MAC	Medium Access Control
AIMD	Additive Increase Multiplicative Decrease
BDP	Bandwidth Delay Product
RTT	Round Trip Time
RTO	Retransmission TimeOut
FTP	File Transport Protocol
HTTP	Hypertext Transfer Protocol
BER	Bit Error Rate
DCF	Distributed Coordination Function
PCF	Point Coordination Function

SNR	Signal to Noise Ratio
RTS	Request To Send
CTS	Clear To Send
FIFO	First In First Out
SIFS	Short Inter–Frame Space
PIFS	Point Inter–Frame Space
DIFS	DCF Inter–Frame Space
DSR	Dynamic Source Routing
AODV	Ad hoc On Demand Distance Vector
TORA	Temporally-Ordered Routing Algorithm
DTPA	Datagram Transport Protocol for Ad hoc networks

Chapter 1

Introduction

The past decade has shown a phenomenal growth in wireless communications. In parallel with the single hop model for today's cellular wireless networks, another type of model, multi-hop model, is currently being developed towards a lot of applications. This newly emerged network, which is called Mobile Ad hoc NETwork (MANET), is a complex distributed system that consists of wireless nodes that can freely and dynamically self–organize. In this way, they form arbitrary and temporary "ad hoc" network topologies, allowing devices to seamlessly interconnect in areas with no pre-existing infrastructure. MANETs have the following salient features:

- Autonomous terminal: Mobile terminals connected by wireless links are free to move randomly and can act as either hosts or routers at the same time.
- Distributed operation: All the mobile terminals are distributed in the network and collaborate to implement functions. MANETs operate without any centralized administration.
- Multi-hop routing: Since there is no infrastructure, when delivering from one source to the destination that is out of the direct wireless transmission range, the content messages have to be forwarded via one or more intermediate nodes.

- Dynamic network topology: Since all the terminals are mobile, the network topology changes rapidly and unpredictably, and the network connectivity also varies with time. The mobile terminals make the routing dynamically established and hence form their own network on the fly.

- Fluctuating link capacity: It is already well-known that the wireless channel has less bandwidth than a wired network. Besides, the wireless transmission channel is greatly subject to noise, fading and interference, which makes the nature of high bit error rate more prominent in MANETs.

- Light-weight terminals: Terminals are often portable and small-sized and hence have less CPU processing capability, small memory size and low power storage.

These special features unique to MANETs bring it great opportunities. Because MANETs can be used in any place where there is little or no infrastructure or existing infrastructure is expensive or inconvenient to use, the application of MANETs is diverse. Some typical applications are as follows. In military battlefields, it can be used for exchange of information among soldiers, vehicles and military information headquarters. In commercial sectors, it can be used to spread and share information in civilian environments like buses, taxicabs, sports stadiums, etc, or among participants with laptops or palmtop computers at a conference or a classroom. Also, it can be used in emergency rescue operations for disaster relief efforts such as in fire, flood and earthquake.

Regardless of these attractive applications, the features of MANETs introduce many challenges. Since the network connection as well as the mobile node characteristics differ from the static wired case, conventional network protocol stacks result in many problems in MANETs. Considerable research efforts have been put on this new challenging paradigm of MANETs. Diverse contributions have been reported in the literature including security, energy efficiency, network architecture, mobility management, Quality of Service (QoS), rout-

ing protocols, Medium Access Control (MAC) protocols, reliable transport protocols such as Transmission Control Protocol (TCP), etc.

Due to the prevalence of TCP application, the research on the TCP performance improvement in wireless ad hoc networks becomes a hot issue. This thesis focuses on the investigation of TCP in MANETs. In the following section, we briefly review and summarize the basic characteristics of TCP in MANETs.

1.1 TCP Performance in MANETs

TCP was originally designed to provide reliable end-to-end delivery of data in conventional wired networks where packet loss is a rare event and packet reordering is infrequent. TCP adopts a window based Additive Increase Multiplicative Decrease (AIMD) congestion control algorithm coupled with the *fast retransmit* and *fast recovery* mechanisms [1–3]. With such a technique, the TCP source keeps increasing the sending rate of packets as long as no packets are lost. When packet losses occur, the TCP source backs off its sending rate by cutting the window size in order to avoid further congestion and packet losses. Thus, basically TCP infers that every packet loss is due to congestion which appears in the form of buffer overflow. TCP has been well tested and studied over the years. Also, a large number of Internet applications such as Hypertext Transfer Protocol (HTTP) and File Transport Protocol (FTP) have already been developed using TCP. According to recent estimates, 95% of the traffic carried today over wide–area Internet Protocol (IP) networks uses TCP as the transport protocol, which amounts to 80% of the overall end–to–end flow count [4]. It is reasonable to think that MANETs will eventually be part of the global Internet because of its attractive applications in many areas. Thus, TCP should naturally become the transport protocol for MANETs.

However, simply extending TCP as used over the wireline links to the wireless links is not an efficient solution due to the different characteristics of the wireline and the wireless links.

The legacy TCP protocol is known to perform poorly in MANETs. This is because TCP is unable to distinguish packet losses due to different reasons and reacts to all packet losses as if they are caused by buffer overflow during network congestion. It has been investigated in [5] that packet losses due to other factors dominate over packet losses due to buffer overflow in ad hoc networks. These factors include: (i) genuine route breakages due to the mobility of the node; (ii) MAC layer medium contention; and (iii) transmission failure due to high Bit Error Rate (BER) of a wireless channel. Typically, factor (ii) and (iii) are specific to IEEE 802.11 MAC protocol [6], in which the MAC layer regards a certain number of failures to transmit a packet as a sign of a broken link and then informs the upper routing layer, which subsequently triggers route error diffusion and route re-establishment process in the network. When a route is broken due to medium constraints, i.e., factor (ii) and (iii), the judgment on route breakage is not accurate because the two communicating nodes are still within each other's transmission range. We term this kind of route breakages as false route breakages. False route breakages can result in many packets being dropped, route maintenance, route error diffusion and excessive retransmissions. This significantly increases routing overhead, prolongs end-to-end delay and deteriorates TCP throughput.

As such, it can be seen that the interaction of TCP layer with the underlying MAC layer plays a critical role in the TCP performance in MANETs. The prevailing MAC protocol used today is IEEE 802.11 MAC protocol with the basic access mechanism DCF (Distributed Coordination Function). Previous studies have also shown that, due to the spatial reuse property of 802.11 MAC protocol in MANETs, BDP of a connection approximates 1/4 of the path length and is as low as several packets which is only a few kilo bytes. This property has been investigated in details in [7] and revisited in [5,8] under TCP perspective. Under such a condition, excessive packets are pumped into the network using a large transmission window relative to its BDP, resulting in a heavy congestion and large packet delay.

In the literature, a great amount of work have been carried out to study and to improve

the TCP performance in MANETs via experiments or simulation. However, it is noticed that the interaction of TCP with the MAC protocol have not been sufficiently explored. Also, it is found that there is little analytical study which aims to model TCP behavior over MANETs. In this thesis, we focus on the interaction between TCP and IEEE 802.11 MAC protocol, and investigate 802.11's inadequacy in handling multiple packet losses which seriously deteriorate the TCP performance. We have carried out a mathematical analysis to TCP protocol over 802.11 based ad hoc networks rather than just conducting simulations or experiments. Based on the study, two schemes are proposed to improve network performance.

1.2 Research Objectives

This thesis first develops an analytical model to quantify the upper bound of end-to-end throughput of a TCP flow across an 802.11 based n-hop string topology [71]. In this model, we remove all the packet losses introduced by buffer overflow, node mobility, wireless channel error and MAC layer contention by making appropriate and careful assumptions so that the network is a static network with infinite buffer and MAC retry limit at each node and without channel error. As such, our derived TCP throughput can be used as a guideline for the best case performance and a basis for our later investigation of how different packet losses contribute to the deterioration of the TCP performance in MANETs.

As stated in Section 1.1, besides buffer overflow, packet losses in MANETs occur not only due to mobility but also due to medium contention and wireless channel error. The network system needs to distinguish the nature of various packet losses so that it can take the most appropriate action for each case. In this thesis, we do not address the mobility and buffer overflow caused packet losses related issues, as buffer overflow hardly occurs in MANETs, and our work can well be utilized together with the early findings done by other researchers in the field of mobility. Instead, we focus on the investigation of packet losses caused by MAC

layer medium contention and wireless channel error arising from the interaction between the TCP layer and the underlying MAC layer, which is still an active research area. Clearly IEEE 802.11, while it is generally available and simple to use, has significant impact on the TCP performance in MANETs [9–11]. Typically, with 802.11, a node assumes a route is broken after seven consecutive failures in sending a Request-To-Send (RTS) packet (denoted as *short retry limit*) or four consecutive failures in sending a TCP data packet (denoted as *long retry limit*). Both medium contention and wireless channel error can lead to packet transmission failures and hence result in false route breakages.

Based on the analysis of the TCP throughput upper bound, we further investigate the impact of packet losses caused by MAC layer medium contention and wireless channel error which can lead to false route breakages. It is found that false route breakages due to RTS transmission failures are mainly attributed to the hidden terminal effect caused by MAC layer medium contention. In contrast, wireless channel error is the dominant factor which leads to TCP data packet transmission failures caused false route breakages. We study the two kinds of packet losses separately.

In the case of false route breakages due to RTS transmission failures, our analysis shows that the likelihood of false route breakages is proportional to the size of TCP segments in a network [72]. Through simulation, we find that false route breakages are dependent on the path length of a source-destination pair, and a 4-hop linear chain suffers the most serious medium contention caused false route breakages. We then present a protocol enhancement that enables IEEE 802.11 MAC protocol to alleviate false route breakages. Our scheme is a simple modification to IEEE 802.11 MAC protocol and hence the problem generated at the lower MAC layer is hidden from the upper routing and transport layers. We achieve the goal of alleviating false route breakages by initiating a HELLO message to the sender whenever the number of RTS received by a receiver exceeds a threshold.

Considering the wireless channel error, we propose a packet level model to investigate

the impact of wireless channel error on TCP performance over IEEE 802.11 based multi-hop wireless networks [73,74]. A Markov renewal approach is used to analyze the behavior of TCP Reno and TCP Impatient NewReno. Considering the spatial reuse property of the wireless channel, the model takes into account the different proportions between the interference range and transmission range. We adopt more accurate and realistic analysis to fast-recovery process, and investigates the interactions among TCP, IP and MAC protocol layers, specifically the impact of 802.11 MAC protocol and DSR routing protocol on TCP throughput performance. The model also provides a theoretical basis for designing an optimum *long retry limit* for IEEE 802.11 to reduce the likelihood of false route breakages.

Finally, in view of the limitations of the exiting proposed transport protocols, we propose a novel and effective Datagram-oriented end-to-end reliable Transport Protocol in Ad hoc networks, which we call DTPA [75]. Because the BDP in 802.11 based MANETs is very small, any AIMD-style congestion control algorithm is costly and hence not necessary for ad hoc networks. On the other hand, the strategy to guarantee a reliable transmission and to recover the frequent packet losses plays a more critical role in the design of a transport protocol. With this basis, our scheme incorporates a fixed window based flow control and a bit-vector based selective ACK strategy where the ACK packets contain a vector of bits representing the receiving status of set of earlier packets. A packet is assumed to be lost if the source finds that at least two ACKs carry the loss information of that packet or if the source cannot receive corresponding ACK within the expected time. Furthermore, we develop a parametrised mathematical model for the behavior of DTPA protocol based on a renewal process. Based on this model, an optimum transmission window is determined for a n-hop chain and is the value of BDP plus 3.

1.3 Organization of the Thesis

The rest of the thesis is organized as follows.

In Chapter 2, a general literature review is presented, including the current TCP study in MANETs and the modelling of TCP in the Internet, on which this research is based.

In Chapter 3, we investigate the TCP performance without considering wireless channel error. A general methodology is firstly presented to calculate the upper bound of the TCP throughput in a static 802.11 based linear ad hoc network under the ideal situation where there is no packet loss. Then, we further investigate the property of false route breakages due to MAC layer medium contention, and present a protocol enhancement to the IEEE 802.11 MAC protocol which increases the reliability of wireless links by reducing false route breakages.

In Chapter 4, we propose a packet level model to investigate the impact of wireless channel error on TCP performance over IEEE 802.11 based multi-hop wireless networks. A Markov renewal approach is used to analyze the behavior of TCP Reno and TCP Impatient NewReno. The results show that the TCP throughput always deteriorates less than $\sim$10% with a packet error rate ranging from 0 to 0.1. Our model also provides a theoretical basis for designing an optimum *long retry limit* for IEEE 802.11 in ad hoc networks.

In Chapter 5, we present that, provided that the BDP is very small and known before the connection establishment, any AIMD-style congestion control is costly and hence not necessary for ad hoc networks. On the contrary, a technique to guarantee reliable transmission and to recover packet losses plays a more critical role in the design of a transport protocol over ad hoc networks. With this basis, we propose a novel and effective Datagram-oriented end-to-end reliable Transport Protocol in Ad hoc networks, which we call DTPA. A mathematical model is developed to evaluate the performance of DTPA and to determine the optimum transmission window used in DTPA.

In Chapter 6, we conclude our research work up to now and envision prospect extensions.

Chapter 2

Literature Review

This chapter reviews and discusses the two major areas related to the work described herein. These are the study of TCP performance in ad hoc networks and the mathematical approaches to model TCP in the Internet.

2.1 TCP in MANETs

2.1.1 Challenges for TCP in MANETs

Unlike wired networks, some unique characteristics of mobile ad hoc networks seriously deteriorate TCP performance. These characteristics include the unpredictable wireless channels due to fading and interference, the vulnerable shared media access due to random access collision, the hidden terminal problem and the exposed terminal problem, and the route breakages due to node mobility. Undoubtedly, all of these pose great challenges on TCP to provide reliable end-to-end communications in mobile ad hoc networks. To understand TCP behavior and improve the TCP performance over MANETs, a considerable amount of research has been carried out over the last few years. Several survey literature [12–17] has summarized the works that have been done in this field. From the point of view of network layered architecture,

the challenges for TCP in MANETs can be broken down into five categories, i.e., the channel error, the power limit, the medium contention and collision, the mobility, and the multi-path routing, whose adverse impacts on TCP are elaborated below in sequence.

A. Lossy Channels

In wireless channels, relatively high bit error rate because of multipath fading and shadowing may corrupt packets in transmission, leading to the losses of TCP data segments or ACKs. The main causes of errors in wireless channel are the following:

- *Signal attenuation*: This is due to a decrease in the intensity of the electromagnetic energy at the receiver (e.g., due to long distance), which lead to low signal-to-noise ratio (SNR).
- *Doppler shift*: This is due to the relative velocities of the transmitter and the receiver. Doppler shift causes frequency shifts in the arriving signal, thereby complicating the successful reception of the signal.
- *Multipath fading*: Electromagnetic waves reflecting off objects or diffracting around objects can result in the signal travelling over multiple paths from the transmitter to the receiver. Multipath propagation can lead to fluctuations in the amplitude, phase, and geographical angle of the signal received at the receiver.

Bit errors cause packets to get corrupted which result in lost TCP data segments or ACKs. When ACKs do not arrive at the TCP sender within the expected time RTO (Retransmission TimeOut), the sender retransmits the data segment, exponentially backs off its retransmit timer for the next retransmission, reduces its congestion control window threshold, and closes its congestion window to one segment. Repeated errors will ensure that the congestion window at the sender remains small resulting in low throughput. It is important to note that error

correction may be used to combat high BER, but it will waste valuable wireless bandwidth when correction is not necessary.

B. Energy Efficiency

As power is limited at mobile nodes, any successful scheme must be designed to be energy efficient. In some scenarios where battery recharge is not allowed, energy efficiency is critical for prolonging network lifetime. Further, route breakages can occur due to the energy constrained operation of nodes, which may invoke congestion control mechanism and route re-computation. In [18], the energy consumption behavior of three versions of TCP−Reno, NewReno, and SACK is compared. The study in [19] showed, there exists a tradeoff between the individual packet transmission energy and the likelihood of retransmission, which ties to the session throughput.

C. Medium Contention

For wireless ad hoc networks, the prevailing MAC protocol used today is IEEE 802.11. The impact of medium access contention on TCP performance can be attributed to three well-known factors. The first one is the famous hidden and exposed terminal problem, which introduces spatial reuse inefficiently and collisions of packets at receivers [11]. Furthermore, TCP may encounter serious unfairness problems [20–24], because the binary exponential backoff scheme always favors the latest successful transmitter. Finally, false route breakages may occur resulting from the repeated transmission failure due to link layer contention [10], although the two adjacent nodes are still in each other's transmission range.

D. Mobility

A route breakage occurs when two adjacent nodes are split into two isolated parts of the network. The main reason of this route breakage in MANETs is node mobility. TCP cannot

distinguish between packet losses due to route failures and packet losses due to congestion. Therefore, TCP congestion control mechanisms react adversely to such losses caused by route breakages [25–28]. Meanwhile, discovering a new route may take significantly longer time than TCP sender's RTO. If route discovery time is longer than RTO, TCP sender will invoke congestion control after timeout. The already reduced throughput due to losses will further shrink.

E. Multi-path Routing

Some routing protocols such as Temporally-Ordered Routing Algorithm (TORA) maintain multiple routes between source-destination pairs, the purpose of which is to minimize the frequency of route re-computation. Unfortunately, this sometimes results in a significant number of out-of-sequence packets arriving at the receiver. The effect of this is that the receiver generates duplicate ACKs which cause the sender (on reception of three duplicate ACKs) to invoke congestion control.

2.1.2 Main Existing Proposals

Early in the course of TCP research in MANETs, most of researchers identify the interaction of TCP layer with the underlying routing layers as a key factor for the poor TCP performance in this highly dynamic scenario, as the mobility-induced network disconnection and reconnection can seriously deteriorate TCP performance [25, 26, 28–38]. As such, numerous solutions have been proposed to minimize the problems arising out of frequent route breakages [25, 29–35], where the TCP source misinterprets the route breakage caused packet losses as congestion losses. In these approaches, TCP is modified to detect route breakages replying on explicit [25, 29–31] or implicit [32–34] feedback information from inside the network and hence enters a frozen state until the route is re-established. In the frozen state, TCP stops sending data packets, and freezes all its variables to their current values. After the route is re-established,

TCP sender goes back to the normal state. One drawback of these schemes is that they do not differentiate genuine and false route breakages, so that TCP reacts adversely when a false route breakage occurs.

Recently, a number of approaches have been proposed to alleviate MAC layer medium contention in MANETs, which reduce the probability of false route breakages. These proposals can be divided into two categories: non-offered load control and offered load control algorithms. In the first category, several ideas have been proposed through modifications to IEEE 802.11 MAC protocol. Desilva et al. in [39] developed a scheme where a node responds with CTS packet transmissions whenever the noise level is below a threshold. The drawback of ignoring the busy channel is that it can lead to serious interference to the networks in some scenarios. In [40], it was proposed that packets with large forward hops have higher chances for retransmission over the wireless link, since the cost of route maintenance for larger hops is higher. [5] proposed an adaptive pacing approach at the link layer for distributing traffic among intermediate nodes in a more balanced way to avoid medium contention. However, both these two schemes delay the detection of a genuine route breakage due to node mobility in a mobile network.

On the other hand, the offered load control algorithms are realized by restricting the injection of redundant traffic into ad hoc networks [41–50], which can reduce medium contention in the network. It is known that, due to the spatial reuse property of 802.11 MAC protocol in MANETs, BDP of a connection approximates 1/4 of the path length and is as low as several packets which is only a few kilo bytes. Under such a condition, excessive packets are pumped into the network using a large transmission window relative to its BDP, resulting in a heavy congestion and large packet delay. The proposals with offered load control algorithms can be further divided into two categories: non-TCP variants and TCP variants. The schemes in [41–43] belong to non-TCP variants where the source adjusts the transmission rate based on the explicit feedback from intermediate nodes along the path. In these schemes, although

relatively accurate congestion information can be obtained, the algorithms cannot retain the end-to-end semantics of a transport protocol. Moreover, they incur complicated mathematical computation and excessive network overhead.

In contrast with non-TCP variants, the majority of the proposals are modified versions of the legacy TCP protocol. In [34, 44, 50], the strategies proactively detect the incipient congestion relying on some measured metrics at the source, such as the variation of the measured RTT, short-term throughput, etc, since the packet loss information alone cannot provide an accurate congestion indication. In [47–49], the authors attempted to minimize the contentions between data and ACK packets by adaptively reducing the number of ACK packet transmissions in the network. The drawback of these TCP-variants is that they still adopt AIMD congestion control algorithm with unnecessary large transmission windows. Chen et al. in [45] proposed that a TCP sender limits the size of its maximum transmission window to the BDP of the path in multihop networks. Though the maximum transmission window is limited at the value of BDP based on the path length, TCP is costly in terms of AIMD with such a small transmission window. Moreover, the TCP source cannot detect the packet loss by receiving enough duplicate ACKs, which can deteriorate the throughput significantly.

It should be noted that all these offered load control algorithms mentioned above are unable to eliminate medium contention caused packet losses completely in MANETs, because they cannot guarantee a balanced distribution of traffic in the networks, since packet losses caused by medium contention in MANETs are location dependent due to hidden or exposed terminal effect. In addition, these algorithms involve modifications of the transport layer and the routing layer, even though it is only the MAC layer that is misbehaving in determining a broken route.

2.2 Mathematical Modelling of TCP

Internet research is driven by simulations, experiments, analysis, and deployment studies designed to address particular problems in the Internet. However, to the best of our knowledge, we find that almost all the studies on TCP performance over MANETs are conducted via experiments and simulations. There is little analytical study which aims to model TCP behavior over MANETs. The mathematical investigation of TCP over MANETs is still an active research area.

In order to gain a deeper understanding of the way TCP works over networks, a considerable amount of analytical works have been developed to model TCP in the presence of packet losses caused by congestion and transmission errors in wired and WLAN networks [51–59]. Almost all these existing works model the network as a single bottleneck network where the network performance depends on only the bottleneck link. Hence, these TCP models are suitable for a wired network where packet loss is rare or a WLAN network with only one wireless link. Another feature of these TCP models is that they usually consider a network with relatively large BDP and assume that the Round Trip Time (RTT) is long enough to accommodate the transmission of packets within one congestion window. The network parameter RTT is hence treated as a fixed value, which has significantly facilitated the derivation of the TCP throughput. However, with the emergence of ad hoc networks, these previously developed TCP models are no longer accurate due to inherent problems in these environments. Firstly, ad hoc networks have multiple wireless radio links which cannot be simply modelled as a single bottleneck link. Secondly, ad hoc networks have very small BDP values of only a few packets; hence the RTT cannot be simply modelled as a fixed value as like previous literature. Thirdly, packet losses due to other factors (such as channel errors, MAC layer medium contention and mobility) dominate over packet losses due to buffer overflow in ad hoc networks [5]. Recovering from these lost packets requires the cooperation of several protocol layers such as routing, MAC and TCP layers. Finally, in previous TCP models, the

fast-recovery process is always ignored or greatly simplified due to its inherent complexity. The authors in [51, 52] have developed TCP models without incorporating timeout events and assumed that any number of packet drops from one window will invoke only one Three-Duplicate-ACK event. Hence, their models are only valid for light to moderate packet loss rates. [53–55] consider timeout events in their analysis, but the fast-recovery process adopted in these models is not realistic. Specifically in [54], the authors have made the unrealistic assumption that all packets transmitted after the first lost packet in a window are also lost. With this assumption, consecutive fast-retransmits are ignored. In [53, 55], the authors assume that a timeout event is invoked when there are more than two packet losses in a window. Although they consider two consecutive fast-retransmit events resulting from two packet losses in a window, this assumption is not practical. In the model, it is assumed that the congestion window and the slow-start threshold are halved for only one time, regardless of the number of consecutive Three-Duplicate-ACK events or timeout events. In a recent paper [60], Kim et al. analyze the detailed behavior of fast-recovery process. However, they do not consider the lost packets transmitted during the fast-recovery process as well as the newly transmitted packets triggered by a successful retransmission during the fast-recovery process. These models do not account for the realistic fast-recovery process and will lead to throughput overestimation if multiple losses in a window occur frequently, especially in ad hoc networks which have multiple unstable wireless links.

So far, we find that only in a recent paper [61], a mathematical model is developed to derive the TCP throughput in an 802.11 based multi-hop ad hoc network, where packet losses are not considered in the derivation. The model leads to a formula that can be used to compute TCP throughput for a string topology. However, the derived formula is only for a two-hop string and consists of a parameter p_a which is the probability that an intermediate node will select a TCP acknowledgement instead of a TCP data to transmit. In [61], p_a appears as an unknown parameter because no method has been given to compute its value but ignored

after arguing that it is negligible compared to other terms. The feasibility of extending the two-hop formula to include more hops is not clear due to the following reasons: (a) As the number of nodes increases, there will be more unknown parameters in the formula that is used to compute the TCP throughput if the same approach is used, and (b) the asynchronous nature of packet transmissions among different wireless links will make the calculation of the mean time between any two states of the Markov Chain excessively complex. In addition, the model has a few unrealistic assumptions such as: (a) There is no contention and no binary exponential backoff at the MAC layer, (b) the transmission range and the carrier sensing range are identical, and (c) the values of the backoff counter are not uniformly but geometrically distributed within a range defined by the maximum contention window size of the 802.11 MAC protocols.

Chapter 3

The Study of TCP Performance Without Considering Wireless Channel Error

3.1 Introduction

Our work is focused on the study of packet losses due to MAC layer medium contention and wireless channel error, instead of buffer overflow and mobility. In MANETs, both wireless physical layer channel error and MAC layer medium contention can result in RTS or data transmission failures, and hence lead to false route breakages. Our investigation shows that RTS transmission failures are mainly attributed to hidden terminal effects due to MAC layer medium contention. In contrast, wireless channel error is the dominant factor which causes false route breakages due to TCP data packet transmission failures. In this chapter, we study TCP performance under the situation where there is no channel error. We analyze the impact of wireless channel error in Chapter 4.

We first develop a novel analytical model to quantify the end-to-end throughput of a single TCP flow across an 802.11 based n-hop string topology under the ideal situation where there

is no packet loss in the network. We remove all packet losses in our model by making careful and appropriate assumptions. As such, the derived TCP throughput is the upper bound and can be used as a guideline for the best case performance. The work is unique because it attempts to model the interaction between TCP's congestion window size and 802.11's MAC contention window size. The analysis shows that MAC layer medium contention can cause RTS transmission to fail frequently, which leads to a high probability of false route breakages. It is found that false route breakages due to RTS transmission failures are dependent on the size of TCP segments in a network. In particular, the likelihood of false route breakages is proportional to the size of the data packet transmitted in a network. Through simulations, we show that a 4-hop linear chain suffers the most serious hidden terminal effect which causes false route breakages.

Furthermore, we present a protocol enhancement that enables IEEE 802.11 MAC protocol to alleviate false route breakages due to RTS transmission failures. Our scheme is a simple modification to IEEE 802.11 MAC protocol and hence the problem generated at the lower MAC layer is hidden from the upper routing and transport layers. Our approach is much simpler and differs fundamentally from the proposals in [5, 39–49]. We achieve the goal of alleviating false route breakages by initiating a HELLO message to the sender whenever the number of RTS received by a receiver exceeds a threshold. We show using simulations that the proposed modification substantially reduces the false route breakages and improves throughput by up to 35%. Our results also suggest that restricting the TCP maximum transmission window to BDP of the path may not always yield good performance. This provides the basis for us to design flow control algorithms in MANETs.

The remainder of this chapter is organized as follows. In Section 3.2, we show how to compute the throughput of a TCP flow across an n-hop string topology under the ideal situation where there is no packet loss. In Section 3.3, we analyze the dependency between false route breakages, data packet sizes and path lengths between the source-destination pairs.

Section 3.4 describes the novel HELLO algorithm that we have proposed to alleviate false route breakages due to RTS transmission failures. Section 3.5 validates the analytical model and presents the comparative simulation results. Section 3.6 concludes this chapter.

3.2 Upper Bound of TCP Throughput

This section focuses on the interaction between TCP and MAC protocol, and present a general methodology for calculating the upper bound of the TCP throughput over IEEE 802.11 based multi-hop linear chain networks.

3.2.1 System Model

We consider a static n-hop string topology where $n \geq 1$ and the first hop is called the 0-th hop. The i-th hop is between node i and node $i+1$, $i = 0, 1, 2, \cdots$. For the n-hop string, a single persistent TCP flow is set up between the source node *0* and the destination node n. Node *0* is an infinite data source that always has packets to send. The size of TCP packet is small enough to be transmitted as one data packet in the MAC layer without fragmentation. The buffers of each node are infinite and FIFO-served. A single wireless channel is shared for transmissions. Since the TCP receiver has a finite resequencing buffer, it advertises a maximum window size, W_{max}, at connection setup time, and the transmitter ensures that there is never more than this amount of unacknowledged data outstanding. We assume that the user application at the TCP receiver can accepts packets as soon as the receiver can offer them in sequence, and hence the receiver buffer constraint is always W_{max}. For ease of exposition, a 6-hop string is illustrated in Fig. 3.1 as an example of the n-hop string. In the figure, as in [5, 7–10, 45], the interference range is twice and slightly greater than the transmission range. Also, the distance between any two adjacent nodes is made to satisfy the condition where one node can only transmit packets to its one-hop neighbors, can only

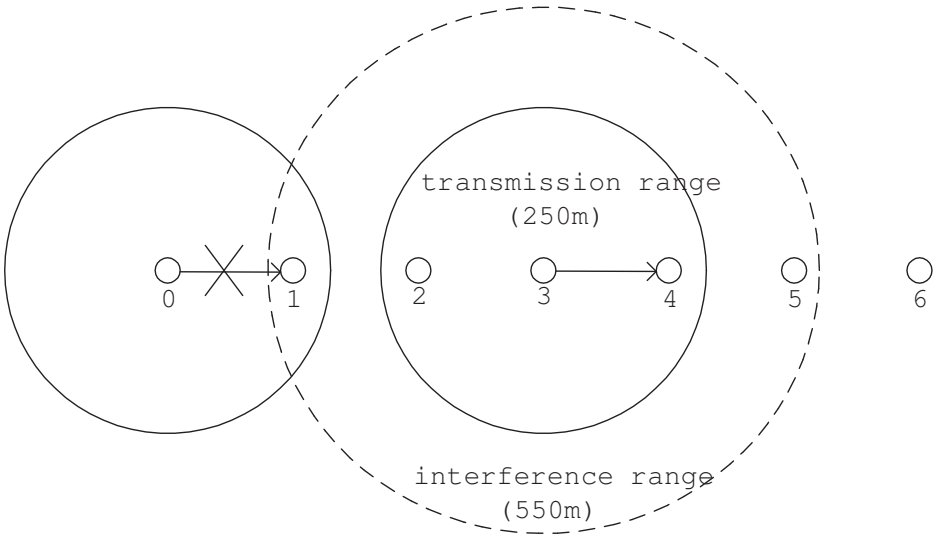


Figure 3.1: An example of an n-hop string topology. Node *3*'s transmission will interfere with node *0*'s transmission at node *1*.

interfere with its two-hop neighbors and cannot sense all other neighbors. By satisfying the condition, the string topology is likely to yield close to the maximum end-to-end throughput because a higher node density will result in more contentions and a larger round trip time.

We assume the string of nodes is located in an open space. As such, the impact to the TCP performance is mainly due to transmission collisions. The collisions occurring in a multi-hop network can be due to either simultaneous transmission attempts or the hidden terminal effect. In our system model, we only consider collisions due to hidden terminal effect and neglect those due to simultaneous transmission attempts. This is justified because the number of nodes involved in one contention is at most five, and thus the collisions due to simultaneous transmissions have little impact on the contention window size of a node.

In the literature, TCP performance in a multi-hop ad hoc network is not only dependent on the MAC protocol but also other protocols, such as routing. Routing protocol can affect the TCP throughput even in a static topology due to false routing failures which is a result of consecutive failures in accessing the channel at the MAC layer. Since our focus is on the MAC and TCP, we eliminate the impact of dynamics in the routing protocol by preventing

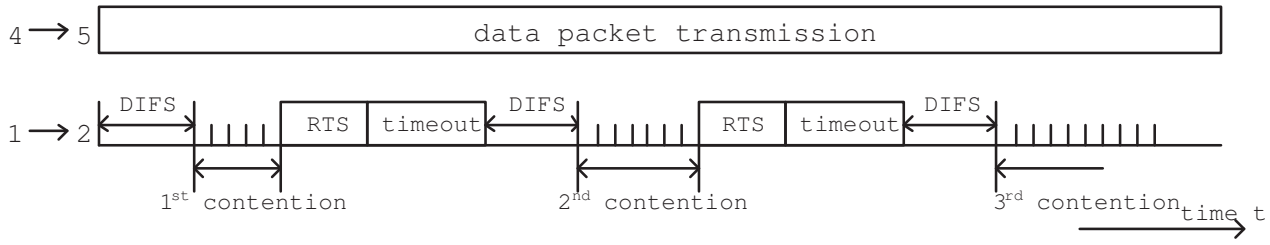


Figure 3.2: Node *1* backoff due to hidden terminal effect

the MAC protocol from issuing route failure message. This is done by setting the *short retry limit*, i.e., a 802.11 MAC layer parameter, to infinity, as opposed to the default value of seven. As such, TCP packets will not be lost due to MAC layer contention and the routing protocol will not start any re-routing in the static topology.

In our model, the effects of two-way traffic are considered. We assume that the MAC protocol has two queues at each node, one for the TCP data packet going from source node to destination node, and one for the TCP acknowledgement packet going from destination node to source node. These two queues compete with each other to access the channel.

3.2.2 TCP Throughput Analysis

With a window-based congestion control mechanism, the number of TCP packets pushed into the network by the source is equal to the TCP congestion window size. However, not all these TCP packets can be transmitted on the fly at the same time and some of the packets will be queued at intermediate nodes. In ad hoc networks, the maximum number of TCP packets in flight is determined by the wireless and spatial reuse properties and is given as the BDP of the network [7, 8, 45].

Let W_{max} denote the maximum congestion window size of the TCP flow. Then, we assume TCP packets will be queued at the intermediate nodes of a string topology only if $W_{max} >$ BDP. Without queueing, the contention window size of 802.11 evolves differently compared to the case when there is queuing. Thus, in the following two sub-sections, we will analyze

separately the TCP throughput for the two cases.

A. Without queuing: $W_{max} \leq$ BDP

In the absence of packet losses, the congestion window size of TCP increases very quickly to its maximum value, i.e., W_{max} during the startup phase and then stays at its maximum value all the times. The startup phase takes only a very short time duration compared to the time after startup and can be ignored.

The steady congestion window size implies that as many as W_{max} TCP packets can be transmitted on the fly within a time interval given by the round trip time. Let RTT(n) be the round trip time of an n-hop string topology. Then, the end-to-end throughput of the n-hop string, i.e., $\widehat{\mu}(n)$ can be written as follows:

$$\widehat{\mu}(n) = \frac{W_{max}}{\text{RTT}(n)}. \tag{3.1}$$

RTT(n) consists of two components: (a) Time required to transmit a TCP data packet beginning from the source node across n hops, and (b) time required to transmit a TCP acknowledgement packet beginning from the destination node across n hops. At each hop, the time taken to transmit either a TCP data or acknowledgement packet can be further broken down into two components: (a) Time taken to count down the backoff counter, and (b) time spent on transmitting the MAC frame exchange sequences which consists of RTS, CTS, MAC data and MAC acknowledgement. Since there is no queuing and no MAC layer contention, the contention window sizes of all nodes remain at its minimum value, i.e., M_{min}. Since the value of backoff counter is selected randomly from [0, M_{min}], the average time spent on counting down at each hop is approximately $M_{min} \times T_{slot}/2$, where T_{slot} is the duration of a time slot used in the count down process. Due to the different packet sizes, the time taken to complete a MAC frame exchange sequence for a TCP data is different from that of a TCP acknowledgement. Let T_D and T_A denote the time to complete a MAC frame exchange sequence for TCP data and TCP acknowledgement, respectively. Then, (3.1) can be rewritten

as follows:

$$\widehat{\mu}(n) = \frac{W_{max}}{n(T_D + T_A + M_{min}T_{slot})}. \tag{3.2}$$

B. With queuing: $W_{max} > BDP$

When the congestion window size of TCP becomes larger than BDP, packet queuing takes place. In our analysis, we consider all the queues are non-empty by setting a large enough maximum congestion window so that each queue will be involved in the contention. This assumption is used to facilitate the analysis of interaction between the TCP and MAC protocols and to differentiate from the case $W_{max} \leq$ BDP where there is no contention in the network. We will later show how we relax this assumption.

To compute the end-to-end throughput of a single TCP flow for a string topology, we calculate the throughput at the bottleneck link. Hence, for an n-hop string, we need to first identify the bottleneck link before computing its throughput, $\widehat{\mu}(n)$ using the following equation:

$$\widehat{\mu}(n) = \frac{1}{T_{int}(n)}, \tag{3.3}$$

where $T_{int}(n)$ is the average time interval between two consecutive transmissions at the bottleneck link of the n-hop string.

Recall that each node has more than one queue. Then, we heuristically define bottleneck link as the forward link at a node whose queue has, among all the nodes and queues, the largest average contention window size prior to a successful MAC frame exchange sequence. Let $C_{max}(n)$ denote the largest average contention window size for an n-hop string topology. Then, with the definition of bottleneck link, $T_{int}(n)$ become approximately a summation of time taken to complete a MAC frame exchange sequence at the bottleneck link and time taken to count down $\frac{C_{max}(n)}{2}$ during a backoff period since the backoff counter is uniformly distributed in [0, $C_{max}(n)$]. Let $T_d(n)$ denote the time taken to transmit the MAC frame

exchange sequence. Then, (3.3) can be rewritten as follows:

$$\widehat{\mu}(n) = \frac{1}{T_d(n) + \frac{C_{max}(n)}{2} T_{slot}}. \tag{3.4}$$

From (3.4), we observe that the problem of finding the bottleneck link has been transformed into a problem of finding $C_{max}(n)$ and $T_d(n)$.

In order to find $C_{max}(n)$, we assume the probability of a queue to complete successfully a MAC frame exchange sequence is inversely proportional to the queue's average contention window size prior to a successful exchange sequence. This assumption is especially true for several infinite sources whose transmissions are mutually exclusive and there are no outside interrupt. For example, consider one scenario where there are only three nodes in the area. These three nodes are all infinite packet sources and can each other's transmission, which means that there is only one packet in transmit at any moment. When one hears that the channel is busy, it will stop its own transmission attempt, i.e., stop counting down its backoff counter and save the current counter value. After a node finishes one packet transmission, it starts with a new backoff counter value. The other two nodes resume counting with the remained values of backoff counters. Assume that the average contention window sizes for the three nodes are respectively 1, 3 and 6. It is not difficult to conclude that the number of packets sent by each node in the long run satisfies the ratio of 6:3:1.

Recall that there are two queues for each TCP flow at each node, one for TCP data and one for TCP acknowledgement. Let P_{Di} and P_{Ai} be the respective probabilities for TCP data and TCP acknowledgement accessing the channel at node i. Similarly, C_{Di} and C_{Ai} be the respective average contention window sizes for TCP data and TCP acknowledgement at node i. Then, the assumption of inverse proportionality can be expressed as follows:

$$\frac{P_{xi}}{P_{yj}} = \frac{C_{yj}}{C_{xi}};\ x, y = \{D, A\}, \forall\ i, j, \tag{3.5}$$

such that $C_{max}(n) = \max_{i;x=\{D,A\}}\{C_{xi}\}$.

By observing our model, we find that the contention window size prior to a successful MAC frame exchange sequence from a queue (say, reference queue) depends on the transmissions from its *interference queues* and *non-interference queues*. We define *interference queues* and *non-interference queues* as follows. We call the downlink queues that are 3 hops away from the reference queue as *interference queues*, since packets transmitted from the reference queue suffer collisions only from its downlink queues that are 3 hops away, which will cause the contention window of the queue to increase. As shown in Fig. 3.1 and Fig. 3.2, node *1* suffers several continuous backoffs due to the packet transmissions of its *interference queue* at node *4*. The evolution of the contention window reacts differently to the TCP data and TCP acknowledgement transmissions of node *4* which have different packet size. We refer to those queues whose transmissions will stop node *1*'s transmission attempts as *non-interference queues* to the reference queue, such as those queues at nodes *0*, *2* and *3*. We do not take into account the queues that do not have any overlapping transmissions with the transmitting queue, such as queues at node *5* and at those nodes with larger indexes.

Let k denote the total number of TCP packets transmitted from *interference queues* and *non-interference queues* between two packet transmissions at the reference queue. Then the probability that the contention window size is affected by these k packets is given by:

$$g(\gamma_1, \gamma_2, \gamma_3) = \frac{k!}{\gamma_1!\gamma_2!\gamma_3!}\phi^{\gamma_1}\varphi^{\gamma_2}\alpha^{\gamma_3}, \tag{3.6}$$

where $\gamma_1 + \gamma_2 + \gamma_3 = k$; γ_1 and γ_2 are respectively the number of TCP data packets and TCP acknowledgement packets transmitted from the *interference queues*; γ_3 is the number of packets transmitted from the *non-interference queues*. The reactions of the contention window size of the reference queue are categorized into three different ways. The probabilities of the three reactions are Binomially distributed, and are given as ϕ, φ and α, respectively. Based on ϕ and φ, $\alpha = 1 - P_{xi} - \phi - \varphi$. Note that ϕ, φ and α are not the probabilities that three different kinds of packets are transmitted from corresponding nodes. As shown in Fig. 3.1, node *0* and node *4* may access the channel concurrently so that transmission

attempt of node *1* is stopped by node *0* rather than is affected in the way shown in Fig. 3.2.

Based on the analysis above, for a reference queue at node i,

$$C_{xi,x=\{D,A\}} = M_{min} \times P_{xi} \times \sum_{k=0}^{+\infty} \sum_{\gamma_1+\gamma_2+\gamma_3=k} g(\gamma_1,\gamma_2,\gamma_3) \times \beta, \tag{3.7}$$

where β indicates the number of exponential backoffs at the reference queue due to *interference queues* and is given as follows:

$$\beta = \min\{2^{n_1+n_2+\cdots+n_{\gamma_1}} \times 2^{m_1+m_2+\cdots+m_{\gamma_2}}, 2^5\}, \tag{3.8}$$

where n_i is the number of exponential backoffs within the i^{th} TCP data packet transmission duration, m_i is the number of exponential backoffs within the i^{th} TCP acknowledgement transmission duration. ϕ and φ are derived and given as follows:

$$\begin{aligned} \phi &= \frac{P_{D(i+3)}(1-P_{N(i-2)}-P_{N(i-1)}-P_{Ni}-P_{N(i+3)})}{1-P_{N(i+3)}-P_{xi}}, \quad \text{and} \\ \varphi &= \frac{P_{A(i+3)}(1-P_{N(i-2)}-P_{N(i-1)}-P_{Ni}-P_{N(i+3)})}{1-P_{N(i+3)}-P_{xi}}, \end{aligned} \tag{3.9}$$

where $P_{Ni} = P_{Di} + P_{Ai}$; the subscript $x(i+j)$ is defined as the downlink queue that is j hops away from the reference queue xi instead of the index of a node.

Observing (3.7), the contention window size of a reference queue can be expressed as a function of probabilities. By combing (3.5) and (3.7), we can then get a system of equations whose parameters are the probabilities of each queue accessing the channel. By solving these probabilities, we can calculate the average contention window size of each queue based on (3.7) and hence get $C_{max}(n)$ of the bottleneck link. However, to solve these probabilities, we need additional equations. In the following, we analyze 3 different cases of an n-hop string network, and present the approaches to solve the probabilities of all the queues accessing the channel as well as $T_d(n)$.

(i) Case 1: $1 \leq n \leq 3$

There is only one packet allowed on the fly along the path at any time. Therefore, the average

interval $T_d(n)$ taken to transmit a MAC frame exchange sequence from the source is given by:

$$T_d(n) \quad = \quad n(T_D + T_A) \qquad 1 \leq n \leq 3. \tag{3.10}$$

In *1*-hop and *2*-hop cases, all nodes can hear each other's transmissions and hence defer their transmission attempts, i.e., there is no *interference node* associated with any queue in the network. Collisions never occur in these two cases since we neglect collisions due to simultaneous transmission attempts. Therefore, the maximum contention window size of these two cases is equal to M_{min}. In the *3*-hop case, in order to obtain $C_{max}(n)$, we need one additional equation to solve the probabilities of all queues accessing the channel. Since only one packet is allowed on the fly, the probability of each node accessing the channel is mutually exclusive. Hence, the sum of the probability that each node accesses the channel satisfies:

$$\sum_{i=0}^{3} P_{Ni} = 1, \quad \text{where} \quad P_{Ni} = P_{Di} + P_{Ai}. \tag{3.11}$$

(ii) Case 2: $4 \leq n \leq 7$

In contrast to the case when $1 \leq n \leq 3$, two packets can be transmitted concurrently along the path in this scenario. More queues are affected by their corresponding *interference nodes*. We use the following equation to calculate $T_d(n)$:

$$T_d(n) = 4T_D + 3T_A + T_I(n). \tag{3.12}$$

where $T_I(n)$ varies with different network lengths.

First, we look at the case where $n = 4$. We define nodes *0*-*3* as a subsystem. This is because only one packet can reside in this subsystem at any one time. For the *4*-hop case, the scenario where there can be two packet transmissions is shown in Fig. 3.3, which is also the only possible scenario where more than one packet can be transmitted simultaneously. If there is always one packet transmission in the subsystem, in a steady state, $T_d(4)$ between two consecutive packets will be $4T_D + 3T_A$. However, in the scenario where no packet transmission in the subsystem occurs, the subsystem is considered to be in an idle state. This occurs when

Figure 3.3: Two concurrent transmission in a 4-hop chain. The average interval $T_d(4)$ between two consecutive packet transmissions from source node *0* is given by $\sum_{i=0}^{3} T_{Di,i+1} + \sum_{i=3}^{1} T_{Ai,i-1}$, where $(i, i+1)$ or $(i, i-1)$ means a packet is transmitted from node i to node $i+1$ or to node $(i-1)$.

node *4* transmits ACK to node *3*. Node *1* may access the channel first to send RTS and hence stop node *0*'s transmission attempt. Furthermore, node *1* fails to send RTS to node *2* due to the hidden terminal effect. This leads to the idle state of the subsystem and results in the increase of the interval between two consecutive packets. It takes place with probability $P_{N4}\frac{P_{D1}}{P_{N1}+P_{N0}}$, which indicates that DATA queue at node *1* accesses the channel to transmit one RTS with probability $\frac{P_{D1}}{P_{N1}+P_{N0}}$ when node *4* is transmitting one ACK to node *3* with probability P_{N4}. Then, the subsystem in an active state with packet transmissions takes place with probability $P_{N0} + P_{N1} + P_{N2} + P_{N3}$. Hence, we can obtain an additional equation as follows and solve the probabilities of all the queues:

$$\sum_{i=0}^{3} P_{Ni} + P_{N4}\frac{P_{D1}}{P_{N1} + P_{N0}} = 1. \tag{3.13}$$

From the viewpoint of a packet in the subsystem, the probability that the subsystem is in an idle state is $\frac{P_{N4}P_{D1}}{(P_{N1}+P_{N0})(P_{N0}+P_{N1}+P_{N2}+P_{N3})}$. Hence $T_I(4)$ is given by:

$$T_I(4) = \frac{P_{N4}P_{D1}}{\sum_{i=0}^{1} P_{Ni} \sum_{i=0}^{3} P_{Ni}} T_{timeout} + o(\cdot), \tag{3.14}$$

where $o(\cdot)$ denotes that we ignore node *1*'s continuous winning of the contentions and capturing the channel. $T_{timeout}$ is the expected time to receive CTS from the instant when RTS is sent out, i.e., $T_{timeout} = T_{rts} + T_{cts} + T_{difs} + T_{sifs}$.

Similarly, based on the concept of the subsystem, we can derive additional equations that

can be used to calculate $C_{max}(n)$. Here, we still look at the subsystem from node *0* to node *3*, and derive the expressions for $T_I(n)$ for other cases where $5 \leq n \leq 7$ as follows:

$$T_I(5) = (\frac{P_{N4}P_{D1}}{\sum_{i=0}^{1} P_{Ni}} + \frac{P_{N5}P_{D2}}{\sum_{i=0}^{2} P_{Ni}})\frac{T_{timeout}}{\sum_{i=0}^{3} P_{Ni}} + o(\cdot), \quad \text{and} \tag{3.15}$$

$$\begin{aligned} T_I(6) = T_I(7) \quad = \quad & (\frac{P_{N4}P_{D1}}{\sum_{i=0}^{1} P_{Ni}} + \frac{P_{N5}P_{D2}}{\sum_{i=0}^{2} P_{Ni}} + \\ & \frac{P_{N6}P_{D3}}{\sum_{i=0}^{3} P_{Ni}})\frac{T_{timeout}}{\sum_{i=0}^{3} P_{Ni}} + o(\cdot). \end{aligned} \tag{3.16}$$

Although $T_I(6)$ and $T_I(7)$ have the same expression of a function, the values of P_{N6} are different. For the case where $n = 6$, node *6* has only one queue to store the acknowledgement packet. In contrast, there are two queues in node *6* for the case where $n = 7$.

(iii) Case 3: $n > 7$

If we use the method given above, the end-to-end throughput of all the $n-$hop cases can be calculated through derivation of $T_d(n)$ and $C_{max}(n)$. However, as the number of hops increases, the computations will become more complex. The number of unknown variables during the derivation is $2(n-1)$. Here, we approximate the throughput of the chain with more than 7 hops to the throughput of a 7-hop chain. The reason is as follows. As shown in Fig. 3.4, we separate the multi-hop linear chain into several subsystems. In previous analysis, we have discovered that transmissions in the second subsystem affects the first subsystem directly and hence lead to the idle period $T_I(n)$ of the first subsystem. By observation of the equations of $T_I(n)$ $(4 \leq n \leq 7)$, $T_I(n)$ is a function of parameters in the first two subsystem and has no relationship with other subsystems, even though n continues to increase. That is to say, other subsystems have no direct impact on the first subsystem. The interval $T_d(n)$ in the first subsystem increases due to the packet transmission in the second subsystem. The packet transmission in the third subsystem results in the increase of the idle duration of the second subsystem, and hence leads to the decrease of the idle duration of the first subsystem. Similarly, the fourth subsystem will increase the $T_d(n)$ of the first subsystem, and the fifth

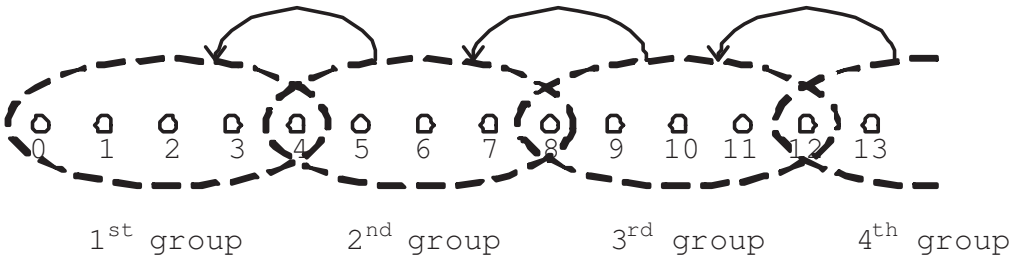


Figure 3.4: $n > 7$ hops case

one will decrease the $T_d(n)$ of the first subsystem, and so on. As such, all subsystems except for the second one will indirectly increase or decrease the idle duration of the first subsystem. The idle duration of the first subsystem in a 7-hop chain is already very small compared to $4T_D + 3T_A$. The indirect impact of other subsystems is also relatively small even though n increases to a large value. Hence, the overall impact of other subsystems is quite insignificant and can be ignored.

3.3 Study on False Route Breakage due to RTS Transmission Failures

In the model above, we make a strong assumption that the *short retry limit* in IEEE 802.11 is infinite, so that false route breakages due to RTS transmission failures never happen. Our analysis in Section 3.2 shows that RTS transmission failures are dependent on TCP segment size, which is well illustrated in Fig. 3.1 and Fig. 3.2. In the figures, node *1* suffers several continuous RTS transmission failures due to the long data packet transmissions of node *4*. If node *4* can capture the channel in the following data packet transmission, node *1* may continue to fail in retransmitting the RTS packet. Following this logic, node *1* has a large probability to fail to transmit the RTS seven times and wrongly assume that the route has broken. Fig. 3.2 shows that the number of RTS transmission failures due to the hidden terminal effect increases with the data packet size. It can be seen that the likelihood of false

route breakages is proportional to the data packet size in the network. In this section, we further investigate the property of false route breakages through simulation.

It is known that IEEE 802.11 may wrongly assume that a route is broken based on a certain number of failures in transmitting either a RTS packet or a data packet. However, in most cases, a route breakage is detected via the failure to retransmit a RTS packet rather than a data packet, because a data packet can access the channel only if a RTS packet has been successfully transmitted. Therefore, in this chapter, we focus on the study of false route breakages due to RTS transmission failures.

Both wireless physical layer channel error and MAC layer medium contentions can lead to RTS transmission failures. The wireless transmission channel is greatly subjected to noise, fading and interference, which can corrupt RTS packets. Consequently, these RTS cannot be correctly decoded at the receiver. In the case of MAC contention, the RTS transmission failures are attributed to either the interferences due to hidden terminal effects or the collisions due to simultaneous transmission attempts from at least two nodes. In the network with a bandwidth of 2Mbps, the relationship between the maximum number of RTS failures during one data packet transmission and the size of that data packet is calculated and given in Table 3.1. It can be seen that RTS transmissions fail with a high probability for a large TCP data packet size. In contrast, the RTS failures due to both wireless channel error and simultaneous transmission attempts are independent of the sizes of data packets transmitted in a network. Since all the retransmissions of a RTS packet are independent events, with the exception of scenarios with extreme conditions (i.e., a channel with a bad quality for a long duration or extremely heavy traffic load in the network), the probability that a false route breakage occurs due to seven consecutive independent transmissions of a RTS packet is small.

We further verify this through simulations using GloMoSim with a static chain topology which has a bandwidth of 2Mbps. TCP and DSR are used as the transport protocol and the routing protocol respectively. There is no channel error. The radio propagation model

Table 3.1: Maximum Number of RTS Failures with One DATA Packet Transmission

Packet Sizes (bytes)	Number of RTS Failures
250	1
500, 750	2
1000 , 1250	3
1500 ,1750 , 2000	4

uses Friss free-space attenuation ($1/r^2$) at near distances and an approximation to two ray ground ($1/r^4$) at far distance by assuming specular reflection off a flat ground plane. The distance between any two adjacent nodes is 300 meters. In GloMoSim, transmission range and interference range of the wireless radio are 367 meters and 670 meters respectively. Each simulation is run for 300 seconds. The simulation results in Fig. 3.5 show that for a static string topology with $n = 1$ to $n = 10$ hops, the total number of route breakages varies with the TCP packet size in a 802.11 based ad hoc network. The packet size varies from 250 bytes to 2000 bytes at a step of 250 bytes. Note that the route breakages here are all false since our network model is a static chain. It can be seen that for the cases of $n > 2$, the simulation results are consistent with our analysis in that the number of route breakages increases with the increase of packet size. In the short chains (i.e., 1 to 2 hops), false route breakages in the chain never take place. This is because all nodes in the chain can hear each other's transmissions and hence defer their transmission attempts. There is no medium contention due to hidden terminal effects. The collisions due to simultaneous transmission attempts is not significant enough to result in seven consecutive failures in transmitting a RTS, since the number of nodes involved in the contention is only at most three.

The results also show that for the cases where the packet size exceeds 500 bytes, the number of false route breakages increases initially with the increase in chain length and peaks when the chain length is 4. When the number of hops exceeds 4, the number of route breakages

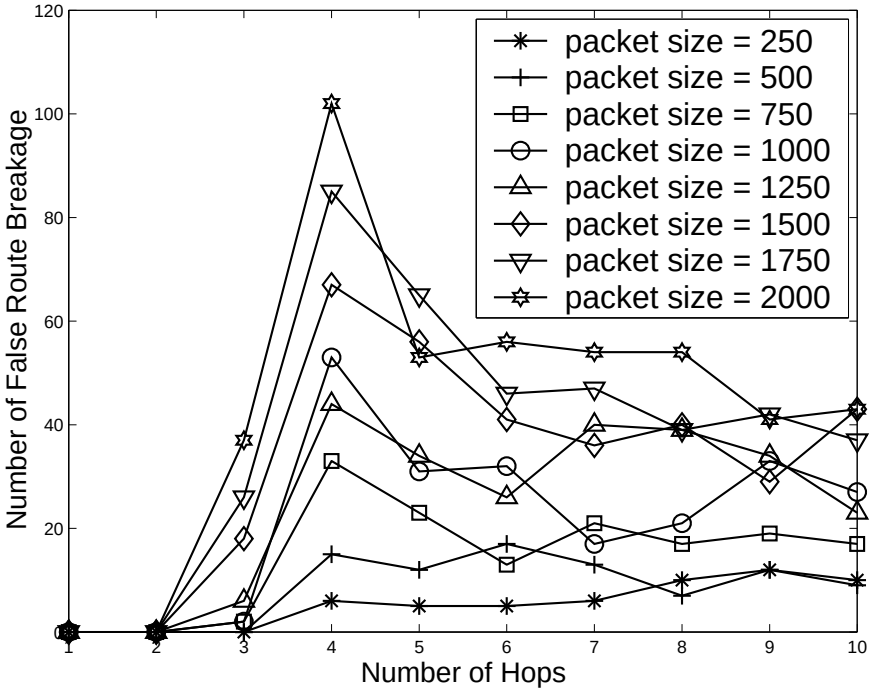


Figure 3.5: Number of false route breakages in linear chains

fluctuates, indicating that the 4-hop chain suffers the most serious medium contention. This phenomenon can be attributed to the effects of two network parameters, the total number of nodes involved in the transmissions and the contention among the nodes. The number of route breakages for the cases of $n \geq 4$ hops is greater than that for the case of $n = 3$ hops, because the total number of route breakages increases with the increase in the number of nodes involved. However, for the cases of $n \geq 4$ hops, as the number of hops increases, the nature of spatial reuse in wireless networks can lead to better balanced and distributed data transmissions among all the nodes. This reduces the medium contention in the network and consequently reduces the probability of false route breakages happening. For example, in a 3-hop chain, when node *0* is in the transmission, node *3* will keep trying to send a RTS packet to node *2*. However, in a 5-hop chain, both node *4* and node *5* will compete with node *3* to access the channel. Their transmissions can be heard by node *3*, and hence node *3* will stop its own RTS transmission attempt. This prevents node *3* from continuous consecutive failures

to send RTS to node *2*. Thus, the false route breakage probability at node *3* is reduced.

3.4 The HELLO Scheme

In this section, we present a simple modification to IEEE 802.11 MAC protocol to mitigate its misbehavior when discovering a route breakage. We name it the HELLO scheme, because a HELLO control message is introduced into the 802.11 protocol as described in Algorithm 1. The main idea of the scheme is that the receiver continuously records the number of RTS received from each individual sender. If the number of received RTS from one sender exceeds a certain threshold, we can assume that the link is inclined to be a breakable link. The receiver will then try to inform the corresponding sender of its existence by initiating a HELLO control message. This HELLO message is sent after the medium is sensed idle for a short period of time known as the Priority Inter-Frame Space (PIFS). The PIFS value is calculated as a SIFS plus one slot time and meets the condition SIFS $<$ slot time $<$ PIFS $<$ DIFS, where DIFS is the sum of a SIFS and two time slots. Note that this PIFS value already exists for high priority messages in the PCF mode of 802.11 standard. Furthermore, the random backoff timer is not needed for the transmission of HELLO message. Hence, the HELLO message has higher priority to access the channel than other packets. If the sender receives the HELLO message, it will reset its counter for RTS transmission failures to zero, and the corresponding contention window size is also reduced to the minimum. The RTS record at the receiver has a lifetime for which it is valid. The record will be discarded after the lifetime has expired.

The basis for not adopting the random backoff timer for a HELLO message transmission is as follows. First, the collisions due to simultaneous HELLO transmission attempts from several nodes are rare. For example, in Fig. 3.6, node G and F may try to transmit packets to node H and A respectively. The RTS transmissions from node *3*, G and F are mutually

Algorithm 1: the HELLO Scheme

Sender

Start:

if RTS failure **then**

Counter = Counter + 1, backoff contention window;

end if

if CTS is received **then**

send data;

else if overhear sender's transmissions or HELLO is received

Counter = 0; reset contention window;

else try to send RTS;

end if

goto Start;

Receiver

Start:

if RTS is received **then**

if Channel idle for SIFS **then**

Counter = 0, send CTS;

else Counter = Counter + 1;

end if

end if

if Counter > Threshold and Channel idle for PIFS **then**

Counter =0, send HELLO;

end if

if Counter timeouts **then**

Counter = 0;

end if

goto Start;

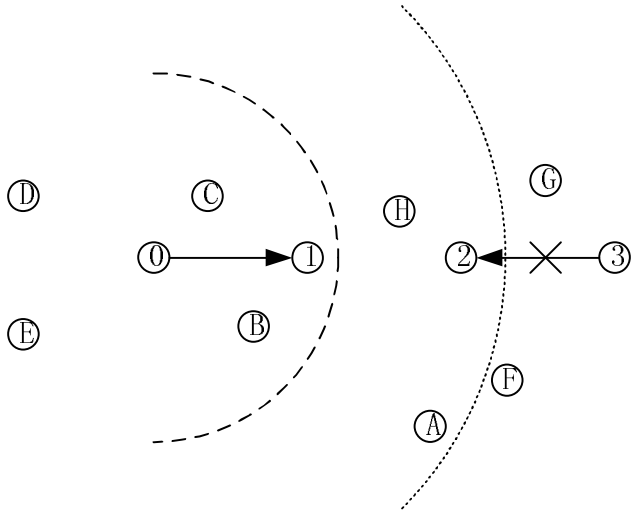


Figure 3.6: An example of ad hoc networks: node *1*, *B* and *C* are in the transmission range of node *0*; node *2*, *A* and *H* are in the interference range of node *0*;

exclusive and their transmissions can stop one another's transmission attempt. If the RTS threshold is 3, node *2*, *H* and *A* can transmit HELLO messages at the same time only if each of them has received 4 RTS packets. This case can hardly happen. In addition, even though the HELLO messages are corrupted, the RTS senders may also learn of the existence of its receivers in the following transmissions before seven consecutive RTS transmission failures happen. Secondly, the backoff timer is a random value which cannot guarantee that the HELLO message is sent in time with a high priority. In IEEE 802.11, the random backoff timer is set as follows: backoff timer = $Random() \times$slot time, where $Random()$ is the pseudo random integer drawn from a uniform distribution between 0 and contention window. The backoff timer may decuple the PIFS or even DIFS and the PIFS is negligible as compared to the backoff timer. As such the PIFS cannot guarantee that a higher priority is being assigned to the HELLO message if the backoff timer is used.

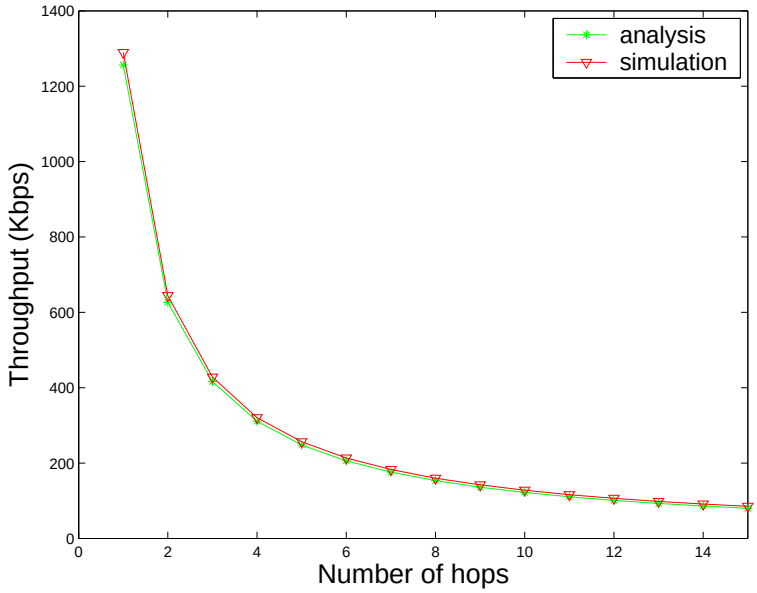


Figure 3.7: TCP-Reno throughput: $W_{max} \leq BDP$, $W_{max} = 1$

3.5 Simulation and Validation

In order to validate the analytical model and to evaluate the HELLO scheme, we run simulations using GloMoSim. All nodes communicate with identical, half-duplex wireless radios which have a bandwidth of 2Mbps. There is no physical layer channel error.

3.5.1 Validate Analytical Model

In the string topology, AODV protocol without *Hello* messages is used to avoid the complexity of computing the routing protocol overhead. The packet size varies from 250 bytes to 2000 bytes at a step of 250 bytes. We run each simulation for 300 seconds to get the average throughput and the average contention window size of each queue with varying number of hops between the TCP source and destination. Due to the similarities for all the cases of different packet sizes, we only show the results for the case that the packet size is 1500 bytes.

Fig. 3.7 and Fig. 3.8 show the results of the end-to-end throughput for the two situations

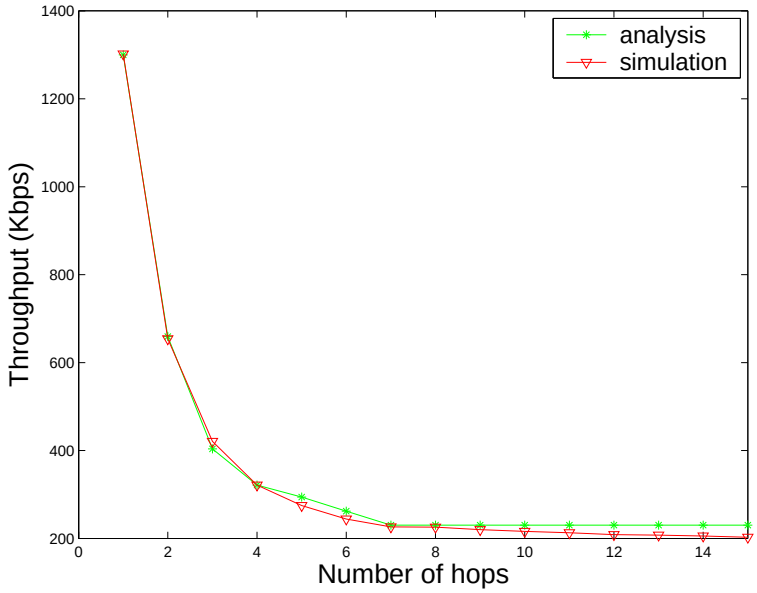


Figure 3.8: TCP-Reno throughput: $W_{max} > BDP$, $W_{max} = 64$

where $W_{max} \leq BDP$ and $W_{max} > BDP$. The maximum congestion window size is set to 1 and 64 packets respectively in the Fig. 3.7 and Fig. 3.8. We plot the average throughput against the number of hops, where the number of hops varies from 1 to 15 hops. The simulation results and analytical results match closely. By observation of the simulated data in $W_{max} > BDP$ case, for $n \leq 7$ hops, the difference of the throughput between n and $n-1$ linear chain is always larger than 10kbps. However, for $n > 7$ hops, the throughput decreases very slowly as the number of hops become large. Consequently, the difference between n and $n-1$ linear chain is small and always less than 10kbps. For instance, the throughput is 224.7kbps for the 7-hop chain and 190.98kbps for the 80-hop chain. The throughput decreases at a rate of less than 0.5kpbs per hop. Therefore, it is reasonable to approximate the end-to-end throughput of $n > 7$ hops chain to that of $n = 7$ hops chain. This approximation provides an estimation of the closed upper bound of the throughput for $n > 7$ hops chain.

We also compare the simulation and analytical results of the average contention window size at different queues. Here we show the results for only two scenarios where $n = 3$ and $n = 4$, as shown in Fig. 3.9 and Fig. 3.10. We can see for those queues which there are no *interference*

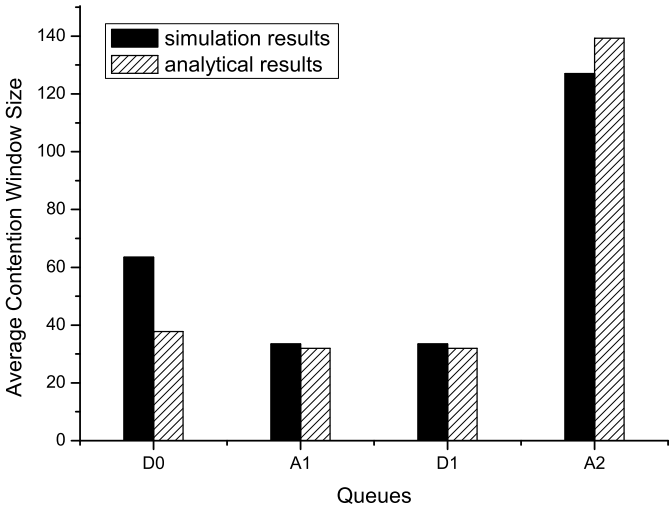


Figure 3.9: Average contention window sizes of different queues: $n = 3$

queues associated with, the analytical average contention window sizes are quantitatively correct and remain around the minimum value of 32. For those queues there are *interference queues* associated with, the analytical results qualitatively reflect the magnitude of the average contention window sizes. The reason for the discrepancy is due to our assumption that there are two separate queues at each node and every node in the network is always involved in the contention.

Remarks: In our analysis, we investigate only two situations, where (i) the maximum congestion window size W_{max} is small enough so that there is no contention in the network, (ii) the maximum congestion window size W_{max} is large enough so that all the buffers are non-empty and are involved in contention in the network. For the first situation, we sum up the time taken to count down the backoff counter at all queues when calculating the interval between packet transmissions. For the second situation, we only need consider the time taken to count down the backoff timer counter at one bottleneck link. The reality is actually the combination of these two situations. However, we can still get accurate analytical throughput

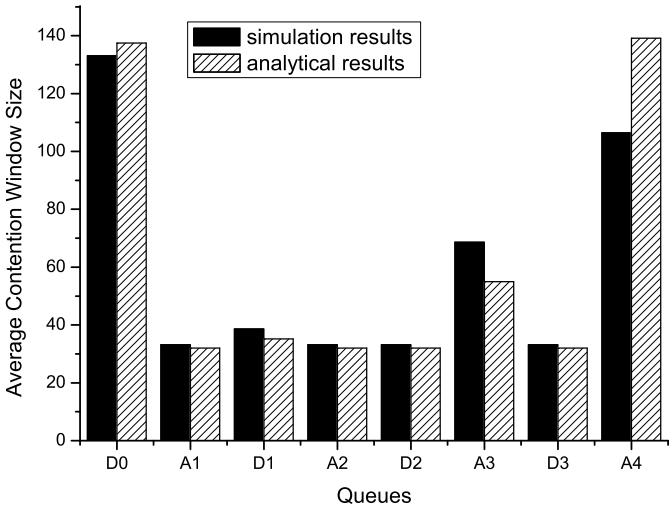


Figure 3.10: Average contention window sizes of different queues: $n = 4$

to a large extent as shown in Fig. 3.7 and Fig. 3.8. The reason is that the time taken to count down the backoff timer does not have significant impact on the throughput compared to the time taken to transmit MAC frames. This is also reflected by the simulation results. When n varies from 1 to 4, the variance of the throughput shows a steep curve. This is because the time interval between two consecutive packet transmissions increases at the rate of the large MAC frame transmission. When $n > 4$, the variance of throughput is mild and depends on the short time taken to count down the contention window as well as the short idle duration defined in our analysis.

3.5.2 Evaluate the HELLO Scheme

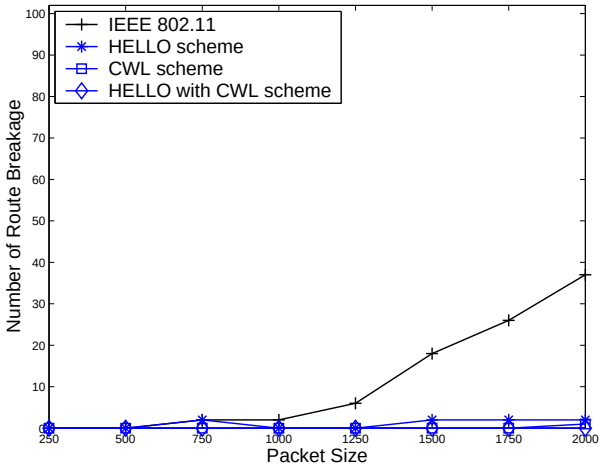
In this section, we run simulations using Glomosim in both static chain topologies and random movement scenarios. DSR is used as the routing protocol. In each simulation run, we measure the throughput and the total number of route breakages of the network. The mobile network we simulate consists of 45 nodes randomly placed in a field at the beginning of a simulation. Two fields with different areas of $1000m \times 1000m$ and $2000m \times 2000m$ are simulated. We

utilize a mobility pattern based on the random waypoint model with speeds within the range of [0, v] m/s, where v varies among 1, 5 and 10. The number of TCP flows changes from 5 to 40 at a step of 5. The TCP packet size is 1000 bytes. Each simulation is conducted for 1000 seconds. Each data point shown gives the average of 10 simulations.

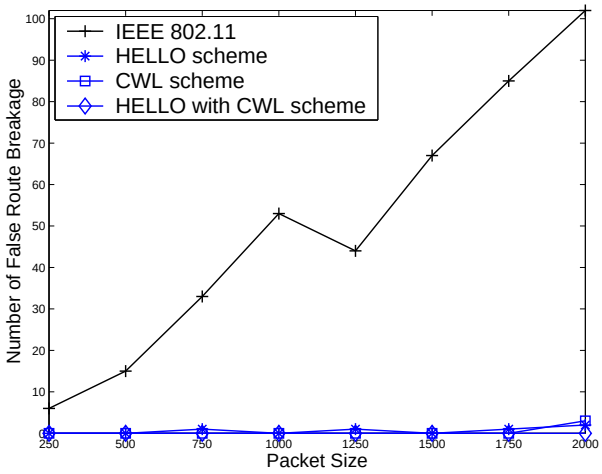
We introduce one of the existing work in the literature [45] which will be used in our comparative performance study. In [45], it was investigated that given n hops in a linear chain, only approximately $\lceil \frac{n}{4} \rceil$ nodes can transmit packets simultaneously due to the spatial reuse nature of wireless channel. This allowed the authors to conclude that there is an optimal size for the maximum transmission window of TCP which is close to the value of $\lceil \frac{n}{4} \rceil$, i.e., the BDP of the path. Any increase in the maximum window size results in more congestion and medium contention in the networks. Their proposed scheme attempts to limit the maximum TCP transmission window to the BDP of the path. Since the path length between the source-destination pair may vary due to node mobility in an ad hoc network, this scheme involves the modification of the TCP and routing layers to allow an adaptive change of the maximum TCP transmission window based on the path length. For brevity, we call this methodology the CWL (Congestion Window Limit) scheme. In the study, we run the simulations for four different cases, the original IEEE 802.11 protocol, the HELLO scheme, the CWL scheme and the combination of the HELLO scheme and the CWL scheme. The threshold of the HELLO scheme is set to 3 in the simulations.

A. Static Chain Topology

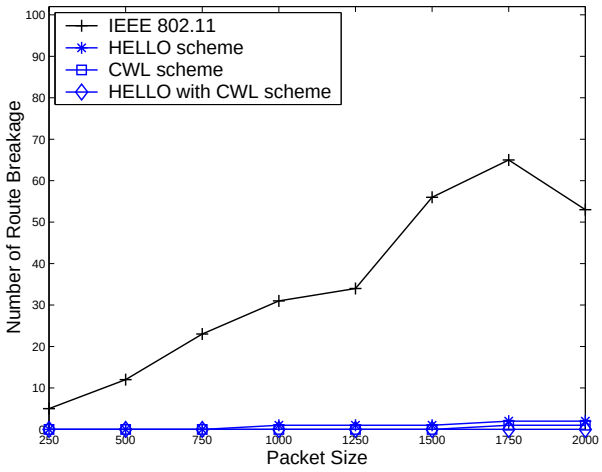
The simulation results in Fig. 3.11 illustrate how the three schemes can reduce the number of false route breakages for different chain lengths and packet sizes in a static string topology. Here, we only show the results for n =3, 4 and 5, as similar results are obtained for other topologies. In all the three sub-figures, the number of false route breakages is reduced substantially and approximates to zero for all the three schemes. However, in Fig. 3.12, which



(a) $n = 3$

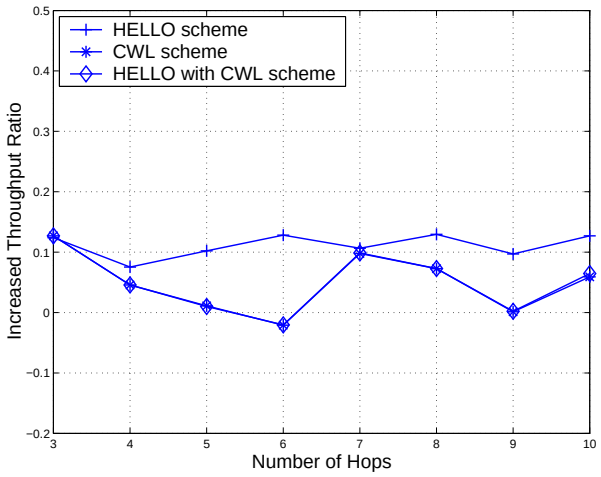


(b) $n = 4$

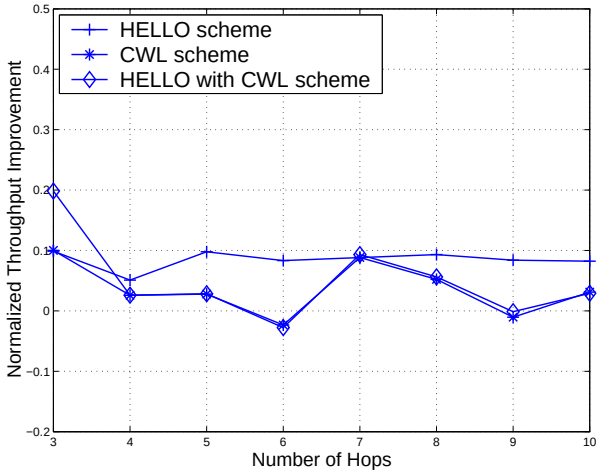


(c) $n = 5$

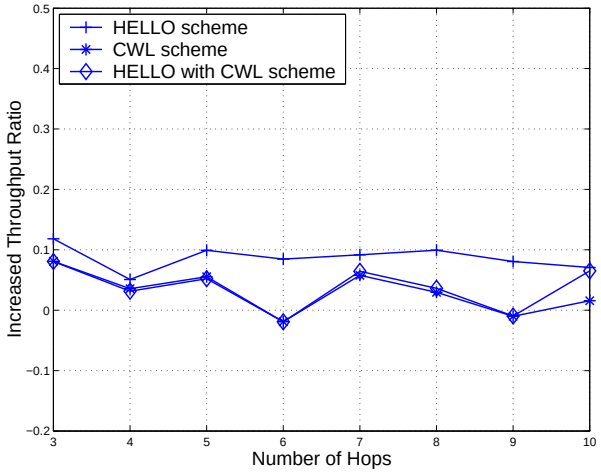
Figure 3.11: Improvement for number of false route breakages in linear chains



(a) Packet Size: 500 bytes



(b) Packet Size: 1000 bytes



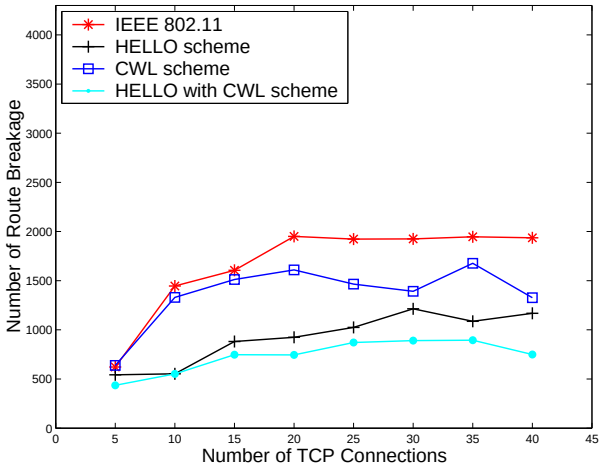
(c) Packet Size: 1500 bytes

Figure 3.12: Increased throughput ratio in linear chains

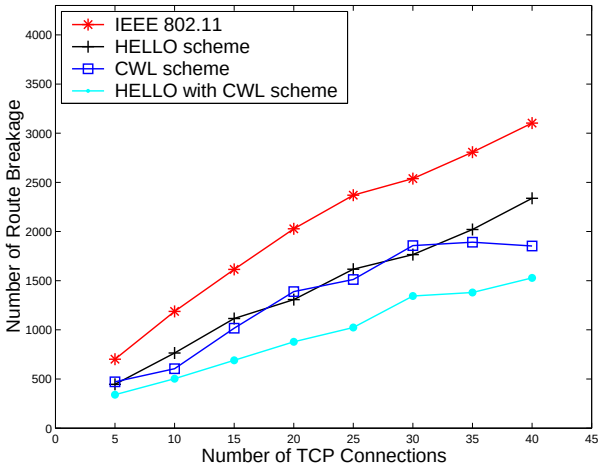
shows the performance improvement in the throughput among the three schemes for packet sizes = 500, 1000, 1500 bytes, it can be seen that our HELLO scheme works better than the other two schemes. The proposed modification improves the throughput performance even with different number of hops and the throughput is increased by up to 13%. However, the other two schemes degrade the throughput at certain points. This can be adequately explained by the small transmission window used by the CWL scheme and the HELLO with CWL scheme. Although the small transmission window is equal to the BDP of the path, the CWL algorithm cannot guarantee that the BDP packets are evenly distributed and transmitted in the network. It is possible that all the BDP packets might queue at one node and the rest of the nodes remain idle due to lack of data packets. Hence, network resource is not utilized effectively. However, our HELLO scheme with a large transmission window injects a large amount of packets into the network, which can guarantee that almost every node is ready to transmit packets rather than remains in an idle state. This reduces the time spent on processing the data packets sequentially at each node because the processing can now be done concurrently in all the nodes. It also avoids the possible wastage of the bandwidth resource due to non-fully utilized networks.

B. Mobile Ad Hoc Networks

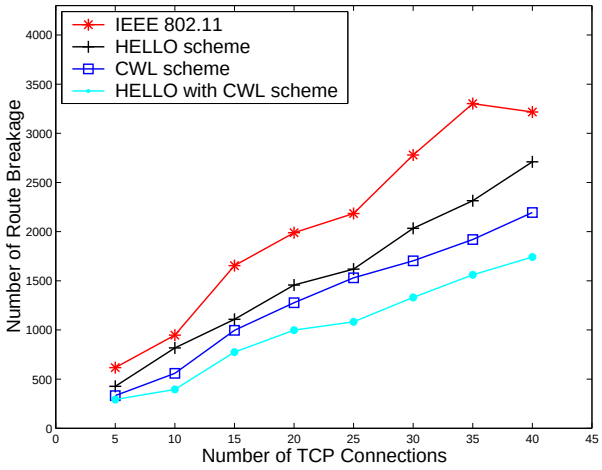
Fig. 3.13 shows the simulation results of route breakages for four cases in the mobile ad hoc network. Note that here the number of route breakages is the sum of genuine route breakages caused by node mobility and false route breakages caused by medium contention. It can be seen that, in all the sub-figures, as compared to the 802.11, the three proposals can reduce the number of route breakages to some extent. Specifically, the HELLO scheme can reduce the number of route breakages by up to 63%. The least number of route breakages is obtained with the HELLO with CWL scheme. It can also be found that the HELLO scheme has fewer route breakages than the CWL scheme in scenarios with low mobility. As the speed increases,



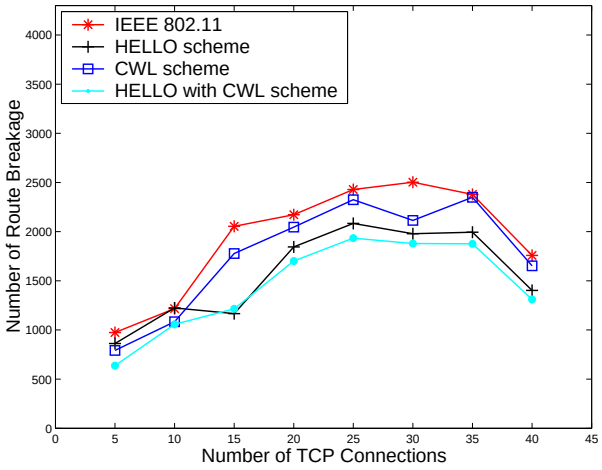
(a) 1 m/s, 1000 × 1000



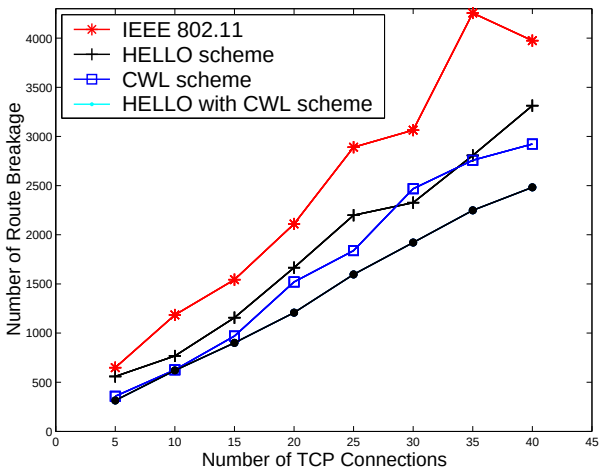
(b) 5 m/s, 1000 × 1000



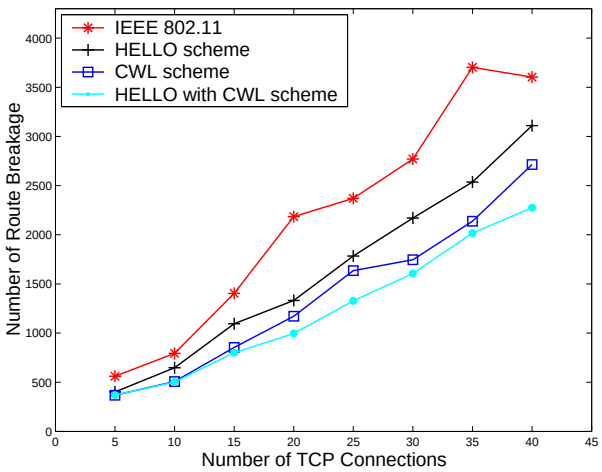
(c) 10 m/s,1000 × 1000



(d) 1 m/s, 2000 × 2000



(e) 5 m/s,2000 × 2000



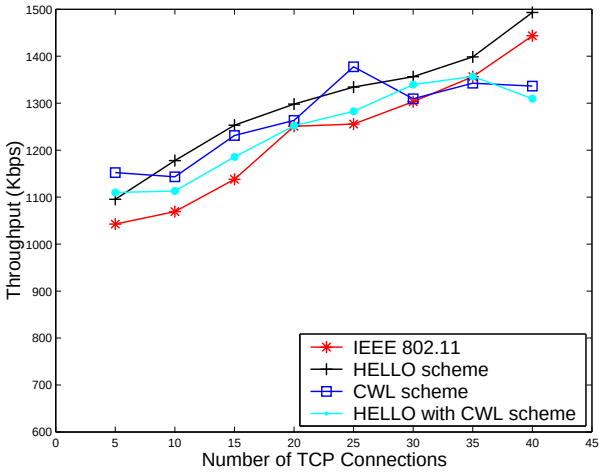
(f) 10 m/s,2000 × 2000

Figure 3.13: Improvement for route breakages in mobile networks

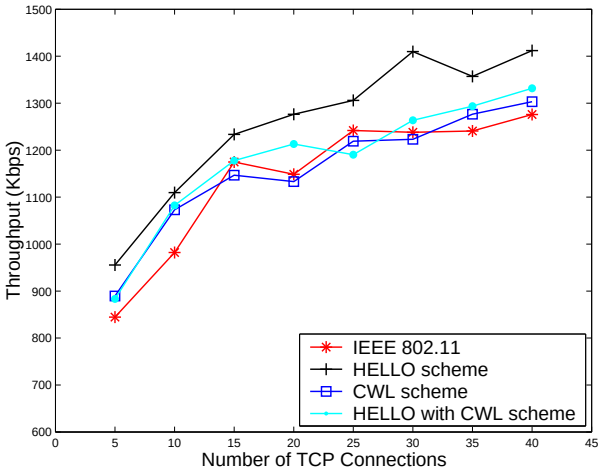
the HELLO scheme has more route breakages than the CWL scheme. This is because for the CWL scheme, the network has only a few route breakage detections due to a small number of packet transmission attempts with the small CWL transmission window.

Fig. 3.14 shows the corresponding simulation results of throughput for the four cases in a mobile ad hoc network. The HELLO scheme has the best performance among the four cases. The proposed modification increases the throughput by up to 35%. The other two schemes do not improve the throughput all the time. They improve the throughput only in the low mobility networks without high traffic load and cause the throughput to deteriorate in the other scenarios. Besides the reasons we have given previously in the linear chain topology scenarios, the nature of TCP protocol can better account for this throughput degradation. As TCP is more likely to timeout with a small transmission window, TCP can fast recover a packet loss only if it receives at least three duplicated acknowledgements. However, the BDP of the path is usually too small to allow the receipt of three duplicate acknowledgements. Once a TCP data packet or a TCP acknowledgement is lost, TCP has to wait until the timeout period to detect the packet loss since it cannot receive sufficient duplicate acknowledgements. The probability of packet losses is high in a mobile network and this can easily create a significant number of timeouts to TCP flows. Furthermore, for schemes with large transmission windows, if a broken route is found, the packets queued at the intermediate nodes will be buffered for some time. These packets may be transmitted to the destination by the intermediate nodes, which can also reduce the time spent on transmitting the packet from the source. Therefore, the results suggest that in MANETs, a TCP flow needs to inject more packets than the BDP into the networks.

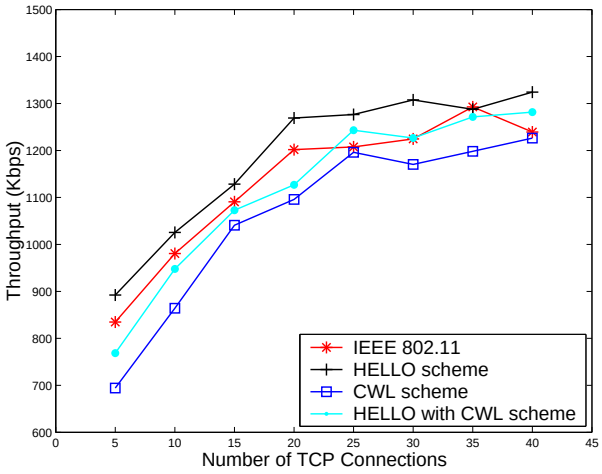
Remarks: In scenarios with 1 m/s in Fig. 3.13, the number of route breakages seems to increase to reach a peak and then decreases as the number of TCP connections increases. This is different from other scenarios with high mobility where the number of route breakages



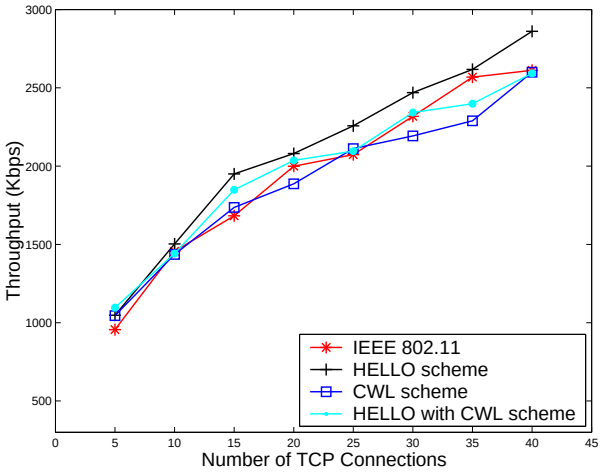
(a) 1 m/s,1000 × 1000



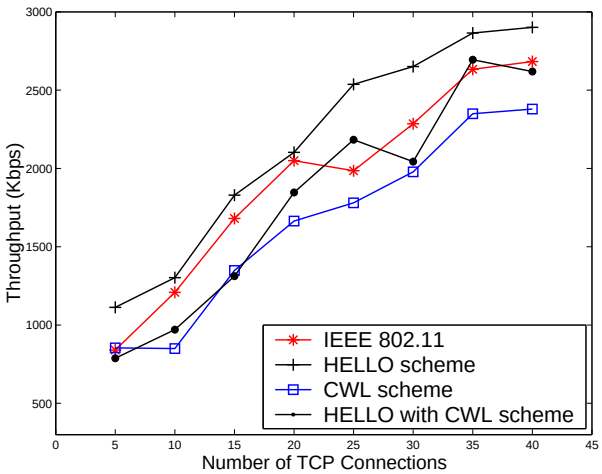
(b) 5 m/s,1000 × 1000



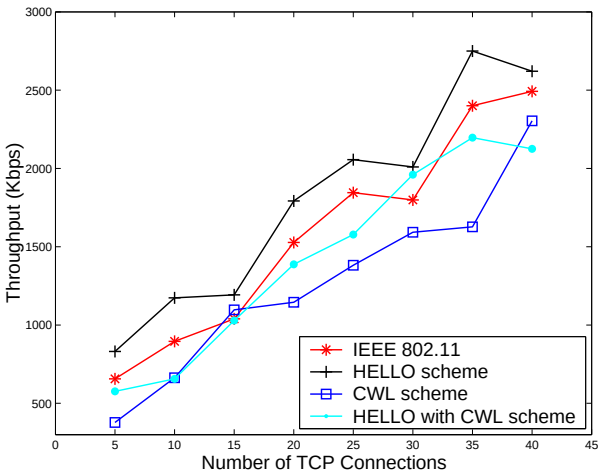
(c) 10 m/s,1000 × 1000



(d) 1 m/s,2000 × 2000



(e) 5 m/s,2000 × 2000



(f) 10 m/s,2000 × 2000

Figure 3.14: Improvement for throughput in mobile networks

increases with the increase of the number of TCP connections. In a low mobility environment, false route breakages due to medium contention dominate over mobility caused genuine route breakages. Hence, the phenomena in Fig. 3.13(a) and Fig. 3.13(d) are mainly attributed to false route breakages. It is noticed that the initial increase of the total number of route breakages is due to the increase of the number of traffic flows which leads to more medium contention. After that, the total number of route breakages reaches a peak and is approximately saturated as shown in Fig. 3.13(a), since medium contention is location dependent and will not continue to increase with saturated traffic in the network. However, Fig. 3.13(d) shows that the number of route breakages decreases from the peak when the number of TCP connections increases from 35 to 40. The reason is still unknown and worthy of further investigation. This will be explored as part of future work, since the purpose of these simulations in the thesis is to check whether the HELLO scheme can outperform other schemes.

3.6 Concluding Remarks

This chapter is an important step towards understanding the interaction between TCP protocol and IEEE 802.11 MAC protocol. We focus on the interplay between TCP and MAC protocol, and present a general methodology for calculating the upper bound of the TCP throughput over IEEE 802.11 based multi-hop string topology. Compared to the existing work [61], the analysis is unique because it quantifies the throughput of a TCP flow after considering the interaction between TCP's congestion window size and 802.11's MAC contention window size. The model considers the existence of contentions, collisions and binary exponential backoffs at the MAC layer as well as how these phenomena affect or are affected by TCP's congestion window size and packet size. Also, the difference between the transmission range and the carrier sensing range is not ignored. Simulations show that the model predicts the upper bound of the TCP throughput to a high accuracy. This work is a basis for our

further studies, and allows to isolate packet losses due to different causes and to investigate the impact of different parameters on the TCP throughput performance.

Furthermore, we investigate the misbehavior of IEEE 802.11 MAC protocol in determining a route breakage due to medium contention which seriously deteriorates the TCP performance. We have analyzed that the likelihood of false route breakages due to RTS transmission failures is proportional to the size of the packet transmitted in a network. Through simulations, we have found that a 4-hop linear chain suffers from the most serious medium contention which causes false route breakages. We propose a HELLO scheme which is a simple modification of the 802.11 MAC protocol. We achieve the goal of alleviating false route breakages by initiating a HELLO message whenever the number of RTS received by a node exceeds a certain threshold. We have shown through simulations that the proposed modification substantially reduces the false route breakages and improves throughput by up to 35%. Our results also suggest that a TCP flow should inject more packets than the BDP into the network to achieve better performance. This provides a basis for us to design a better flow control algorithm for ad hoc networks.

In Section 3.5.1, AODV protocol without Hello messages is used to validate the TCP model in the case of no packet loss. This is because AODV without HELLO messages will not introduce any routing protocol overhead during the data transmission when there is no packet loss in the network. However, DSR always introduces routing protocol overhead since every data packet in transmit contains a DSR header. Hence, with AODV, the complexity of computing the routing protocol overhead can be avoided in this section. In contrast, the rest of the thesis uses DSR instead of AODV for simulation and analytical studies. The reasons are as follows. It is noticed that DSR is used widely in the existing relevant literature papers on TCP study, which makes our study useful as a reference work. Moreover, the DSR routing protocol is easier tractable theoretically which can greatly facilitate the analysis of TCP for further study. This is reflected in Chapter 4 in the thesis.

Chapter 4

The Impact of Wireless Channel Error on TCP Performance

4.1 Introduction

This chapter studies the impact of wireless channel error on the upper bound of TCP throughput which is derived in Chapter 3. We propose a packet level model to investigate the impact of channel error on TCP performance over IEEE 802.11 based multi-hop wireless networks. A Markov renewal approach is used to analyze the behavior of TCP. We study two versions of TCP: Reno [1–3] and NewReno [64, 65]. NewReno, which has become the predominant TCP algorithm in today's Internet [66], is a simple but significant modification to the fast-recovery process of the standard Reno implementation. It helps in avoiding frequent timeouts caused by multiple packet losses within one window. We only evaluate Impatient NewReno that is recommended by the latest RFC 3782 [65]. Another variant of NewReno, called Slow-but-Steady [64], is not recommended by RFC 3782 due to its poor performance in TCP connections with large congestion windows.

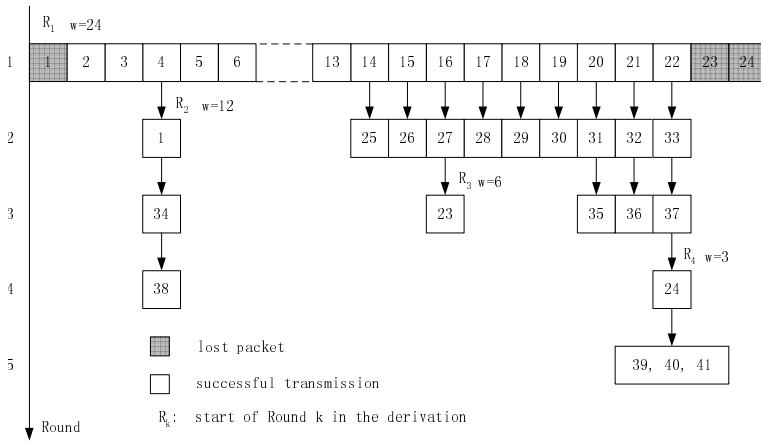
Compared to previous work, our main contributions are as follows: i) modeling of multiple lossy links; ii) investigating the interactions among TCP, IP and MAC protocol layers, specif-

ically the impact of 802.11 MAC protocol and DSR routing protocol on TCP throughput performance; iii) considering the spatial reuse property of the wireless channel, the model takes into account the different proportions between the interference range and transmission range; iv) adopting more accurate and realistic analysis to fast-recovery process, and showing the dependency of throughput and the risk of experiencing successive fast-retransmits and timeouts on the packet error probability. The results show that the impact of the channel error is reduced significantly due to the packet retransmissions on a per-hop basis and small values of BDP over ad hoc networks. The TCP throughput always deteriorates less than $\sim$10% with a packet error rate ranging from 0 to 0.1. Our analysis shows that wireless channel error is the dominant factor causing TCP data packet to be lost, which leads to false route breakages and hence seriously deteriorates TCP performance. Hence, our model provides a theoretical basis for determining the approximate optimum *long retry limit* for IEEE 802.11 to eliminate false route breakages due to wireless channel error.

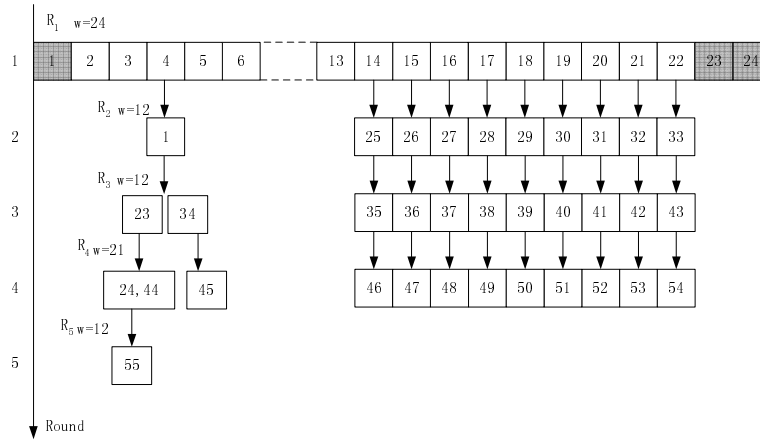
The remainder of this chapter is organized as follows. In Section 4.2, we use examples to illustrate the fast-recovery process for the two TCP variants. In Section 4.3, we describe the network model as well as various simplifying assumptions. In Section 4.4, we present the analytical TCP model to calculate the throughput for the two TCP variants without considering ACK losses. Section 4.5 discusses the approach to include the modelling of ACK losses. Section 4.6 contains the evaluation results and related discussions. Section 4.7 concludes the chapter. The Appendix contains a thorough investigation of the fast-recovery process for the two TCP variants.

4.2 Preliminaries

TCP Reno and NewReno differs in fast-recovery process. This section uses examples to illustrate fast-recovery process for the two TCP variants. To maintain continuity, the mathemat-



(a) Reno



(b) NewReno

Figure 4.1: An example of fast-recovery process for TCP Reno

ical analysis of the fast-recovery process is put in the Appendix. The analysis is independent of the type of networks and can be applied to TCP in any kind of networks including wireline and WLAN networks. We assume the reader to be familiar with the slow-start and congestion-avoidance mechanisms used in both versions of TCP and will not explain them.

We consider TCP in terms of round as done in [54], where a round starts when the sender begins the transmission of a window of packets and ends when the sender receives an acknowledgment for one or more of these packets. The several rounds of fast-recovery process for Reno and NewReno are best explained via the examples shown in Fig. 4.1. In Fig. 4.1, suppose that packet 1, 23 and 24 are lost when the window reaches $w = 24$, and that the packets 2 through 22 are successful. This window 24 is called the loss window. In this case, Reno and NewReno have the same behaviors in the second round. The source first receives 21 duplicate ACKs, each with a sequence number requesting packet 1. The first three ACKs trigger fast retransmit of packet 1, and cause the window to drop to 12. Then, this window is temporarily inflated by the number of duplicate ACKs. Once the duplicate ACK triggered by packet 13 is received, the window is inflated to $12 + 12 = 24$. The number of packets in the pipeline known by the source is still 24 since each ACK carries the sequence number requesting for packet 1. During this period, transmission of new packets is not permitted by $w \leq 24$, because the source has already transmitted packets beyond the allowed window. For every subsequent duplicate ACK, TCP continues to inflate its window and transmits one new packet, ending up transmitting 9 new packets, with the largest sequence number $24 + 9 = 33$.

A. Reno

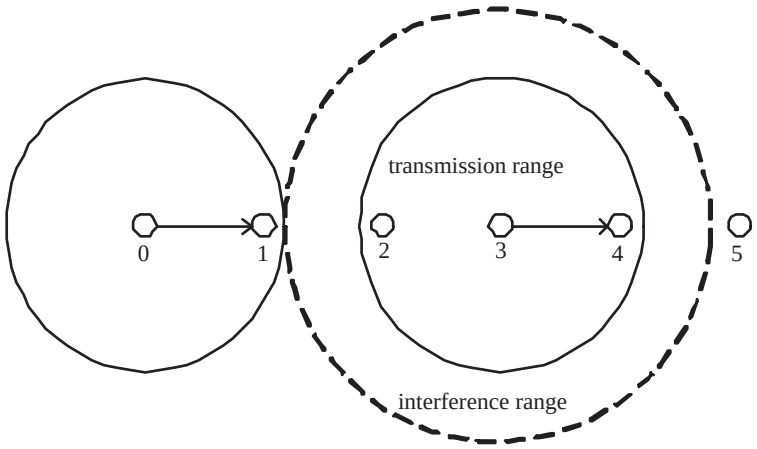
In Reno, as shown in Fig. 4.1(a), this inflation is removed and the window is cut back to 12 when the sequence number carried by ACK advances, i.e., when the ACK for the retransmission comes back with a sequence number requesting packet 23. At that point, the current window size of 12 allows Reno TCP to transmit only one new packet 34, since the outstanding

packet is $33 - 22 = 11$. In the third round, the lost packet 23 is retransmitted with the arrival of four duplicate ACKs requesting packet 23. The window is further decreased in half and followed by 4 new packet transmissions due to the window inflation. Following this logic, in the fourth round, the window drops to 3 and no new packets can be sent. The fifth round starts with the exit of fast-recovery process when packet 24 is successfully retransmitted.

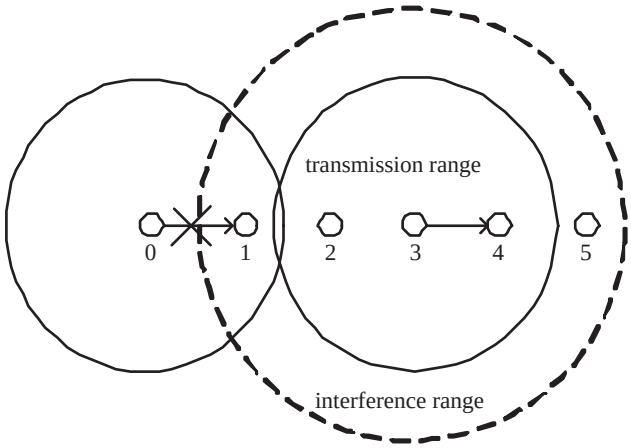
B. NewReno

In NewReno, as shown in Fig. 4.1(b), with the successful retransmission of packet 1, one partial acknowledgment requesting packet 23 arrives at the TCP source. Packet 23 is immediately retransmitted without waiting for enough duplicate ACKs. The congestion window is deflated by the amount of new data acknowledged minus one segment and is $33 - 22 + 1 = 12$. One more packet 34 is allowed to be transmitted since the outstanding packet is 11. For each additional duplicate ACK received, the congestion window is incremented which allows more new packets to be sent as shown in the figure. Following this logic, the congestion window is artificially inflated to $12 + 9 = 21$ when ACK requesting packet 24 arrives. It is not deflated since only one new packet is acknowledged. With the successful retransmission of packet 24, the window is cut back to 12 again.

The two variants of NewReno differ in their attempts to reset the retransmit timer. The Slow-but-Steady variant resets the retransmit timer after each partial acknowledgment. The Impatient variant resets the retransmit timer only upon the receipt of first partial ACK. In this case, when a large amount of packets are dropped, the source retransmit timer will ultimately expire, which invokes slow-start phase to probe the available capacity of the network from the beginning.



(a) Relatively large distance between two adjacent nodes



(b) Relatively small distance between two adjacent nodes

Figure 4.2: Two different linear chains with $R_{tx} < R_{in} < 2R_{tx}$

4.3 System Model

We consider a static, multi-hop string topology with n hops (node 0 through node n). A single persistent TCP connection is set up between the source node 0 and the sink node n. Node 0 is an infinite data source that always has packets to send. At each node, the FIFO buffer size is infinite. The distance between any two adjacent nodes is identical.

While considering the spatial reuse nature of the wireless channel, in previous literature [5, 7–10, 45], the interference range of a node is usually chosen to be twice and slightly greater than the transmission range so that each node: (i) can only transmit to its one-hop neighbors; (ii) will undoubtedly interfere with its two-hop neighbors; and (iii) cannot sense all other nodes. However, in practice, the interference range R_{in} and the transmission range R_{tx} are determined by various factors such as antenna sensitivity, antenna direction, transmission power, etc. Their relationship also varies with different network scenarios and may not always satisfy the commonly-used assumption that $2R_{tx} \approx R_{in}$ and $2R_{tx} < R_{in}$. In our model, we take into account different proportions between R_{in} and R_{tx}, and for any transmission ranges, the distance satisfies the condition where transmissions from one node can only reach its one-hop neighbors. Fig. 4.2 depicts two different scenarios for an 802.11 based linear chain network with six ($n = 5$) nodes where $R_{tx} < R_{in} < 2R_{tx}$. In Fig. 4.2(a), node 1 is outside the interference range of node 3, hence the transmission between node 0 and node 1 is successful. On the contrary, in Fig. 4.2(b), node 0 fails to transmit to node 1. Node 1 cannot send CTS to node 0 since it is within the interference range of node 3 and it can hear the ongoing transmission of node 3. The second scenario as depicted in Fig. 4.2(b) corresponds to the commonly used network model where $2R_{tx} \approx R_{in}$ and $2R_{tx} < R_{in}$.

Since our objective is to investigate the impact of channel error on TCP performance, we ignore the packet losses caused by MAC layer contention by setting the RTS *short retry limit* to infinite, as opposed to the default value of seven in IEEE 802.11. Hence, our model does not experience any packet losses invoked by buffer overflow, mobility and MAC layer

contention. This model allows the isolation of packet losses due to different causes and aids the investigation of the impact of different loss parameters on the TCP throughput performance. Meanwhile, we assume that TCP data packets may be lost, but TCP ACKs are never lost due to their small sizes.

As in [51, 52, 58], we adopt an i.i.d packet error model. In previous work, a burst packet loss model is also frequently adopted where several packets can be dropped when the channel experiences bad link quality for a long duration. However, in our model which uses IEEE 802.11, it is not realistic to model burst packet error because a TCP packet transmission can occur only when the MAC layer RTS-CTS frame exchange is successful. The bad channel condition can only corrupt several consecutive transmissions of a RTS packet. In addition, a corrupted packet may not always be discarded. A TCP data packet is discarded only when its number of retransmission attempts reaches the *long retry limit* on one link. Let p denote the probability that a packet is lost and $\bar{p} = 1 - p$. Let q denote the error probability of a packet on one wireless link and b denote the *long retry limit*. Since reliable link forwarding fails only when all b transmissions fail, the probability of a link packet transmission failure is given by q^b. Hence, the probability of unsuccessful end-to-end delivery of a packet is given by $p = 1 - (1 - q^b)^n$.

4.4 Throughput Calculation without ACK Losses

It is well known that TCP shows a dynamic cyclic evolution of its congestion window in the event of packet losses and that the cycles form a renewal process. As shown in Fig. 4.3, one cycle in our model is defined as the interval between the end of one fast-recovery process and the end of the next fast-recovery process. The cycle duration is a random variable which is further divided into two non-overlapping components at the time when the first packet loss occurs. Let Y and Z denote the duration before and after the first packet loss within a cycle

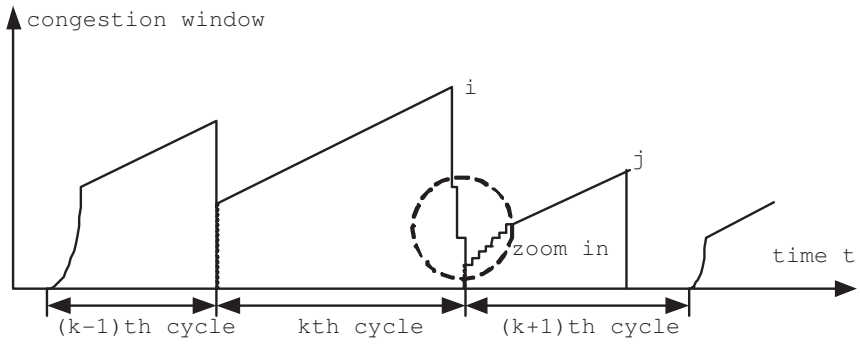


Figure 4.3: Cyclic evolution of TCP congestion window

respectively. The cycle duration is given by $Y+Z$. Let A and B denote the number of packets transmitted within Y and Z respectively. Hence, the total packets transmitted within a cycle is $A+B$. As such, we define the long-term steady-state TCP throughput for a n-hop linear chain, μ as follows:

$$\mu = \frac{E(A+B)}{E(Y+Z)} = \frac{E(A)+E(B)}{E(Y)+E(Z)}. \tag{4.1}$$

In the rest of this section, we will derive the expressions for the mean of A, B, Y and Z for Reno and Impatient NewReno.

A. Analyzing the Renewal Process of TCP

This section presents a quantitative approach to model the renewal process of TCP behavior, which provides a basis for further derivation of the parameters in (4.1). Let i be the congestion window size at which the first packet loss occurs in the k-th cycle. This window i is called the loss window. We model the process $\{i\}$ as a single dimensional semi-Markov chain over the state space $\{1, 2, \ldots, W_{max}\}$, where W_{max} is the maximum congestion window size. For the Markov chain, the transition probability from state i to state j, which we denote as P_{ij}, depends on the TCP behavior. Specifically, the transition probability must be calculated depending on whether the TCP fast-recovery process ends without or with a timeout. Let $\bar{\alpha}_{ij}$ and α_{ij} be the transition probabilities from state i to state j given that the fast-recovery

process in state i ends without and with a timeout respectively. Then $P_{ij} = \bar{\alpha}_{ij} + \alpha_{ij}$.

To determine $\bar{\alpha}_{ij}$ and α_{ij}, let l denote the number of detected packet losses via the receipt of three duplicate ACKs or partial ACKs during the fast-recovery process. The value of l is a random variable and is dependent on the packet loss scenario and the fast-recovery mechanism. We have $\bar{\alpha}_{ij} = \sum_{l=0}^{i} \bar{\alpha}_{ij}(l)$ and $\alpha_{ij} = \sum_{l=0}^{i} \alpha_{ij}(l)$, where $\bar{\alpha}_{ij}(l)$ and $\alpha_{ij}(l)$ are the values of $\bar{\alpha}_{ij}$ and α_{ij} given l. The conditional transition probabilities $\bar{\alpha}_{ij}(l)$ and $\alpha_{ij}(l)$ are given as follows:

$$\bar{\alpha}_{ij}(l) = \begin{cases} 0 & 1 \le j < x \\ \bar{\gamma}_{il}(1-\bar{p}^j)\bar{p}^{\bar{a}(x,j-1)} & x \le j < W_{max} \\ \bar{\gamma}_{il}\bar{p}^{\bar{a}(x,j-1)} & j = W_{max}, \end{cases} \tag{4.2}$$

$$\alpha_{ij}(l) = \begin{cases} \gamma_{il}p & j = 1 \\ \gamma_{il}(1-\bar{p}^j)[\bar{P}_\tau \bar{p}^{a(x,j-1)} + P_\tau \bar{p}^{a(2,j-1)}] & 2 \le j < W_{max} \\ \gamma_{il}[\bar{P}_\tau \bar{p}^{a(x,j-1)} + P_\tau \bar{p}^{a(2,j-1)}] & j = W_{max}, \end{cases} \tag{4.3}$$

where $\bar{\gamma}_{il}$ and γ_{il} are the stationary probabilities of having only l detected packet losses without and with a timeout respectively, and that the first packet loss occurs when the congestion window size is i. The detailed derivation of $\bar{\gamma}_{il}$ and γ_{il} is presented in the Appendix. $\bar{a}(x_1, x_2)$ is the number of packets transmitted during the congestion avoidance period within the cycle when the congestion window starts from x_1 and ends with x_2; $a(x_1, x_2)$ is the number of packets transmitted after the fast-recovery process when the congestion window starts from 1 with slow-start and ends with x_2 given the slow-start threshold x_1. After the fast-recovery process, congestion window increases continuously until a packet loss occurs at congestion window $x_2 + 1$. Hence, $\bar{a}(x_1, x_2) = x_1 + (x_1 + 1) + (x_1 + 2) + \cdots + x_2$ and $a(x_1, x_2) = 1 + 2 + 2^2 + 2^3 + \cdots + x_1 + (x_1 + 1) + (x_1 + 2) + \cdots + x_2$.

In (4.2), the fast-recovery process ends without a timeout, which causes TCP to enter a congestion avoidance phase with congestion window x. During the congestion avoidance

phase, the congestion window increases linearly from x until it reaches j. Therefore, there are no circumstances in which window j is less than x. In the case of $x \leq j < W_{max}$, packet losses happen within the loss window j. If j is the maximum congestion window size W_{max}, the necessary condition for TCP to reach W_{max} is that there must be no packet loss during the period when the congestion window increases from x to $W_{max} - 1$. Packet losses may happen after several W_{max} number of packets have been sent.

Similarly, we obtain (4.3), where P_τ is the probability that a retransmitted packet is lost. This leads to consecutive timeouts taking place within a cycle. In this case, the window threshold drops to the minimum value of 2. Otherwise, the window threshold x is dependent on the value of l. Therefore, $P_\tau = \sum_{i=1}^{+\infty} p^i(1-p) = p$ and $\bar{P}_\tau = 1 - P_\tau$.

For TCP Reno and NewReno, we calculate x in (4.2) and (4.3) as shown below.

Reno: In TCP Reno, we consider the fact that there can be at most three packet losses that can be detected by the receipt of three duplicate ACKs during the fast-recovery process (see Appendix). In (4.2), let $l = \{1, 2, 3\}$ be the number of detected packet losses via Three-Duplicate-ACKs events. The congestion window size is halved at each detected packet loss; therefore, after the fast-recovery process, both the congestion window and the slow-start threshold of the congestion window are given by $x = \lfloor i/2^l \rfloor$.

For the case in (4.3) that the fast-recovery ends with a timeout, in contrast with the case of no timeout, $l = \{0, 1, 2, 3\}$ here can be zero. Specifically, $l = 0$ implies that the fast-recovery process enters the timeout directly without detecting any packet losses through the receipt of three duplicate ACKs. For instance, if there is one packet loss given that the current loss window is 2, the source cannot receive 3 duplicate ACKs and hence enters the timeout process. When a timeout event occurs, the congestion window of the current cycle will always end with 1. However, the slow-start threshold after the fast-recovery process is dependent on l. It is always halved as many as l times during the occurrence of Three-Duplicate-ACK

events and then halved again during the occurrence of a timeout event; hence, the slow-start threshold is $x = \lfloor i/2^{l+1} \rfloor$. However, in the presence of consecutive timeouts, the slow-start threshold is at its minimum value of 2.

NewReno: Contrary to Reno where the congestion window can be halved several times during one fast-recovery process, the congestion window of NewReno can only be halved once. Hence, $x = \lfloor i/2 \rfloor$ for both $\bar{\alpha}_{ij}(l)$ and $\alpha_{ij}(l)$.

Based on the above analysis, we can get the $W_{max} \times W_{max}$ transition probability matrix of $\{P_{ij}\}$. The stationary probability π_i can be obtained by solving the Markov chain numerically given the transition probability matrix.

B. Finding $E(A)$

Let A_j denote the number of packets transmitted before the first lost packet in state j during the cycle. Then, $E(A)$ in (1) is given as follows:

$$E(A) = \sum_{j=1}^{W_{max}} \pi_j A_j. \tag{4.4}$$

To obtain A_j, recall that a fast-recovery process can end with three scenarios: no timeout, one timeout and consecutive timeouts. For the two kinds of timeouts, slow-start phase in next cycle starts with different slow-start threshold. Hence, considering these three scenarios, A_j is determined as follows:

$$A_j = \sum_{i=1}^{W_{max}} \sum_{l=0}^{i} [\bar{\alpha}_{ij}(l)\bar{f}(x^*, j) + \alpha_{ij,1}(l) f(x^{**}, j) + \alpha_{ij,2}(l) f(2, j))], \tag{4.5}$$

where $\alpha_{ij,1}(l)$ and $\alpha_{ij,2}(l)$ are the probabilities of non-consecutive and consecutive timeouts respectively during the fast-recovery process such that $\alpha_{ij}(l) = \alpha_{ij,1}(l) + \alpha_{ij,2}(l)$. Then, according to (4.3), we have:

$$\alpha_{ij,1}(l) = \begin{cases} \gamma_{il}\bar{P}_\tau p & j = 1 \\ \gamma_{il}(1-\bar{p}^j)\bar{P}_\tau \bar{p}^{a(\lfloor i/2^{l+1} \rfloor, j-1)} & 2 \le j < W_{max} \\ \gamma_{il}\bar{P}_\tau \bar{p}^{a(\lfloor i/2^{l+1} \rfloor, j-1)} & j = W_{max}. \end{cases} \tag{4.6}$$

In (4.5), $\bar{f}(x_1, x_2) = \bar{a}(x_1, x_2 - 1) + \lfloor x_2/2 \rfloor$, and $f(x_1, x_2) = a(x_1, x_2 - 1) + \lfloor x_2/2 \rfloor$, where, in both functions, the second term, i.e., $\lfloor x_2/2 \rfloor$ represents the average number of packets transmitted within the loss window x_2 before the first packet loss occurs. Here, we assume this number of packets to be uniformly distributed between 1 and x_2. The values of x^* and x^{**} in (4.5) vary with different flavors of TCP. For Reno, $x^* = \lfloor i/2^l \rfloor$ and $x^{**} = \lfloor i/2^{l+1} \rfloor$. For NewReno, $x^* = x^{**} = \lfloor i/2 \rfloor$.

C. Finding $E(B)$

Since $E(B)$ is about the average number of packets transmitted during the fast-recovery process of a cycle and the process can end with or without a timeout, we further divide $E(B)$ into two components such that $E(B) = +E(\bar{C}) + E(C)$, where $E(\bar{C})$ and $E(C)$ are the average number of packets transmitted when the fast-recovery process ends without and with a timeout respectively.

Let $\bar{\Phi}_k$ and Φ_k denote the number of packets successfully transmitted in the k-th round of the fast-recovery phase for the cases without and with a timeout respectively. The derivation of $\bar{\Phi}_k$ and Φ_k for TCP Reno and NewReno is included in the Appendix. The mean number of packets transmitted during fast-recovery process in the two cases are given by:

$$E(\bar{C}) = \sum_{i=1}^{W_{max}} \sum_{l=0}^{i} \sum_{k=1}^{l+1} \pi_i \bar{\gamma}_{il} \bar{\Phi}_k, \qquad \text{and}$$

$$E(C) = \sum_{i=1}^{W_{max}} \sum_{l=0}^{i} \sum_{k=1}^{l+1} \pi_i \gamma_{il} \Phi_k. \tag{4.7}$$

Note that $\bar{\Phi}_k = \Phi_k = 0$ for $k > l + 1$, which indicates that there is no successful data transmission within the duration of consecutive timeouts.

D. Finding $E(Y)$

To calculate $E(Y)$, corresponding to (4.4) and (4.5), let Y_j denote the time taken to transmit A_j number of packets. Then, the mean transmission time $E(Y)$ in each cycle, before the first

packet loss, is given by:

$$E(Y) = \sum_{j=1}^{W_{max}} \pi_j Y_j, \tag{4.8}$$

where Y_j is determined as follows:

$$Y_j = \sum_{i=1}^{W_{max}} \sum_{l=0}^{i} [\bar{\alpha}_{ij}(l)\bar{g}(x^*, j) + \alpha_{ij,1}(l)g(x^{**}, j) + \alpha_{ij,2}(l)g(2, j))]. \tag{4.9}$$

In the equation above, $\bar{g}(x_1, x_2)$ and $g(x_1, x_2)$ are the times taken to transmit $\bar{f}(x_1, x_2)$ and $f(x_1, x_2)$ number of packets respectively.

Let $\widehat{\mu}(n, W)$ denote the TCP throughput of an n-hop ad hoc network without any packet losses given a congestion window size W. $\widehat{\mu}(n, W)$ is the upper bound performance of the TCP throughput which can be obtained by either the simulation or the calculation in Chapter 3. Therefore, for an n-hop network, the times taken to transmit $\bar{f}(x_1, x_2)$ and $f(x_1, x_2)$ number of packets without considering packet error are then given by:

$$\widehat{\bar{g}}(x_1, x_2) = \sum_{i=x_1}^{x_2-1} \frac{i}{\widehat{\mu}(n, i)} + \frac{\lfloor x_2/2 \rfloor}{\widehat{\mu}(n, x_2)}, \quad \text{and}$$

$$\widehat{g}(x_1, x_2) = \sum_{i=0}^{\lfloor \log_2 x_1 \rfloor} \frac{2^i}{\widehat{\mu}(n, 2^i)} + \sum_{i=x_1}^{x_2-1} \frac{i}{\widehat{\mu}(n, i)} + \frac{\lfloor x_2/2 \rfloor}{\widehat{\mu}(n, x_2)}. \tag{4.10}$$

However, the transmitted packets may be corrupted due to channel error in ad hoc networks. The corrupted packets will not be discarded until the number of retransmissions on one link reaches *long retry limit b*. The retransmissions on the links contribute to the total time taken to transmit $E(A)$ packets. Hence, considering these retransmissions, the transmission times are rewritten as follows:

$$\bar{g}(x_1, x_2) = \widehat{\bar{g}}(x_1, x_2) + \kappa \bar{f}(x_1, x_2) \sum_{i=1}^{b-1} i q^i (1-q) T_D, \quad \text{and}$$

$$g(x_1, x_2) = \widehat{g}(x_1, x_2) + \kappa f(x_1, x_2) \sum_{i=1}^{b-1} i q^i (1-q) T_D, \tag{4.11}$$

where T_D is the time duration from the instant a node sends a RTS packet successfully to the instant the node receives the expected link-layer acknowledgement for the TCP data packet on one link; κ is a coefficient associated with the number of links on which one packet is retransmitted on a link basis. Due to spatial reuse in the topology, the time spent by an individual node in retransmissions may overlap. Hence, κ may not be the actual path length and may vary with different network topologies. In the case of $R_{tx} < R_{in} < 2R_{tx}$, for two different network scenarios Fig. 4.2(a) and Fig. 4.2(b), κ is given by $min(3, n)$ and $min(4, n)$ respectively, as there is almost always one packet transmission every 3 or 4 hops in the chain of nodes for the two scenarios. As a result, we can get κ for other network scenarios by considering different proportions between R_{in} and R_{tx} as $\kappa = min(4 + i, n)$, where $(2 + i)R_{tx} \leq R_{in} < (3 + i)R_{tx}$. With κ determined as above and using the same π_j in (4.4), $E(Y)$ in (1) can now be calculated using (4.8).

E. Finding $E(Z)$

Let $E(\bar{D})$ and $E(D)$ be the time required to transmit $E(\bar{C})$ and $E(C)$ respectively, as determined previously in (4.7). To calculate $E(\bar{D})$ and $E(D)$, similar to the calculation of $E(Y)$, the expected time duration taken to recover the lost packets without considering packet error is the ratio of the corresponding number of packets and transmission rate:

$$E(\widehat{\bar{D}}) = \sum_{i=1}^{W_{max}} \sum_{l=0}^{i} \sum_{k=1}^{l+1} \pi_i \bar{\gamma}_{il} \frac{\bar{\Phi}_k}{\widehat{\mu}(n, \bar{\Phi}_k)}, \quad \text{and}$$

$$E(\widehat{D}) = \sum_{i=1}^{W_{max}} \sum_{l=0}^{i} \sum_{k=1}^{l+1} \pi_i \gamma_{il} \frac{\Phi_k}{\widehat{\mu}(n, \phi_k)}. \tag{4.12}$$

Furthermore, we take into account the impact of packet error. A TCP data packet is discarded when its number of retransmission attempts reaches the *long retry limit* on one link. The node that drops the packet interprets this as a route failure and sends route error messages to the sender. Consequently, the time spent on data retransmissions on a link basis

as well as the time spent on the route re-establishment on a path basis contribute to the degradation of TCP throughput performance. Hence, the total time taken for the case of no timeout is given by:

$$E(\bar{D}) = E(\widehat{\bar{D}}) + \kappa E(\bar{C}) \sum_{i=1}^{b-1} iq^i(1-q)T_D + \sum_{i=1}^{W_{max}} \sum_{l=0}^{i} \pi_i \bar{\gamma}_{il} T_r(l), \qquad \text{and}$$

$$T_r(l) = lT_{err} + T_{disc} + T_{rep} + l(\lceil \frac{n}{2} \rceil + 1)T_D + \kappa l \sum_{j=1}^{b-1} jq^j(1-q)T_D, \tag{4.13}$$

where $T_r(l)$ includes the time spent on route error packet transmissions (T_{err}) invoked by l lost packets, route discovery (T_{disc}), rout reply packet transmissions (T_{rep}), retransmissions of the lost packets along the path and along the links. We assume l lost packets invoke only l route error messages to be transmitted by the intermediate nodes, and only one route discovery message and one route reply message are sent by the source and the destination respectively. The packet loss may occur at any one of links along the path. We assume that the packet loss is uniformly distributed on each link; the average impact of packet loss can then be viewed at the middle link on the path.

Similarly, the total time for transmissions in the case of timeout is given by:

$$E(D) = E(\widehat{D}) + \kappa E(C) \sum_{i=1}^{b-1} iq^i(1-q)T_D + E(T_t), \tag{4.14}$$

where $E(T_t)$ is the average timeout interval. The timeout interval is independent of the route re-establishment process. The length of a timeout interval is determined by the probability that the first packet sent after a timeout period is lost again. For each immediate consecutive timeout, the waiting time is doubled up to a maximum of $2^6 T_o$, where T_o is the unit of RTO (retransmission timeout). Following the calculation in [54], the timeout interval T_t is:

$$T_t = T_o \frac{1 + \sum_{i=1}^{6} 2^{i-1} p^i}{1-p}. \tag{4.15}$$

In our model, we obtain an approximation of the unit of RTO T_o as follows. After a packet loss, there may still be some packets in the buffer of the source node. The source node starts

a retransmission timer after sending the last packet in the current window, i.e., the packets in its buffer. Ideally, the exact timeout should be set for the epoch at which the ACK for this last transmitted packet is expected. We approximate this exact timeout as the time to transmit all the packets of the current loss window except the lost one. A coarse timeout can then be obtained by adding half of the timer granularity (T_{og}) to this exact timeout epoch. The actual timeout T_o is taken as the maximum value between the coarse timeout and the minimum timeout (T_{omin}):

$$T_o(j,n) = \max(\frac{j-1}{\widehat{\mu}(n,j)} + \frac{T_{og}}{2}, T_{omin}), \tag{4.16}$$

where j is congestion window and n is the path length. Then, the average timeout interval $E(T_t)$ is approximately given by:

$$E(T_t) = \sum_{j=1}^{W_{max}} \sum_{l=0}^{j} \pi_j \gamma_{jl} T_o(j,n) \frac{1 + \sum_{i=1}^{6} 2^{i-1} p^i}{1-p}. \tag{4.17}$$

4.5 Discussion of ACK Losses

For ease of derivation, we assume that ACK packets are not lost due to their small size in our model. The assumption has been widely used in previous work [51–60]. In principle, it is possible to exactly compute the throughput based on ACK loss analysis, and we sketch this method in the following, which we will pursue in more detail in future.

Let p_d and p_a denote the packet loss probabilities of the forward path and the reverse path in the network respectively. Then, considering the condition that a given TCP data packet reaches the receiver successfully, the probability that i data packets and j ACK packets within window w reach their destinations is given by:

$$h(i,j) = \bar{p_d} \times \binom{w-1}{i-1} p_d^{w-i} \bar{p_d}^{i-1} \times \binom{i}{j} p_a^{i-j} \bar{p_a}^{j} \tag{4.18}$$

where $\bar{p_d} = 1 - p_d$, $\bar{p_a} = 1 - p_a$. In (4.18), the second term means that $(w-i)$ TCP data packets are lost during transmission; hence i ACK packets can be triggered with the receipt of

the same number of TCP data packets. The third term indicates that within i ACK packets, j ACK packets are transmitted to the source successfully.

Since TCP employs a cumulative ACK technique, if an ACK is dropped, a cumulative ACK that is transmitted later will inform the source that the data has actually been received. In $h(i, j)$, as long as $j > 0$, the TCP data packet is acknowledged and new data packets can be transmitted. Otherwise, a timeout event occurs. Hence, given any data packet, TCP may experience three possible scenarios: i) The data packet is dropped with probability p_d, and TCP enters fast-recovery process. ii) The data packet arrives at the receiver successfully with probability $\bar{p_d}$, but TCP enters timeout and retransmits the data packet due to the loss of all ACKs sent by the receiver. iii) The data packet is successfully transmitted and acknowledged, which allows TCP to generate and transmit a new data packet.

Our model in this chapter only analyzes scenarios i) and iii) by ignoring ACK loss. The ACK loss introduces a new condition for timeout events, which leads to timeout and fast-retransmit probabilities which follow different distributions and hence affecting the TCP throughput performance. If the three scenarios are modelled, TCP throughput can be calculated using the similar approach in this chapter. The TCP throughput degrades due to the increase of the timeout probability. However, it should be noted that as long as a data packet can be acknowledged, the greater the number of ACK packets lost, the more effective the bandwidth can be used. This behavior is similar to the technique of delayed acknowledgements, which leads to the increase in TCP throughput.

4.6 Simulation and Validation

We use the GloMoSim to validate the analytical model. The transmission range and interference range of the wireless radio are 367m and 670m respectively with channel bandwidth as 2Mbps. All nodes communicate with identical, half-duplex wireless radio. In order to obtain

accurate results, the straight linear chain topology is perturbed and the nodes are no longer in a straight line. The simulated network consists of 45 nodes randomly placed in an area of $4000m \times 4000m$. A single TCP connection is sequentially set up between node *0* and one of the rest nodes from node *1* to node *44*. Each TCP connection is run for 600s and 50 simulations are run. The result of each n-hop source-destination pair is collected only if the scenario satisfies the condition of Fig. 4.2(a) or Fig. 4.2(b). The TCP data packet size is set to 1460 bytes. The values of T_{og} and T_{omin} are 500ms and 1s respectively. To facilitate the mathematical calculation, DSR protocol is used. Since the transmission of routing overheard has priority over other packets, we can easily calculate T_{err}, T_{disc} and T_{rep} without considering the time spent on MAC layer contention among various nodes. They are given as follows:

$$T_{err} = 16\lceil\frac{n}{2}\rceil^2 + 1312\lceil\frac{n}{2}\rceil \quad \mu s \tag{4.19}$$

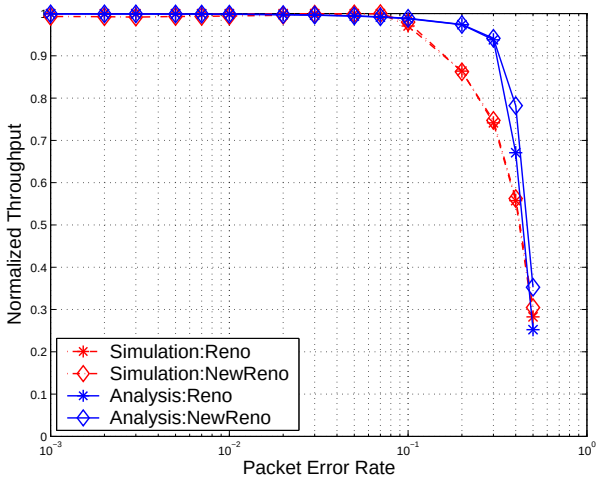
$$T_{disc} = \frac{164}{3}\lceil\frac{n}{2}\rceil^3 + 272\lceil\frac{n}{2}\rceil^2 + \frac{4996}{3}\lceil\frac{n}{2}\rceil + 600 \quad \mu s \tag{4.20}$$

$$T_{rep} = 2\lceil\frac{n}{2}\rceil^2 + 986\lceil\frac{n}{2}\rceil + 136 \quad \mu s \tag{4.21}$$

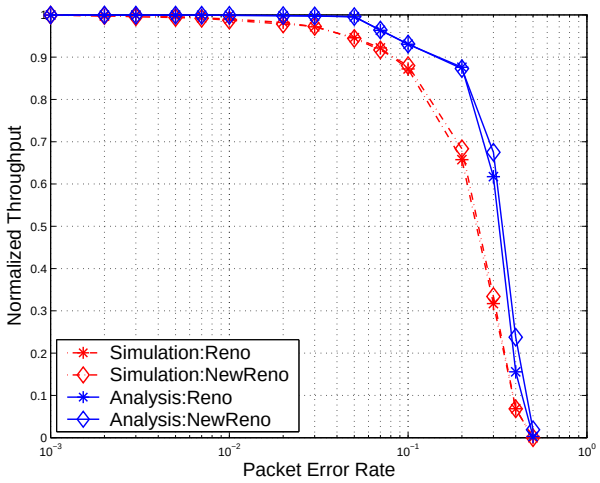
We have obtained the TCP throughput for different number of hops ranging from 1 to 10 hops combined with different W_{max} values which vary among 8, 16, 24 and 32. In this section, we present results for the scenario Fig. 4.2(b) only, as results obtained for the scenario Fig. 4.2(a) are similar and the scenario Fig. 4.2(b) is used widely in ad hoc networks research nowadays. Also, we only present the results for $n = 1$, 4, and 8 with $W_{max} = 32$ due to their similarities.

4.6.1 Throughput Validation

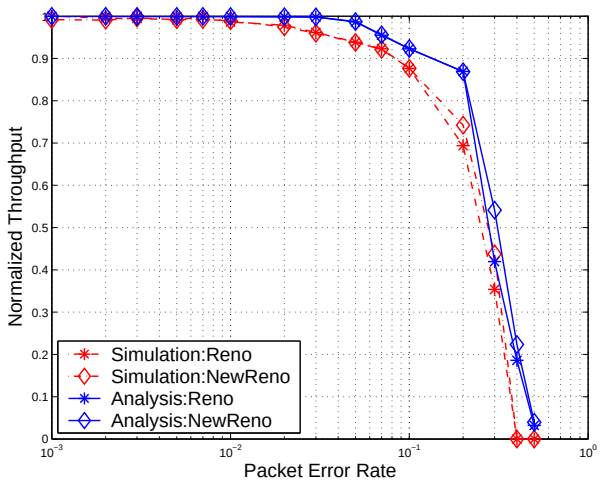
Fig. 4.4 compares the simulation results to the theoretical results. In the figures, throughput μ is normalized to $\widehat{\mu}(n, W_{max})$ which is measured under the situation where there is no packet loss in the static chain. The normalized throughput is plotted against packet error probability



(a) $n = 1$



(b) $n = 4$



(c) $n = 8$

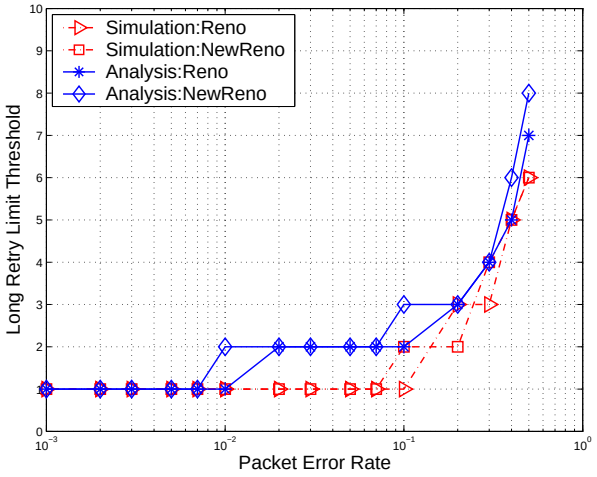
Figure 4.4: Throughput validation; $W_{max} = 32$

q. In all subfigures, it can be seen that the throughput for both TCP variants degrades with the increasing packet error probability q. The results from the analysis of the proposed models match closely with the simulations for packet error rates ranging from 0 to ∼0.1. Within this range, the packet error rates do not have significant impact on TCP performance in multi-hop ad hoc networks, where the TCP throughput do not deteriorate more than ∼10%. With higher packet error rates exceeding 0.1, the simulation results and analytical results show similar trends and there is a precipitous drop in the TCP throughput. It can be seen that Reno and Impatient NewReno have the same throughput with light and moderate packet error rates, and Impatient NewReno outperforms Reno as packet error rates increase.

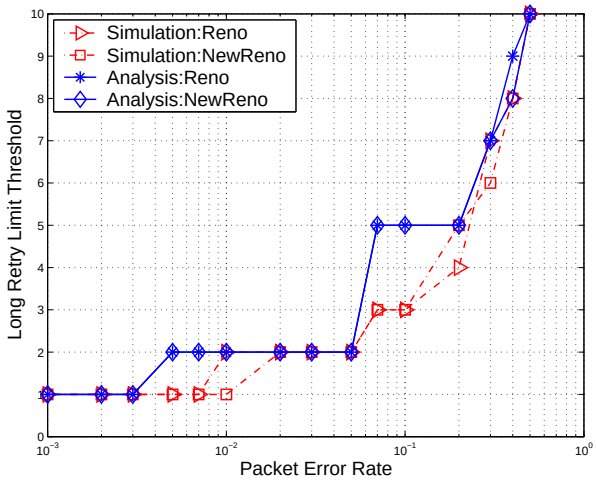
Our results show that TCP over ad hoc networks has much better performance than that in one-hop networks. In [53], there is a precipitous drop of TCP throughput for loss probabilities exceeding 0.01. However, in ad hoc networks, the throughput degradation remains at less than 10% for packet error probabilities ranging from 0 to ∼0.1, especially for one-hop ad hoc networks. This is attributed to the link-level retransmissions of error packets in IEEE 802.11, which can largely reduce the impairment of channel error. In addition, the network pipeline in a wired network or WLAN with a large BDP cannot be fully utilized due to the abrupt reduction of the transmission window caused by packet losses. Multiple round-trip times are required to increase the transmission window and to fill the pipeline again. On the contrary, ad hoc networks with small BDP values are not sensitive to variances in the congestion window.

4.6.2 Study of Long Retry Limit

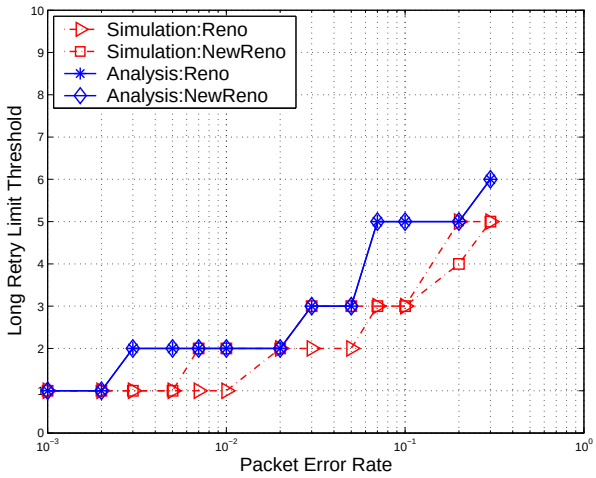
In this section, we investigate the effect of link-level retransmissions of data packets on the two TCP variants. We obtain the analytical and simulation throughput for different packet error rate q and *long retry limit* b ranging from 1 to 15. Our results show that for both TCP variants, with a given packet error rate, the throughput firstly increases with the increase of



(a) $n = 1$



(b) $n = 4$



(c) $n = 8$

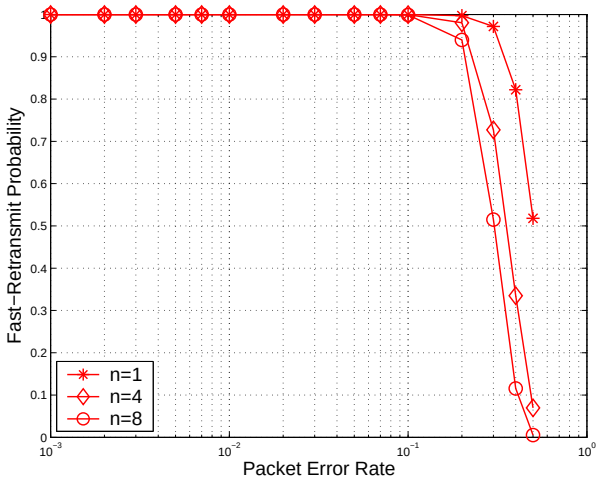
Figure 4.5: The study of long retry limit; $W_{max} = 32$

the *long retry limit b*. When b reaches a threshold, the throughput remains steady regardless of b. The values of threshold for different scenarios are obtained as shown in Fig. 4.5. It can be seen that the threshold increases with packet error rate. The analytical threshold provides an approximate upper bound for the simulation result. In addition, not all the scenarios obtain the best TCP performance with the default value of $b = 4$.

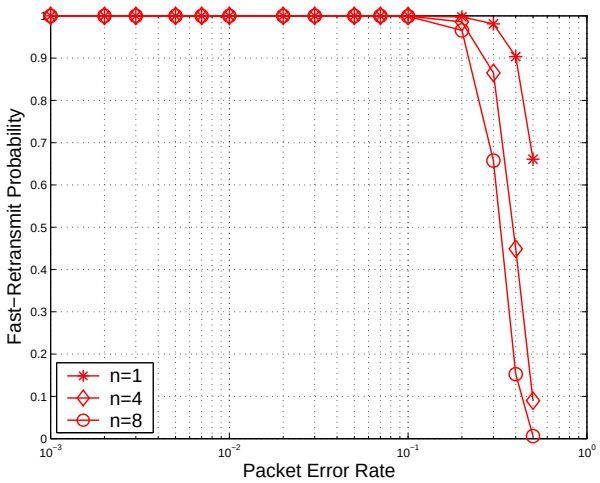
When considering the design of a suitable *long retry limit*, a small value of b is in favor of quick detection of a genuine route breakage, while a large value of b can help to prevent false route breakages as illustrated in our model. However, it should be noted that, in IEEE 802.11, a genuine route breakage is usually identified by a failed RTS transmission instead of a failed TCP data packet transmission, as a TCP data packet is sent only if a RTS packet is transmitted successfully. Therefore, the channel error can be regarded as the dominant factor in the design of a suitable b value. With this basis, the value of b should not be less than the threshold given in Fig. 4.5, and the best suitable value of b is the threshold value. In the figure, it can be observed that the threshold becomes greater than 4 in the case of $q > 0.1$, which indicates that TCP works well only for packet error rates ranging from 0 to 0.1 with the default $b = 4$. Since the threshold of our proposed model provides an approximate upper bound for the simulation result, our model provides us with a theoretical basis for determining the approximate optimum *long retry limit*.

4.6.3 Fast-Retransmit Probability

This section investigates the dependency of TCP throughput on timeout probabilities. It is known that ad hoc networks have very small BDP values of only a few packets. If the packet loss rate is light, the TCP congestion window is usually greater than the BDP of the path even though the congestion window is decreased during fast-retransmit events. With the small BDP, the network pipeline is usually fully utilized. The throughput degradation due to light packet loss rates is negligible. The TCP performance greatly depends on the timeout



(a) Reno



(b) Impatient NewReno

Figure 4.6: Fast-Retransmit probability for $n = 1, 4, 8$; $W_{max} = 32$

events that can significantly deteriorate the throughput.

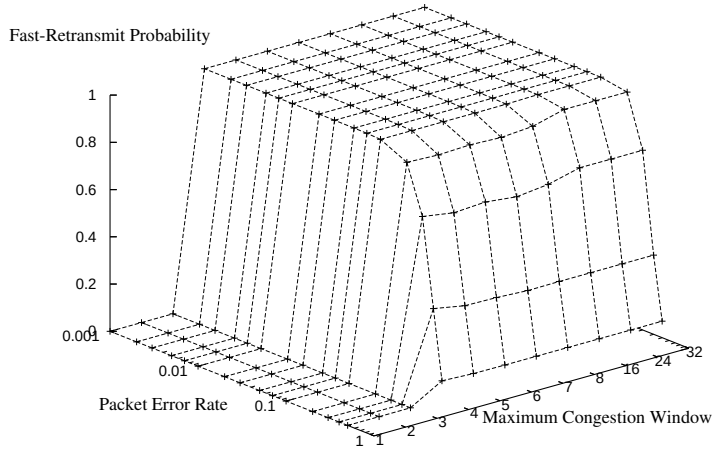
We can obtain the average probabilities that the fast-recovery process of a TCP flow ends with and without timeouts. We call the latter the fast-retransmit probability as all the lost packets can be recovered by fast-retransmit during the fast-recovery process. Let $\bar{Q}$ and Q denote the fast-retransmit and timeout probabilities respectively. $Q + \bar{Q} = 1$ and $\bar{Q}$ is given as follows:

$$\bar{Q} = \sum_{i=1}^{W_{max}} \sum_{l=0}^{i} \pi_i \bar{\gamma}_{il}. \tag{4.22}$$

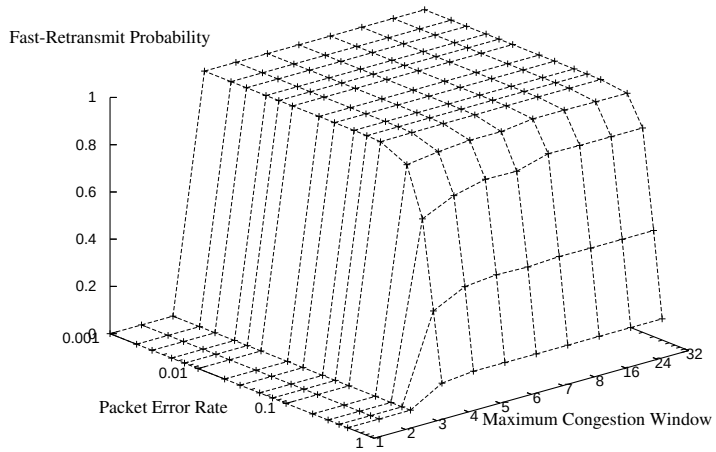
Fig. 4.6 shows the fast-retransmit probabilities $\bar{Q}$ defined by (4.22) for different packet error rates and number of hops. In both TCP variants, for $q = 0 \sim 0.1$, the timeout event hardly occurs. Subsequently, as packet error rate increases, we can see that the fast-retransmit probability has a precipitous drop, which implies a sharp increase in timeout probability. This is consistent with the results in Fig. 4.4 where the throughput drops significantly due to timeouts occurring to TCP. In addition, it is observed that the fast-retransmit probability also decreases with the increase in number of hops due to the large packet loss probability for long linear chains. Compared to the results in [53] and [60], the fast-retransmit probability in ad hoc networks is much higher than that in networks with one lossy link due to the existence of a *long retry limit* in IEEE 802.11 based wireless networks.

Furthermore in Fig. 4.7, we can obtain the relationship between $\bar{Q}$ and W_{max}, where $\bar{Q}$ is plotted against packet error rate and maximum congestion window size. W_{max} varies among 1 – 8, 16, 24, and 32. It is obvious that the Three-Duplicate-ACK events will never occur for $W_{max} \leq 3$. For $W_{max} > 3$, the fast retransmit probabilities for different W_{max} are almost identical for a given packet error rate. As a result, the value of W_{max} has no significant impact on the throughput when considering the impact of channel error induced packet losses.

Finally, in both Fig. 4.6 and Fig. 4.7, it can be seen that Reno has higher timeout probabilities than Impatient NewReno with heavy packet error rates, which matches the results in Fig. 4.4 where Impatient NewReno outperforms Reno in terms of throughput.



(a) Reno



(b) Impatient NewReno

Figure 4.7: Fast-Retransmit probability for different W_{max}

4.7 Concluding Remarks

In this chapter we have proposed a model for TCP Reno and Impatient NewReno that captures the details of the fast-recovery process in the event of packet losses induced by channel error over IEEE 802.11 based n-hop static string ad hoc networks. Considering the spatial reuse property of the wireless channel, the model takes into account the different proportions between the interference range and the transmission range. The model also investigates the interactions between 802.11 MAC, DSR and TCP protocols. Our analysis emphasizes the critical need for studying interactions between protocol layers when designing wireless network protocols.

A node that drops a packet induced by channel error interprets this as a route failure and triggers a route re-establishment process. This results in degradation of the TCP throughput performance. Our model provides the mathematical calculation of the time spent on the detailed DSR route re-establishment process, including route discovery as well as route reply message and route error message transmissions. The DSR routing protocol is used in our analysis instead of other ad hoc routing protocols, because it can facilitate the analysis. As part of future work, we can extend our model to include other routing protocols.

We assume the packet error induced by the wireless channel is i.i.d distributed. We find that the TCP performance for different path length varies with different values of the *long retry limit*, and the default value of four does not always provide the best TCP performance for packet error rate $q > 0.1$. Our model provides us with a theoretical basis for the design of an optimum *long retry limit* to eliminate false route breakages.

The proposed model is based on the semi-Markov renewal reward process and has been thoroughly evaluated through random event simulations using GloMoSim. It is shown that NewReno outperforms Reno with heavy packet loss scenarios in terms of throughput and timeout probability. For both Reno and NewReno, the model is quantitatively valid for packet error rate ranging from 0 to $\sim$0.1, and is qualitatively valid for packet error rate greater than

0.1. The results show that the TCP throughput always deteriorates less than ∼10% when the packet error rate ranges from 0 to 0.1. Compared to previous work which considers only one lossy link in the network [53], our results show that the impact of the channel error is reduced significantly due to the packet retransmissions on a link basis and the small BDP values of ad hoc networks. In addition, the results also show that the timeout probability of a TCP flow is negligible when the packet error rate ranges from 0 to 0.1, and then increases sharply when the packet error rate is more than 0.1. The timeout probability is also proportional to the number of hops in ad hoc networks. Finally, the results show that the maximum congestion window size ($W_{max} > 3$) of a TCP flow has no significant impact on the throughput and the timeout probability.

Chapter 5

DTPA: A Reliable Datagram Transport Protocol over MANETs

5.1 Introduction

This chapter focuses on two key functions for a transport protocol: the congestion control algorithm and the strategy used to guarantee reliable delivery, and propose an effective reliable transport protocol over MANETs, which we call DTPA (Datagram Transport Protocol for Ad hoc networks).

As a prevalent reliable transport protocol in the Internet, TCP uses two key techniques: AIMD (Additive Increase Multiplicative Decrease) congestion control and cumulative ACK technique to guarantee delivery. However, with these two techniques, TCP becomes lowly efficient in ad hoc networks that have a much lower BDP and frequent packet losses due to various reasons, since TCP adjusts its transmission window based on packet losses. BDP represents the maximum amount of allowed unacknowledged data in flight at any moment in the network. The value of BDP plays a critical role in the congestion control algorithm for a window based transport protocol like TCP. Fig. 5.1 shows the dependency of a transport protocol on a network with various BDP. TCP was originally designed for a general wired

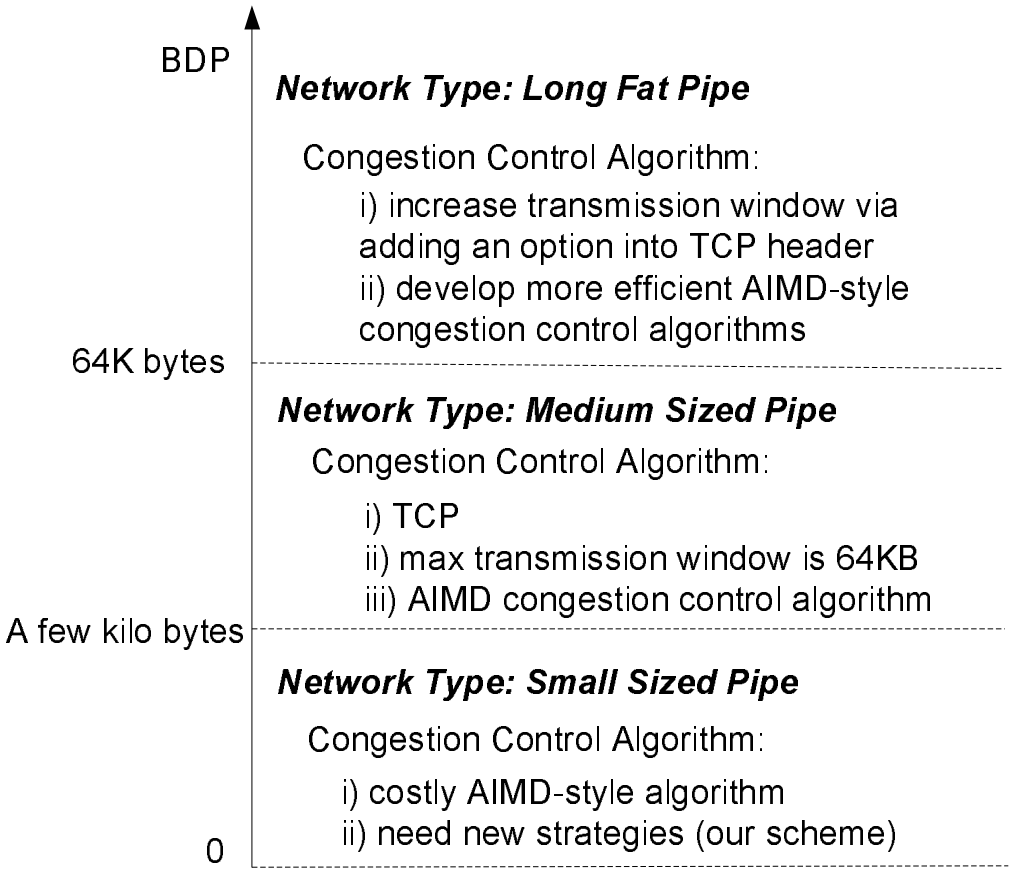


Figure 5.1: The dependency of congestion control algorithm on BDP

network where the BDP is not very large and packet loss rarely occurs. However, with the emergence of various types of networks such as high-speed and satellite networks, the way that TCP works can no longer guarantee a good utilization of the path bandwidth. In the past years, intense efforts have been put on developing new algorithms to fully utilize the network pipeline for those *Long Fat Pipe* networks with large bandwidth and long propagation delay. Nowadays, the newly emerged MANETs with extremely small BDP and frequent packet losses pose another major challenge to the existing transport protocol.

It is known that, due to the spatial reuse property of 802.11 MAC protocol in MANETs, BDP of a connection approximates 1/4 of the path length and is only a few packets. This property has been investigated in details in [7] and revisited in [5, 8] under TCP perspective. Under such a condition, excessive packets are pumped into the network using a large

transmission window relative to its BDP, resulting in a heavy congestion and large packet delay. Hence, it is desirable that the system knows the BDP for each connection in advance so that it can limit the amount of traffic pumped into the network to maintain the optimum throughput. In traditional wired networks, the TCP source is not aware of the available BDP but dynamically determining it by creating congestion during the transmission. However, in MANETs, BDP can be determined in advance by using some routing protocols such as DSR and AODV. This allows the transport layer to intelligently set its transmission window before the connection establishment.

In our scheme, the transmission window of DPTA is fixed to a small value. We have developed a mathematical model to find the optimum transmission window size which happens to be the BDP value plus 3. With such a small value, any AIMD-style congestion control algorithm becomes costly and hence is not necessary. Specifically, with the small transmission window, the source will not proactively generate heavy congestion in the network. In addition, the source will not misbehave in throttling traffic based on the detection of packet losses, since packet losses cannot be an accurate congestion sign in MANETs.

Different from AIMD algorithm in TCP, with the small BDP, the congestion control mechanism of DPTA becomes less important in MANETs. In contrast, the detection and recovery of packet losses in MANETs are more critical because of the frequent packet losses due to reasons such as buffer overflow, channel error, MAC layer contention and mobility. Thus, more efforts should be put on the scheme's ability to quickly detect and recover packet losses.

To guarantee reliable delivery, various ACK schemes have been proposed for unreliable flows in the past and these ACK schemes can be divided into three categories: positive ACK (PACK), negative ACK (NACK) and selective ACK (SACK). With PACK and NACK, the reliable detection of each lost packet requires at least one ACK packet. For instance, in TCP's cumulative PACK scheme, to detect a packet loss, it needs at least three duplicate ACKs, i.e.,

four identical ACKs without any other intervening packets. Also, it is derived in [60] that at most three packet losses within one window can be recovered with TCP's PACK scheme under certain conditions. For a network with a high packet loss frequency, TCP has to heavily rely on timeout to detect packet losses, which degrades the TCP performance significantly. Prompted by the deficiency of TCP's PACK scheme, SACK scheme is proposed and one SACK option is added to the header of ACK packets. The option carries both the negative and positive information of the packet transmissions. However, since the TCP option field has a fixed length of 40 bytes, the ACK packets can carry at most four blocks of information which contain contiguous sequence space occupied by data that has been received. It is possible that the TCP option often needs to give out some space for other functions such as Timestamp. As such, the number of SACK blocks is further reduced. Under such a condition, the SACK scheme can not lead TCP to a good performance.

In our scheme, we employ a cumulative bit-vector based SACK scheme where each bit in the vector stands for the receiving status of one packet. Hence, each ACK packet can acknowledge a wide range of packet with small overheard. Correspondingly, our transport protocol essentially becomes a datagram oriented protocol.

The remainder of this chapter is organized as follows. Section 5.2 illustrates the novel transport protocol in detail. In Section 5.3, a mathematical model is developed and provides the way to determine the optimum transmission window used in the protocol. Section 5.4 has a comparative evaluation to our proposal by simulation. We presents the conclusions for this chapter in section 5.6.

5.2 Scheme Illustration

In this section, we outline the key design elements of our proposed scheme. Unlike TCP where the congestion control and reliability mechanisms are tightly coupled through dependency on

ACK arrival, these two mechanisms are decoupled in our scheme.

(i) Datagram based Technique

A fundamental notation in our design is that, different from byte stream based transport protocol, DTPA provides datagram oriented services. Each datagram is sequenced instead of each octet. The sequence number in each data header does not represent the highest byte of that data like TCP. With a datagram protocol, IP fragmentation of a packet is highly undesirable during the transmission, because the fragmentation can cause inefficient resource usage due to the incurrence of more MAC overhead transmission. Moreover, the loss of a single fragment requires the source to retransmit all of the fragments in the original datagram, even if most of the fragments are received correctly at the destination. In the past, some schemes [68] have been proposed to avoid fragmentation in the Internet. Our proposal can be utilized with them together to avoid fragmentation. For ease of explanation, we assume that the datagram is too small to be fragmented in our system.

(ii) Cumulative Bit-Vector based SACK Technique

Fig. 5.2 shows part of ACK header that illustrates the new ACK strategy. H is the highest sequence number of the datagram that has been received. There is a bit in the header named L to indicate the existence of an out-of-order data. The L flag is turned on whenever an out-of-order segment arrives, which implies that there may be missing packets. The vector field consists of k bits representing the receiving status of a set of earlier packets. Let a_i be a single bit that indicates the arrival status of the packet with sequence number $\eta = H - i$. The a_i is set to 1 if η has been received and 0 otherwise.

We use the L flag for the realization of a cumulative acknowledgment mechanism. If the L flag in a received ACK packet is turned off, this means the transmission is in a normal status without data packet losses or out-of-order transmissions. Hence, an acknowledgment

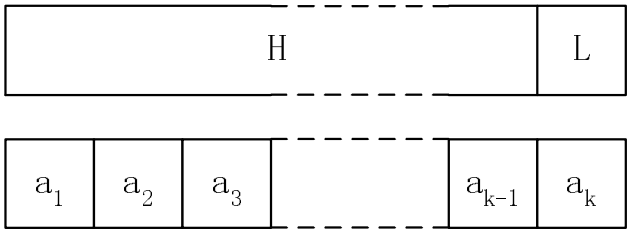


Figure 5.2: Part of ACK header

of sequence number H indicates that all datagrams up to H have been received. The condition of a successful packet transmission with a sequence number η can be expressed as:

$$\{L = 0, \eta \leq H\} \cup \{\eta = H - i, a_i = 1\} \quad 1 \leq i \leq k. \tag{5.1}$$

(iii) Fixed window based flow control

DTPA is based on a sliding-window scheme where given the number of hops n, the window size is fixed at $w(n) = BDP(n) + \alpha(n)$, where $\alpha(n)$ is a small value used to guarantee that there can be enough packet transmissions and ACK arrivals at the source in the case of packet losses. For instance, with $w = BDP = 1$, the source cannot detect the packet loss through ACK because there is only one packet in transmission and it is the one that is missing. In our scheme, although w exceeds the available BDP by $\alpha(n)$, this strategy can guarantee that the network pipeline is at least fully utilized and that there is no heavy congestion and contention in the networks. The value of $\alpha(n)$ is derived later in Section 5.3.

The DTPA source grabs a number of bytes from the transmit buffer, encapsulates these bytes into a datagram with an appropriate sequence number, and tries to transmit them reliably to the destination with the permission of transmission window. Upon receiving an ACK for an outstanding packet, the DTPA source performs the following steps: i) checks whether there are possible packet losses; ii) computes how many packets can be sent; and iii) sends packets. On the other hand, whenever a timeout timer related to a packet expires, the source marks the packet with a new timer and executes steps ii)-iii) as above.

(iv) New Retransmission Technique

We have a new mechanism for deciding to retransmit the lost packet. Similar to TCP, the DTPA source decides to retransmit a packet by either via the receipt of ACKs or a timeout event. For the former, the source DTPA depends on the L flag and the bit vector in ACK header to detect the possible packet losses. It keeps a retransmission buffer to store the incoming ACK information with a turned-on L flag on a per connection basis. A packet with a sequence number $\eta = H - i$ is assumed to be lost only if:

$$L = 1, \quad a_i = 0 \qquad 1 \leq \gamma \leq i \leq k. \tag{5.2}$$

where the condition $\gamma \leq i$ indicates at least γ ACKs carry the information of the packet that has not been received. Under condition (5.2), the lost packet is retransmitted immediately. When receiving ACKs with the header $a_i = 0$ $(i < \gamma)$, the source assumes that the packet is still in the out-of-order transmission in the network and is not missed yet. However, since the packet may be lost with a certain probability, a new packet will be transmitted to conservatively guarantee that there are at least a fixed-window size of packets in flight. Here, γ is set to 2 in our implementation.

For a timeout event, the DTPA source assumes that there is no outstanding packet in the network. Since the transmission window is fixed at a value greater than one, besides retransmitting those lost packets recorded by the retransmission buffer, the source also transmits new packets if the transmission window allows. In DTPA, the RTO timer does not increase exponentially like TCP, because the timeout event here implies a window's worth of packet losses rather than a heavy congestion in the system.

5.3 Mathematical Analysis

This section presents a parametrised, analytically tractable model for the proposed DTPA protocol. The model can yield quantitative comparisons between the DTPA and TCP proto-

col, and can be used for evaluating the effects of the various parameters. It also provides the theoretical basis for determining the appropriate transmission window $w(n)$.

5.3.1 Network Model

The objective of modelling a transport protocol is usually to calculate the end-to-end throughput with given network parameters. Our basic network model assumes an infinite source that releases packets into a n-hop static linear chain. All the nodes are distributed with the same distance to its neighboring nodes. The distance between any two adjacent nodes satisfies the situation where one node can only transmit packets to its one-hop neighbors, interfere with its two-hop neighbors and cannot hear other nodes.

Table 5.1 lists the variables used in our analysis. In our model, we consider that both the forward and reverse links drop packets randomly and independently of one another. We assume that the forward channel of each link has an identical loss probability of p_1 and the reverse loss probability of each link is identically p_2. Correspondingly, the data and ACK loss probabilities on a path basis are given by $p_d = 1-(1-p_1)^n$ and $p_a = 1-(1-p_2)^n$ respectively. Although this assumption is not completely accurate, it facilitates the analysis and allows us to evaluate how the DTPA protocol can tolerate different degrees of packet losses. In our mathematical model, the expression for the throughput μ is derived as a function of the data loss probability p_1 on one wireless link, ACK loss probability p_2 on one wireless link, the number of hops n and transmission window w.

5.3.2 Throughput Calculation

Similar to TCP, the behavior of the DTPA protocol can also be regarded as a cyclic evolution. In our model, we define one cycle as the interval between the end of one timeout event and the end of the next timeout event. The cycles form a renewal process due to the independent packet losses. The cycle duration is a random variable which is further divided into two non-

Table 5.1: Glossary

n	the number of hops in a string topology
w	transmission window size
μ	throughput considering packet losses
μ^*	throughput without considering packet losses
p_1	data loss probability at each link
p_2	ACK loss probability at each link
p_d	data loss probability at a n-hop chain
p_a	ACK loss probability at a n-hop chain
x	the duration before a timeout event within a cycle
y	the timeout duration within a cycle
τ	the duration with packet transmissions within a timeout
N_x	the number of packets transmitted within x
N_y	the number of packets transmitted within y
δ_o	the probability of a timeout event
λ_r	the probability of a retransmission
λ_s	the probability of a successful transmission
λ_o	the probability of a timeout event happening to a non-retranmission
ζ_s	the probability of a successful retransmission
ζ_o	the probability of a timeout event happening to a retransmission

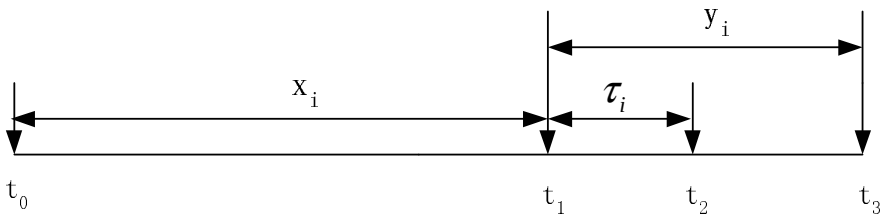


Figure 5.3: An illustration of a cycle

overlapping components at the time when the timeout occurs. Fig. 5.3 shows an example of the i-th cycle. Let x and y denote the duration before and after the instant of a timeout within a cycle respectively. The cycle duration is given by $x + y$. Let N_x and N_y denote the number of packets transmitted within x and y respectively. Hence, the total packets transmitted within a cycle is $N_x + N_y$. As such, we define the DTPA throughput for a n-hop static linear chain, μ as follows:

$$\mu = \frac{E[N_x + N_y]}{E[x + y]} = \frac{E[N_x] + E[N_y]}{E[x] + E[y]}. \tag{5.3}$$

In order to derive an expression for μ, the long-term steady-state DTPA throughput, we must next derive expressions for the mean of N_x, N_y, x and y.

A. Derivation of $E[N_x]$ and $E[x]$

Consider a period of x as shown in Fig. 5.3. During this period, timeout events never take place. Given a packet timeout probability δ_o, the probability that exactly j packets are successfully acknowledged before a timeout event is then given by:

$$P[N_x = j] = \delta_o(1 - \delta_o)^j, \quad j = 1, 2, \cdots . \tag{5.4}$$

The mean of N_x is thus:

$$E[N_x] = \sum_{j=1}^{+\infty} j\delta_o(1 - \delta_o)^j = \frac{1 - \delta_o}{\delta_o}. \tag{5.5}$$

Considering those retransmitted packets, $E[x]$ is given by:

$$E[x] = \frac{E[N_x](1 + \zeta_s)}{\widehat{\mu}}, \tag{5.6}$$

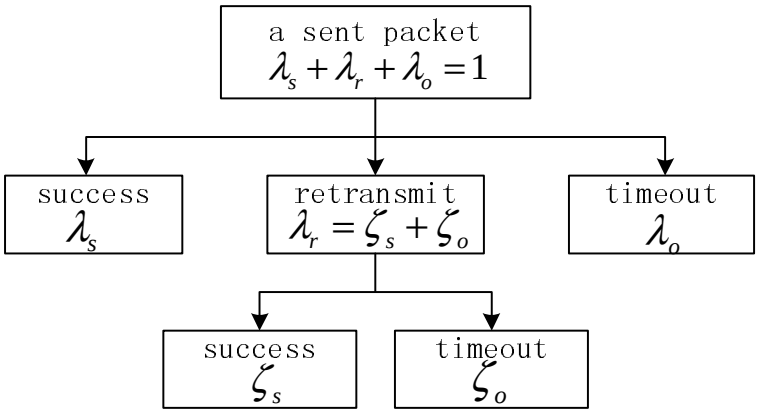


Figure 5.4: An illustration of a packet transmission

where ζ_s is the probability that a packet is successfully retransmitted; $\widehat{\mu}$ is the upper bound throughput of a n-hop linear chain derived in Chapter 3.

The derivation of δ_o and ζ_s is presented as follows. Fig. 5.4 shows that, within a cycle, a sent packet experiences one of the three scenarios: i) being successfully transmitted with probability λ_s; ii) the packet is lost and is detected via a timeout event with probability λ_o; iii) the packet is lost and is detected by the receipt of ACKs with probability λ_r. These three scenarios satisfy the condition $\lambda_s + \lambda_o + \lambda_r = 1$. For the third case, the retransmitted packet may succeed or fail in transmission with probability ζ_s or ζ_o, and hence there is $\lambda_r = \zeta_s + \zeta_o$. The failed retransmission further leads to a timeout event. Therefore, given a sent packet, we can know that finally the packet can either be transmitted successfully with probability $\lambda_s + \zeta_s$ or lead to a timeout event with probability $\delta_o = \lambda_o + \zeta_o$.

Given a sent packet, the packet loss can directly lead to a timeout event with probability λ_o because of two scenarios as follows. Firstly, with the bit-vector based SACK scheme, there can be up to k ACK packets acknowledging every data packet. A timeout event takes place when all the w ACKs within the transmission window are missed. Secondly, recall (5.2). For a lost packet with sequence number η, all the ACKs satisfying the condition (5.2) are missed, and not all the ACK packets with $i < \gamma$ are missed. The source receives ACKs with $i < \gamma$,

which leads to up to γ new packet transmissions. A timeout event occurs if the ACKs for these following transmitted packets are all missed, too.

Recall that γ is set to 2 in our implementation. The probability λ_o is then given by:

$$\lambda_o = f(w) + p_d(1-p_d)(1-p_a)f(w-2)f(2), \tag{5.7}$$

where $f(w) = (p_d + (1-p_d)p_a)^w$. This is because each opportunity for an ACK may be missed due to either a data packet loss in the forward path or an ACK packet loss in the reverse path - with probability $p_d + (1-p_d)p_a$.

In the DTPA protocol, a packet has only one chance to be retransmitted within one cycle. Hence, a failed retransmission can only be detected via timeout events. The probability that a timeout event happens to a retransmission is given by:

$$\zeta_o = \lambda_r(p_d + p_a(1-p_d)f(w-1)). \tag{5.8}$$

Similarly, to derive λ_r, given a lost packet, the scenarios that a retransmission event occurs are as follows. First of all, not all the ACK packets satisfying the condition (5.2) are missed. Secondly, if the first case is not true, then not all the ACK packets with $i < \gamma$ are missed, and there is at least one successful transmission among the following newly transmitted packets triggered by ACK with $i < \gamma$. Therefore, the probability λ_r is given by:

$$\lambda_r \;=\; p_d(1-f(w-2)) + p_d(1-p_d)(1-p_a)f(w-2)(1-f(2)). \tag{5.9}$$

B. Derivation of $E[N_y]$ and $E[y]$

To derive $E[N_y]$ and $E[y]$, consider a sample of a timeout period as in Fig. 5.5. We view the DTPA behavior in terms of rounds where a round starts when the sender begins the transmission of a window of packets and ends when the sender receives an acknowledgment for one or more of these packets. A timeout period is the duration of a RTO within which a transmitted data packet cannot get the corresponding acknowledgment before the RTO timer expires. Obviously, $E[y] = RTO$.

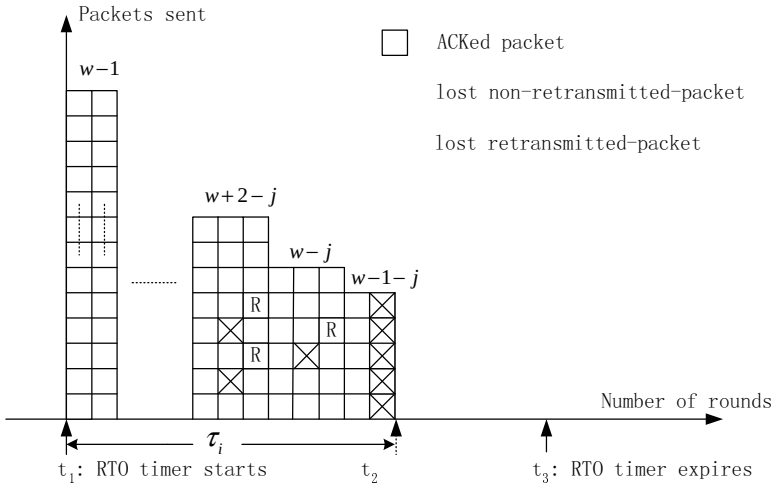


Figure 5.5: A sample of a timeout event

Different from TCP, in DTPA, because of the fixed transmission window size, there can be continuous new rounds of transmitted packets during the timeout period. The number of transmitted packets within each round is dependent on the number of ACKs received. As shown in Fig. 5.5, the effective transmission window of DTPA during the timeout actually starts from $w-1$, since the ACKs received by the source can only acknowledge at most $w-1$ packets within this round due to the lost packet which results in a timeout event. In the following rounds, more ACKs may not be able to arrive due to retransmission failure. The effective transmission window is further decreased until it reaches zero. During the timeout, let τ denote the period within which there are packet transmissions. Obviously, the longer the τ, the better the throughput.

In Fig. 5.5, the period of τ ends in two ways: i) all the last $w-1-j$ ACKs are missed; ii) there are still effective transmissions within the last round, and $\tau = RTO$. For the former, the last round can end with the two scenarios given previously in the derivation of (5.7) and the second scenario in the derivation of (5.8). Hence, given a packet loss, the probabilities that τ ends with three different scenarios are:

$$a_1(w,j) = f(w-1-j) \quad 0 \le j \le w-2, \tag{5.10}$$

$$a_2(w,j) = p_d(1-p_d)(1-p_a)f(w-1-j) \quad 0 \le j \le w-3, \tag{5.11}$$

$$a_3(w,j) = \lambda_r p_a(1-p_d)f(w-2-j) \quad 0 \le j \le w-2. \tag{5.12}$$

Given j retransmission losses, the probability that there can be m packets transmitted is given by:

$$h(m,j) = \binom{m}{j}(\lambda_r p_d)^j(\lambda_s + \zeta_s)^{m-j}. \tag{5.13}$$

Note that here λ_r, λ_s and ζ_s are a function of effective transmission window w which varies during the timeout. To facilitate the analysis, we conservatively regard their parameter transmission window as $w-j$, so that the computed $h(m,j)$ is the lower bound of itself.

Table 5.2: BDP of a n-hop Linear Chain

Number of Hops	BDP
1, 2, 3	1
4, 5, 6	2
7, 8, 9	3
10	4

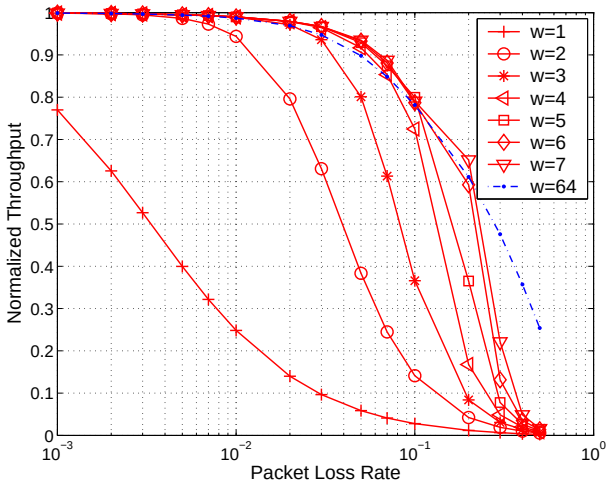
From (5.13), there are $m-j$ successful transmissions before τ ends. The mean of N_y is thus:

$$\begin{aligned} E[N_y] &= \sum_{j=0}^{w-2}\sum_{m=j}^{N_y^{max}-w+1}\frac{m-j}{w-1}h(m,j)a_1(w,j) \\ &+\sum_{j=0}^{w-3}\sum_{m=j}^{N_y^{max}-w}\frac{m-j}{w-2}h(m-1,j)a_2(w,j) \\ &+\sum_{j=0}^{w-2}\sum_{m=j}^{N_y^{max}-w+1}\frac{m-j}{w-1}h(m,j)a_3(w,j) \\ &+\sum_{j=0}^{w-2}\sum_{m=N_y^{max}-w+2}^{N_y^{max}-j}\frac{m-j}{w-1}h(m,j). \end{aligned} \tag{5.14}$$

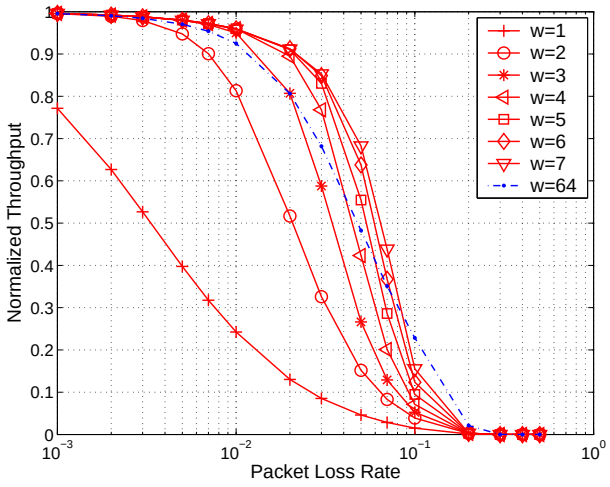
where N_y^{max} is the maximum number of transmitted packets during the timeout. We approximate it as $N_y^{max} = \frac{RTO\times\mu^*}{1+\zeta_s}$. The final item in (5.14) corresponds to the case that there are still effective transmissions within the last round and $\tau = RTO$.

5.3.3 Determine $w(n)$

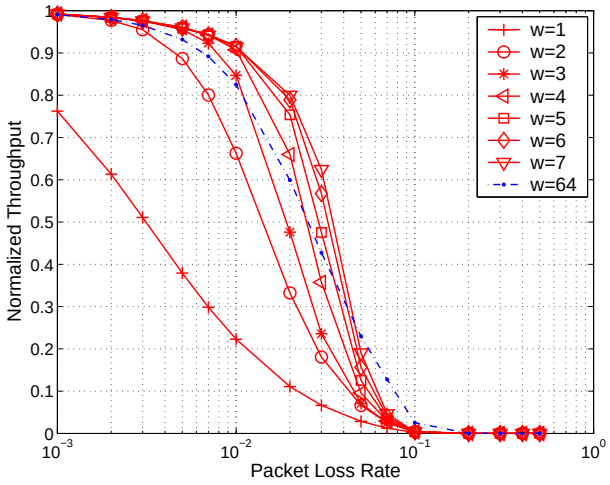
For a n-hop chain, we need to determine a proper transmission window size $w(n) = BDP(n)+\alpha(n)$ for the throughput defined by (5.3), where $\alpha(n)$ is unknown. Table 5.2 lists the BDP of the path in MANETs. Fig. 5.6 shows the throughput for different number of hops n. Here, the throughput μ is normalized to $\widehat{\mu}$ of the path. We only show the results for n =1, 4 and 8, as similar results are obtained for other n-hop linear chains. The x-axis of each graph indicates



(a) $n = 1$



(b) $n = 4$



(c) $n = 8$

Figure 5.6: Normalized throughput of the analytical model: RTO= 4 ticks, 1 tick = 500 ms

packet loss rate of each transport layer packet including data packet and ACK packet which satisfies $p_1 = p_2$. To minimize the complexity, the values of p_1 and p_2 are set to be identical in our study. Each curve in the sub-figure corresponds to a specified transmission window size which varies between $w = 1 \sim 7$ and $w = 64$. We can see that, with the increase of packet loss rate, the throughput decreases as expected. For the case of $w = 1 \sim 7$, the throughput increases as the transmission window becomes large. However, this phenomenon is not true for all the values of w. In all the sub-figures, some points in the case of $w = 64$ shows poorer throughput performance than those with $w < 64$. This implies that for a given packet loss rate in a n-hop chain, there must exist an optimum value of w with which the throughput can be maximized. However, such an optimum value of transmission window may be very large, which leads to heavy congestion and large latency in the network. To avoid these problems, a small value of transmission window is expected. Considering the tradeoff, we choose a sub-optimum value of transmission window which meets $\alpha(n) = 3$. This is because for all the cases in the figure, given a large packet loss rate range from 0 to 0.1, it can be seen that the throughput varies sharply in the case of $w(n) <= 3$ and slowly in the case of $\alpha(n) >= 3$. It should be noted that we only consider the number of hops $1 \leq n \leq 10$ here, because a large number of hops is undesirable in reality due to the fragile connectivity of MANETs.

5.4 Performance Comparison Study

In this section, we use the GloMoSim simulator to validate the analytical model and to evaluate the performance of DTPA. We run simulations in static topologies, because the study of mobility is not within the scope of this chapter and our proposal can work together with other strategies to handle the mobility issues. In the simulation scenarios, all nodes communicate with identical, half-duplex wireless radios which have a bandwidth of 2Mbps. The radio propagation model uses Friss free-space attenuation ($1/r^2$) at near distances and

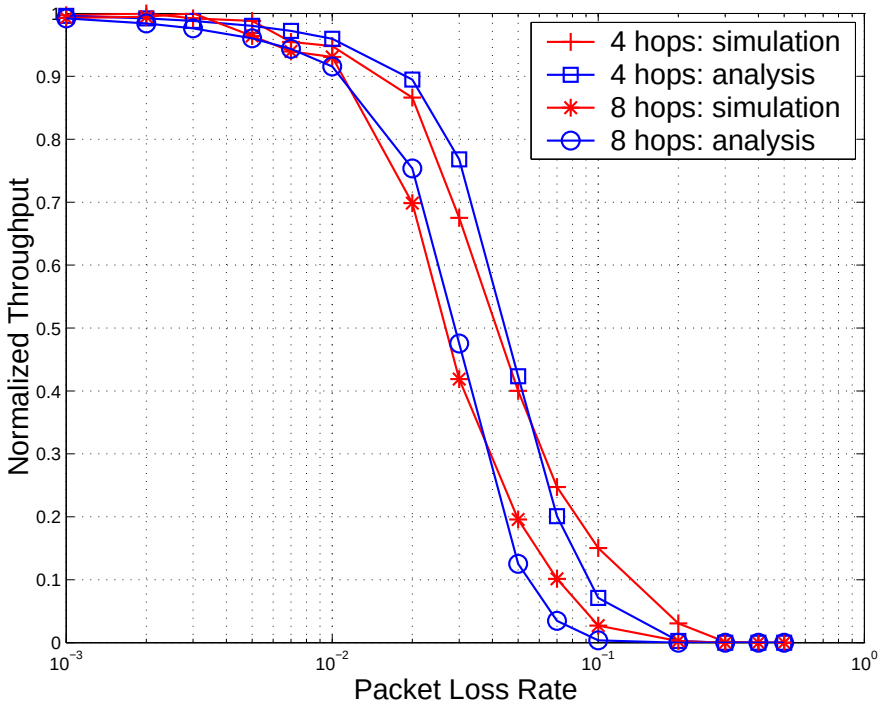


Figure 5.7: A comparison between simulation and analytical results

an approximation to two ray ground ($1/r^4$) at far distance by assuming specular reflection off a flat ground plane. DSR is used as the routing protocol. The packet size is set to 512 bytes and fragmentation does not take place during transmission.

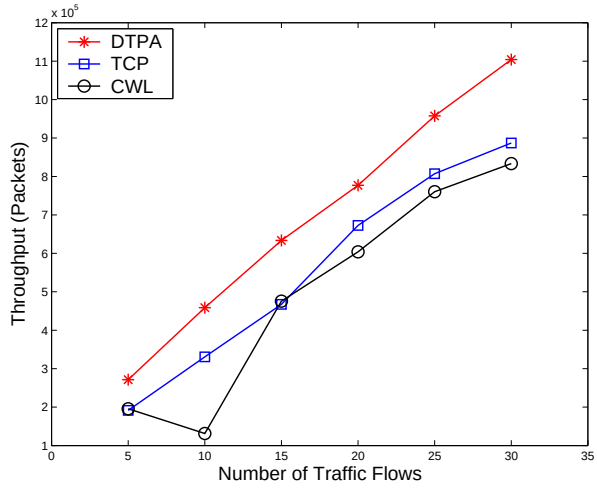
To validate the analytical model, we run a DTPA flow over a n-hop static linear chain defined in the model. Each simulation is run for 500 seconds. The data and ACK packets are discarded with probability $p_1 = p_2$ by the receiving node rather than dropped due to buffer overflow or channel error, because modelling of buffer overflow and channel error in a mathematical way is not within the scope of this chapter. Moreover, our goal is to analyze the dependency of DTPA on the packet losses and to investigate whether DTPA can work better than other protocols in the event of frequent packet losses.

Fig. 5.7 shows that the throughput comparisons between simulation results and theoretical results of the proposed analytical model for different values of n. Here, we only show the results for $n = 4$ and 8, as similar results are obtained for other topologies. The throughput

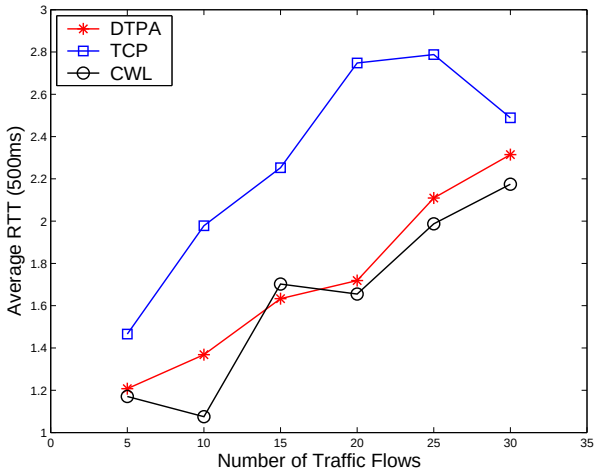
is plotted against packet loss rate of each transport layer packet, which corresponds to $p_1 = p_2$. Both results show that the throughput degrades with either the increasing packet loss rate or the number of hops n. The results from the analysis of the proposed model match closely with the simulations. It can be seen that the packet loss rates from 0 to 0.01 do not have significant impact on the throughput performance in multi-hop ad hoc networks, where the DTPA throughput always deteriorate less than 10%. With higher packet loss rates exceeding 0.01, the simulation results and analytical results show similar tendency and there is a precipitous drop in the DTPA throughput.

We further compare our DTPA protocol to TCP and CWL. The CWL scheme attempts to limit the maximum TCP transmission window to just the BDP of the path. The network we simulate consists of 45 static nodes which are randomly and uniformly distributed in a field. Two fields with different areas of $1000m \times 1000m$ and $2000m \times 2000m$ are simulated. The number of traffic flows increases from 5 to 30 with a step size of 5, using persistent traffic flows and randomly generated source–destination pairs. Each simulation is conducted for 3000 seconds. Each data point shown is averaged from 30 simulations. In each simulation run, we measure four metrics: throughput, average RTT of each flow, average maximum IP layer queue size in the network and the number of retransmissions. The values of p_1 and p_2 are set to 0 so that packet losses are due to realistic reasons instead of manual configuration of p_1 and p_2. Hence, the packets are lost mainly due to MAC contention which cause false routing failures in the network. The performance comparison between the three protocols is shown in Fig. 5.8. We only present the results in the case of $2000m \times 2000m$, as similar results are obtained for the case of $1000m \times 1000m$.

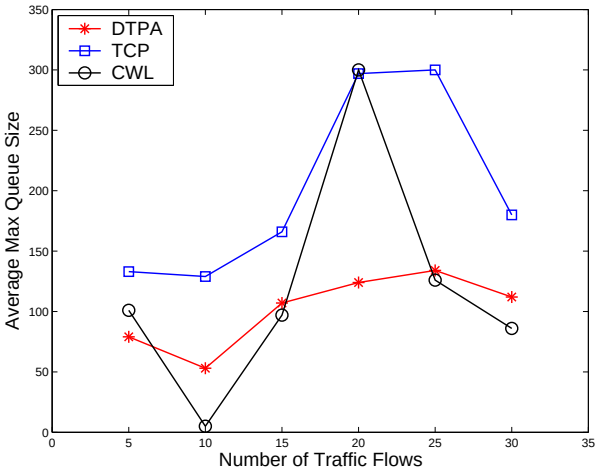
Fig. 5.8 shows that our proposed DTPA protocol yields performance improvements in all the four metrics we have measured. Comparing to TCP, the throughput, average RTT, average maximum IP queue size and the number of retransmissions can respectively be improved by up to 41%, 37%, 60% and 98%. Our results in Fig. 5.8(b) and Fig. 5.8(c) show that RTT



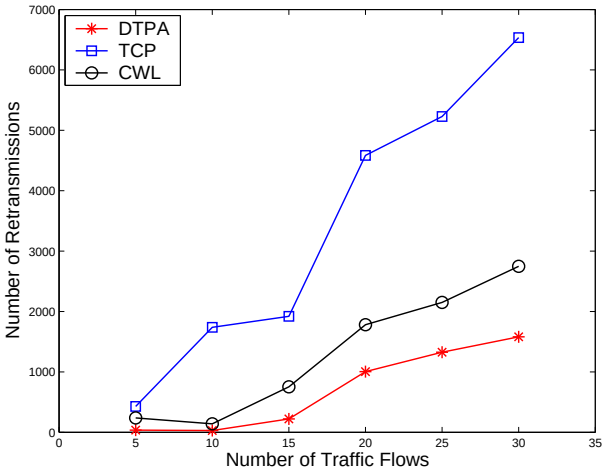
(a) throughput



(b) average RTT



(c) average max IP queue size



(d) number of retransmissions

Figure 5.8: Performance of the DTPA protocol

and average maximum IP queue size for CWL have better performance than DTPA at some points, which is due to the small transmission window size of CWL. However, it can be seen that the CWL throughput in Fig. 5.8(a) are much worse than that of TCP and DTPA, which is different from the results obtained in [45], whereby the CWL scheme improves TCP throughput performance. This is because in the literature, the CWL scheme was studied only in a static string topology with a single flow where packet losses hardly occur. CWL can maintain its optimum transmission window during the whole lifetime of the connection. In contrast, in our topologies with multiple traffic flows, packet losses occur frequently due to MAC contention. Using a small transmission window, the CWL source sends a limited amount of packets into the networks, so that it cannot detect packet losses via the reception of enough ACK packets. The source has to heavily rely on timeout events to detect and retransmit the lost packets, and correspondingly the CWL goes back to slow start phase with its transmission window dropped to one, which results in the throughput degradation. As such, our DTPA scheme can be utilized in a general ad hoc network for efficient bandwidth utilization.

5.5 DISCUSSION

5.5.1 Comparisons with Rate–Based Schemes

In this paper, we have compared the DTPA protocol to the CWL scheme which employs a window-based congestion control approach. We have also tried to compared the DTPA to other rate-based approaches. After studying several related papers [41], [42] and [44], we find that we cannot reproduce the accurate simulation results according to their algorithms, because the rate-based schemes involve complicated computation and we cannot guarantee our code's accuracy. Hence, in our paper, we do not provide the comparison results between our scheme and other rate-based approaches.

However, it is still believed that our scheme outperforms the rate-based ones. The reason is as follows. Firstly, it is obvious that rate-based flow control usually incurs additional computation complexity or overhead, especially for the explicit rate based scheme like [41], where each intermediate node needs to calculate the rate and pass the information to the source. Secondly, the rate obtained by the source is usually less than $\frac{1\text{packet}}{4\text{–hop propagation delay}}$, which approximates the maximum packet transmission rate for ad hoc networks, since there is only one transmission every 4 hops due to the spatial reuse property of ad hoc networks. For example, the scheme in [44] adaptively sets its transmission rate using an estimate of the current 4-hop propagation delay and the coefficient of variation of recently measured round trip times. The scheme in [42] computes the sending rate based on round-trip times, which implies the average sending rate cannot exceed the maximum transmission rate. Both the schemes show that the number of packets transmitted in the networks is no more than BDP, since BDP of a path approximates 1/4 of the path length. When these two schemes are utilized together with the TCP's cumulative ACK technique, the schemes become similar to the CWL scheme where maximum TCP transmission window to just the BDP of the path. As a result, these rate-based schemes have the same drawback as the CWL scheme and they cannot detect packet losses quickly by receiving enough ACK packets.

Hence, even though we only compared our protocol to one window based TCP variant recently proposed in ad hoc networks, we believe the comparison is typical and representative enough to judge our scheme's effectiveness.

5.5.2 Fairness

Significant TCP unfairness in ad hoc networks has been revealed and studied during the past several years. It has been pointed out in [69] that TCP unfairness is mainly attributed to the unfairness of MAC protocol which results from the nature of shared wireless medium and location dependency. In order to solve the TCP unfairness in ad hoc networks, different

schemes have been proposed to modify various protocol layers ([69], [70]). Since DTPA employs a window based congestion control coupled with an ACK technique similar to TCP, it is believed that DTPA also experiences severe unfairness among competing flows in ad hoc networks. Hence, in order to provide fairness for DTPA flows in ad hoc networks, new algorithms need to be further developed.

5.6 Concluding Remarks

In this chapter, we propose a novel reliable Datagram Transport Protocol over 802.11 based Ad hoc networks which we call DTPA protocol. Because the BDP in 802.11 based MANETs is very small and can be known before the connection establishment with the usage of some routing protocols such as DSR and AODV, any AIMD-style congestion control algorithm is costly and hence not necessary for ad hoc networks. On the other hand, the strategy to guarantee a reliable transmission and to recover the frequent packet losses plays a more critical role in the design of a transport protocol. With this basis, our scheme incorporates a fixed window based flow control and a bit-vector based selective ACK strategy where the ACK packets contains a vector of bits representing the receiving status of set of earlier packets. A packet is assumed to be lost if the source finds that at least two ACKs carry the loss information of that packet or if the source cannot receive corresponding ACK within the expected time.

Furthermore, we develop a parametrised mathematical model for the behavior of DTPA protocol based on a renewal process. The model can be used for evaluating the effects of the various parameters including the two-way traffic loss probabilities, path length, transmission window of a flow, timeout probability, etc. Based on this model, an optimum transmission window is determined for a n-hop chain and is the value of BDP plus 3. With this value, the DTPA protocol proactively avoids generating heavy congestion. We use GloMoSim simulator

to evaluate the proposed DTPA. The simulation results show that the scheme respectively improves the network throughput, average RTT, average maximum IP queue size in the network and number of retransmissions by up to 41%, 37%, 60% and 98%. DTPA can either be developed as a totally novel transport protocol or be extended as an option of TCP with the deployment consideration.

Chapter 6

Conclusion and Future Work

The main focus of this thesis is the investigation and development of the TCP transport protocol in IEEE 802.11 based mobile ad hoc networks. It is initially noted that the poor TCP performance in MANETs is due to its inability to recognize packet losses caused by four reasons: MAC layer medium contention, channel error, buffer overflow and mobility. Investigation of mobility issues has been carried out by many researchers, and also it has been found that buffer overflow caused packet losses is rare in MANETs. In this thesis, we focused on packet losses due to the other two factors which are attributed to misbehavior of IEEE 802.11 MAC protocol. Our objective was to develop mathematical approaches to model the impact of the two kinds of packet losses and to develop novel schemes and transport protocols to solve the problems caused by packet losses. Our study emphasizes the critical need for studying interactions between protocol layers when design wireless network protocols. The major contributions of the work are summarized in this chapter and some areas for future work are suggested.

6.1 Contributions

To more accurately investigate the impact of different packet losses on the TCP performance, an analytical model has been firstly developed to quantify the upper bound of the end-to-end throughput of a TCP flow across an 802.11 based multi-hop string topology [71]. In the analysis, all the packet losses are removed in the model by making careful and appropriate assumptions: static scenarios, error-free wireless channel, infinite buffer size at each node and infinite *short retry limit*. Therefore, the derived throughput is the upper bound and can be used as a guideline for the best case performance. Compared to the existing work [61], the analysis is unique because it tries to quantify the throughput of a TCP flow after considering the interaction between TCP's congestion window size and 802.11's MAC contention window size. The model considers the existence of contentions, collisions and binary exponential backoffs at the MAC layer as well as how these phenomena affect or are affected by TCP's congestion window size and packet size. Also, the difference between the transmission range and the carrier sensing range is not ignored. Simulations show that the model predicts the upper bound of the TCP throughput to a very high level of accuracy. The work is an important step towards understanding the interaction between TCP segment transmission and IEEE 802.11 control packets (RTS–CTS) transmission. The model is amendable to precise analysis in the future, and allows to isolate packet losses due to different causes and to investigate the impact of different parameters on the TCP throughput performance.

Based on the analysis of the TCP throughput upper bound, packet losses due to MAC layer medium contention and wireless channel error are further investigated. These two kinds of factors caused packet losses in MANETs can be followed by false route breakages which seriously deteriorate TCP performance. It is found that false route breakages due to RTS transmission failures are mainly attributed to the hidden terminal effect caused by MAC layer medium contention. In contrast, wireless channel error is the dominant factor which leads to TCP data packet transmission failures caused false route breakages. We study the two kinds

of packet losses separately.

In the case of false route breakages due to RTS transmission failures, our analysis shows that the likelihood of false route breakages is proportional to the size of TCP segments in a network [72]. Through simulation, we find that false route breakages are also dependent on the path length of a source-destination pair, and a 4-hop linear chain suffers the most serious medium contention caused false route breakages. Our simulation results also show that up to 63% route breakages can be false due to the misbehavior of 802.11 in some mobile scenarios. Under such a situation, the wireless channel usage is in a high possibility to be dominated by the route control packets used for unnecessary route discoveries. Hence, we present a protocol enhancement that enables IEEE 802.11 MAC protocol to alleviate false route breakages due to RTS transmission failures. Our scheme is a simple modification to 802.11 MAC protocol and hence the problem generated at the lower MAC layer is hidden from the upper routing and transport layers. We achieve the goal of alleviating false route breakages by initiating a HELLO message to the sender whenever the number of RTS received by a receiver exceeds a threshold. Simulation results show that the proposed modification substantially reduced false route breakages and improve throughput by up to 35%. The results also suggest that restricting the TCP maximum transmission window to BDP of the path may not always yield good performance. This provides a basis for us to design flow control algorithms in MANETs.

In the case of false route breakages due to TCP packet transmission failures, we propose a packet level model to investigate the impact of wireless channel error on TCP performance over IEEE 802.11 based multi-hop wireless networks [73,74]. The proposed model is based on the semi-Markov renewal reward process, and we study two versions of TCP: Reno and Impatient NewReno. Considering the spatial reuse property of the wireless channel, the model takes into account the different proportions between the interference range and transmission range. We adopt more accurate and realistic analysis to fast-recovery process which is applicable to any kind of networks including wired and WLAN networks. We also investigate the

interactions among TCP, IP and MAC protocol layers, specifically the impact of 802.11 MAC protocol and DSR routing protocol on TCP throughput performance. Our model provides the mathematical calculation of the time spent on the detailed DSR route re-establishment process, including route discovery as well as route reply message and route error message transmissions. In the model, the packet error induced by the wireless channel is assumed to be i.i.d distributed. We find that the TCP performance for different path length varies with different values of the *long retry limit*, and the default value of four does not always provide the best TCP performance for packet error rate $q > 0.1$. Our model provides us with a theoretical basis for the design of an optimum *long retry limit* for IEEE 802.11 MAC protocol to eliminate false route breakages caused by wireless channel error.

In the model, it is shown that NewReno outperforms Reno with heavy packet loss scenarios in terms of throughput and timeout probability. For both Reno and NewReno, the model is quantitatively valid for packet error rate ranging from 0 to $\sim$0.1, and is qualitatively valid for packet error rate greater than 0.1. The results show that the TCP throughput always deteriorates less than $\sim$10% when the packet error rate ranges from 0 to 0.1. Compared to previous work which considers only one lossy link in the network [53], our results show that the impact of the channel error is reduced significantly due to the packet retransmissions on a link basis and the small BDP values of ad hoc networks. In addition, the results also show that the timeout probability of a TCP flow is negligible when the packet error rate ranges from 0 to 0.1, and then increases sharply when the packet error rate is more than 0.1. The timeout probability is also proportional to the number of hops in ad hoc networks. Finally, the results show that the maximum congestion window size ($W_{max} > 3$) of a TCP flow has no significant impact on the throughput and the timeout probability.

Finally in the thesis, we focus on two key functions for a transport protocol: the congestion control algorithm and the strategy used to guarantee reliable delivery, and propose a novel reliable Datagram Transport Protocol over 802.11 based Ad hoc networks which we call DTPA

protocol [75]. Because the BDP in 802.11 based MANETs is very small and can be known before the connection establishment with the usage of some routing protocols such as DSR and AODV, any AIMD-style congestion control algorithm is costly and hence not necessary for ad hoc networks. On the other hand, the strategy to guarantee a reliable transmission and to recover the frequent packet losses plays a more critical role in the design of a transport protocol. With this basis, our scheme incorporates a fixed window based flow control and a bit-vector based selective ACK strategy where the ACK packets contains a vector of bits representing the receiving status of set of earlier packets. Hence, each ACK packet can acknowledge a wide range of packet with small overheard. Correspondingly, our transport protocol essentially becomes a datagram oriented protocol. A packet is assumed to be lost if the source finds that at least two ACKs carry the loss information of that packet or if the source cannot receive corresponding ACK within the expected time. Furthermore, we develop a parametrised mathematical model for the behavior of DTPA protocol based on a renewal process. The model can be used for evaluating the effects of the various parameters including the two-way traffic loss probabilities, path length, transmission window of a flow, timeout probability, etc. Based on this model, an optimum transmission window is determined for a n-hop chain and is the value of BDP plus 3. With this value, the DTPA protocol proactively avoids generating heavy congestion. The simulation results show that the scheme respectively improves the network throughput, average RTT, average maximum queue size in the network and number of retransmissions by up to 41%, 37%, 60% and 98%. DTPA is applicable to any network with extremely small values of BDP. It can either be developed as a totally novel transport protocol or be extended as an option of TCP with the deployment consideration.

A substantial part of this thesis lies in the development of analytical models for TCP or DTPA in a multi-hop wireless network. In Chapter 3, the time taken to transmit packets in a system is divided into two components: the time taken to transmit packets and the time taken to count down the backoff timer which is uniformly distributed in (0, contention

window]. The probability theory is then used as a tool to analyze how contention window sizes evolve at each node. Based on this, the two types of times are computed and hence the throughput is derived. In Chapter 4 and Chapter 5, the transport protocol is modeled as a renewal process through which the throughput is derived as the mean number of packets transmitted within a renewal cycle. In the analysis of TCP, the loss window is modeled as a single dimensional semi-Markov chain. Various network scenarios and parameters are involved and analyzed through solving the transition probability matrix of the Markov chain, which is very complicated in computation. Different from this, the analysis of DTPA uses the probability theory and is much simpler due to the simplicity of the protocol algorithm. All these models are solved numerically through MATLAB codes.

In the thesis, all analytical results and proposals are verified and validated using simulator GloMoSim. Further study in the research area of reliable transport protocols, MAC and routing protocols over MANETs are quite promising. Several possible extensions of our research are described in the following section.

6.2 Future work

On the study of MAC layer medium contention caused packet losses, it should be noted that our proposed HELLO scheme as well as other researchers' proposals are unable to completely eliminate all false route breakages due to RTS transmission failures. In the HELLO scheme, if the number of RTS received by a node exceeds the threshold, the node is ready to transmit a HELLO message. However, the channel may be busy for a long period of time, this delaying the receipt of the HELLO message by the sender. Hence, the sender will wrongly assume that the route is broken. This phenomenon can appear frequently in a network where a packet transmission time is long due to a large packet size. For example, in Fig. 3.6, node *2* will not be able to transmit a HELLO message to node *3* during the long transmission time

duration of node *0*. There are several ways to mitigate this problem. First, the value of the threshold can be adaptive based on the various packet sizes in the network. A small value of threshold is expected for a network with a large packet size. Secondly, for a network with a large packet size, the scheme can be modified to allow the routing layer to delay for a short period of time before transmitting a route error message, so that the sender has a higher possibility of receiving the HELLO message. Both this short period of time and the values of the threshold need further investigation. In addition, the RTS or HELLO control messages may be corrupted due to the wireless channel error or due to several simultaneous RTS or HELLO message transmission attempts so that the corresponding nodes cannot receive the RTS or HELLO messages successfully. In order to achieve better performance, our scheme needs to be utilized together with some error correction schemes or the schemes developed in [5,39–49]. As a result, the further development of schemes to eliminate false route breakages could be pursued.

In Chapter 4, the DSR routing protocol is used instead of other ad hoc routing protocols in the model, as it facilitates the mathematical calculation of the time spent on the route failure notification and route re-establishment processes. In the analysis, we simplify the calculations of DSR. For example, we assume that during the transmission of packets within one window, l lost packets invoke only l route error messages to be transmitted by the intermediate node, and to recover the route, only one route discovery message and one route reply message are sent by the source and the destination respectively. This is not entirely accurate and this simplification may partly account for the deviation between the simulation results and the analytical results in our thesis. As part of our future work, we will include more accurate analysis of various routing protocols including DSR and AODV in our TCP model. We believe that it is worthwhile to study the route failure notification time and route re-establishment time for various routing protocols. The time taken to recover a route between the source and the destination is a random variable that is dependent on the mobility, traffic pattern

and network topology, etc. Once the time has been modelled accurately, it can be easily incorporated into our TCP model, i.e., by modifying the expression of $T_r(l)$ in (4.13), so that our model is applicable to mobile networks.

Furthermore, our work in Chapter 4 is based on the standard Reno flavor of TCP as well as TCP NewReno which is one of the most popular implementations in the Internet today. The similar approach of our model can be applied to other versions of TCP. Besides TCP NewReno, TCP SACK has been currently widely employed in Internet. In [66], the authors found that the fraction of tested SACK-capable web servers in Internet has increased from 41% in 2001 to 68% in 2004. Therefore it is useful to extend our model to involve the version of TCP SACK. In TCP SACK, a SACK option is added to the header of ACK packets. The option carries several blocks of information which contain contiguous sequence space occupied by data that has been received. Hence, one ACK can inform the source of more than one lost packets. Similarly, the analysis of SACK behaviors differs from Reno and NewReno only in the fast-recovery process. There is a different distribution of fast-retransmit and timeout probabilities. As part of our future work, we will analyze the fast-recovery process of SACK.

On the analytical study of TCP performance in the thesis, we always adopt a static string topology. Mobility, more complex topology and interactions between multiple TCP flows are not taken into account. One of the main reasons is that we believe that the TCP protocol itself is complex enough and it is necessary to first understand how the set of protocol stacks including TCP, IP and MAC behave in this baseline scenario, before exploring the impact of additional variables. In addition, although the model considers the simplest scenario of ad hoc networks, the analysis and results in our paper are very useful. In the future, a more realistic and accurate model is expected to be developed to involve more complex situations.

Bibliography

[1] W. Stevens, TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms, *Internet IETF RFC2001*, 1997.

[2] M. Allman, V. Paxson, and W. Stevens, TCP congestion control, *Internet IETF RFC2581*, 1999.

[3] V. Jacobson, Modified TCP congestion avoidance algorithm, message to end2end-interest mailing list, Apr. 1990, available at ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt.

[4] S.Giordano, A.Lombardo, M.Meo, and G.Schembra, Overview of the Research Results on Source Modeling, MQOS Workshop, http://www1.tlc.polito.it/mqos/sorgenti.html, 2000.

[5] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, The impact of multihop wireless channel on TCP throughput and loss, *In Proc. of IEEE INFOCOM*, vol. 3, pp. 1744–1753, 2003.

[6] ANSI/IEEE std 802.11, 1999 Edition: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specificaitons.

[7] J. Li, C. Blake, D. S. J. Couto, H. I. Lee, and R. Morris, Capacity of ad hoc wireless networks, *in Proc. of MobiCom*, pp. 61-69, 2001.

[8] K. Chen, Y. Xue, S. H. Shah, and K. Nahrstedt, Understand bandwidth-delay product

in mobile ad hoc networks, *Elsevier Computer Communications (ComCom) Journal, Special Issue on Protocol Engineering for Wired and Wireless Networks*, vol.27, no. 10, pp. 923–934, 2004.

[9] 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.

[10] S. Xu, and T. Saadwi, Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks?, *IEEE Communications Magazine*, vol. 39, no. 6, pp. 130-137, 2001.

[11] K. Xu, M. Gerla, and S. Bae, How effective is the IEEE 802.11 RTS/CTS handshake in ad hoc networks? *in Proc. of IEEE GLOBECOM*, Vol. 1, pp. 72–76, 2002.

[12] R. Oliveira, and T. Braun, TCP in wireless mobile ad hoc networks, *Technical Report, No. IAM–02–003*, University of Berne, Switzerland, July 2002.

[13] A. Al Hanbali, E. Altman, and P. Nain, A survey of TCP over mobile ad hoc networks, *INRIA Research Report, No. RR–5182*, May 2004.

[14] F. Wang, and Y. Zhang, A survey on TCP over mobile ad-hoc networks, *in Y. Xiao and Y. Pan (Ed.) Ad Hoc and Sensor Networks*, Nova Science Publishers, 2005.

[15] X. Chen, H. Zhai, J. Wang, and Y. Fang, TCP performance over mobile ad hoc networks, *Canadian Journal of Electrical and Computer Engineering (CJECE) (Special Issue on Advances in Wireless Communications and Networking)*, Vol. 29, No. 1/2, pp.129–134, January/April 2004.

[16] X. Chen, H. Zhai, J. Wang, and Y. Fang, A survey on improving TCP performance over wireless networks, *Resource Management in Wireless Networking (Springer Network*

Theory and Applications Series, Vol. 16), edited by M. Cardei, I. Cardei and D.-Z. Du, pp. 657–695, Kluwer Academic Publishers/Springer, 2005.

[17] S. Medidi, J. Ding, and M. Medidi, Performance of transport protocols in wireless networks, *Journal of Annual Reviews of Communication*, vol. 59, 2006.

[18] H. Singh, and S. Singh, Energy consumption of TCP Reno, NewReno, and SACK in multi-hop wireless networks, *in Proc. of ACM SIGMETRICS*, pp. 206–216, 2002.

[19] I. Ali, R. Gupta, S. Bansal, A. Misra, A. Razdan, and R. Shorey, Energy efficiency and throughput for TCP traffic in multi-hop wireless networks, *in Proc. of IEEE INFOCOM*, pp. 210–219, 2002.

[20] L. Yang, W. Seah, and Q. Yin, Improving fairness among TCP flows crossing wireless ad hoc and wired networks, *in Proc. of ACM MOBIHOC*, pp. 57–63, Jun 2003.

[21] K. Tang, and M. Gerla, Fair sharing of MAC under TCP in wireless Ad Hoc networks, *in Proc. of IEEE Multiclass Mobility and Teletraffic for Wireless Mobile Communications Workshop*, 1999.

[22] K. Xu, S. Bae, S. Lee, and M. Gerla, TCP behavior across multihop wireless networks and the wired internet, *in Proc. of the ACM Workshop on Wireless Mobile Multimedia*, pp. 41–48, 2002.

[23] K. Xu, M. Gerla, L. Qi, and Y. Shu, Enhancing TCP fairness in ad hoc wireless networks using neighborhood red, *in Proc. of ACM MOBICOM*, pp. 16–28, 2003.

[24] G. Anastasi, M. Conti, and E. Gregori, IEEE 802.11 ad hoc networks: performance measurements, *in Proc. of MWM in conjunction with ICDCS*, pp. 758–763, May 2003.

[25] G. Holland, and N. Vaidya, Analysis of TCP performance over mobile ad hoc networks, *ACM Wireless Networks*, vol. 8, no. 2, pp. 275–288, 2002.

[26] G. Holland, and N. Vaidya, Impact of routing and link layers on TCP performance in mobile ad hoc networks, *in Proc. of IEEE WCNC*, vol. 3, pp. 1323–1327, 1999.

[27] Z. Fu, X. Meng, and S. Lu, How bad TCP can perform in mobile ad hoc networks, *in Proc. of ISCC*, pp. 298–303, 2002.

[28] J. Monks, P. Sinha, and V. Bharghavan, Limitations of TCP-ELFN for ad hoc networks, *in Proc. of MoMuC*, 2000.

[29] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, A feedback based scheme for improving TCP performance in ad-hoc wireless networks, *In Proc. of ICDCS*, vol. 8, no. 2, pp. 34–39, 1998.

[30] J. Liu, and S. Singh, ATCP: TCP for mobile ad hoc networks, *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 7, pp. 1300–1315, 2001.

[31] D. Kim, C. Toh, and Y. Choi, TCP-Bus: Improving TCP performance in wireless ad hoc networks, *Journal of Communications and Networks*, vol. 3, no. 2, pp. 175–186, 2001.

[32] T. Dyer, and R. Boppana, A comparison of TCP performance over three routing protocols for mobile ad hoc networks, *in Proc. of ACM MobiHoc*, pp. 56–66, 2001.

[33] F. Wang, and Y. Zhang, Improving TCP performance over mobile ad hoc networks with out-of-order detection and response, *in Proc. of ACM MobiHoc*, Lausanne, Switzerland, pp. 217–225, Jun. 2002.

[34] Z. Fu, B. Greenstein, X. Meng, and S. Lu, Design and implementation of a TCP-friendly transport protocol for ad hoc networks, *In Proc. of IEEE ICNP*, pp. 216–225, 2002.

[35] S. Kopparty, S. Krishnamurthy, M. Faloutous, and S. Tripathi, Split TCP for mobile ad hoc networks, *in Proc. of IEEE GLOBECOM*, vol.1, pp. 138–142, 2002.

[36] M. Gerla, K. Tang, and R. Bagrodia, TCP performance in wireless multi-hop networks, *in Proc. of IEEE WMCSA*, pp. 41–50, 1999.

[37] M. Gerla, R. Bagrodia, L. Zhang, K. Tang, and L. Wang, TCP over wireless multi-hop protocols: simulation and experiments, *in Proc. of IEEE ICC*, vol. 2, pp. 1089–1094, 1999.

[38] C. Gomez, X. Marchador, V. Gonzalez, and J. Paradells, Multilayer analysis of the influence of mobility models on TCP flows in AODV ad-hoc networks, *in Proc. of LANMAN*, pp. 6, 2005.

[39] S. Desilva, and R. V. Boppana, On the impact of noise sensitivity on performance in 802.11 based ad hoc networks, *in Proc. of IEEE ICC*, vol. 7, pp. 4372–4376, 2004.

[40] Q. Chen, and Z. Niu, A delayed adaptive retransmission scheme for false route failure in MANET, *in Proc. of Asia–Pacific Conference on Communications*, vol. 2, pp. 858–862, 2004.

[41] K. Chen, K. Nahrstedt, and N. Vaidya, The utility of explicit rate-based flow control in mobile ad hoc networks, *in Proc. of IEEE WCNC*, vol. 3, pp. 1904–1909, 2004

[42] H. Zhai, X. Chen, and Y. Fang, Rate-based transport control for mobile ad hoc networks, *in Proc. of IEEE WCNC*, vol. 4, pp. 2264–2269, 2005.

[43] K. Sundaresan, V. Anantharaman, H.-Y. Hsieh, and R. Sivakumar, ATP: a reliable transport protocol for ad-hoc networks,*IEEE Transactions on Mobile Computing*, vol. 4, no. 6, pp. 588–603, 2005.

[44] S. M. ElRakabawy, A. Klemm, and C. Lindemann, TCP with adaptive pacing for multihop wireless networks, *in Proc. of ACM MobiHoc*, pp. 288–299, 2005.

[45] K. Chen, Y. Xue, and K. Nahrstedt, On setting TCP's congestion window limit in mobile ad hoc networks, *in Proc. of IEEE ICC*, vol. 2, pp. 1080–1084, 2003.

[46] S. Desilva, and R. V. Boppana, Sustaining performance under traffic overload, *in Proc. of International Conference on Wireless Networks*, pp. 3–8, 2004.

[47] A. Singh, and K. Kankipati, TCP-ADA: TCP with adaptive delayed acknowledgement for mobile ad hoc networks, *in Proc. of IEEE WCNC*, no. 1, pp. 1679–1684, 2004.

[48] E. Altman, and T. Jimenez, Novel delayed ACK techniques for improving TCP performance in multihop wireless networks, *in Proc. of Personal Wireless Communications*, pp. 237–253, 2003.

[49] R. Oliveira, and T. Braun, A dynamic adaptive acknowledgment strategy for TCP over multihop wireless networks, *In Proc. of IEEE INFOCOM*, pp. 1863–1874, 2005.

[50] C. Kai, Y. Chen, and N. Yu, An improvement scheme applied to TCP protocol in mobile ad hoc networks, *in Proc. of Mobile Technology, Applications and Systems*, pp. 1-6, 2005.

[51] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, The macroscopic behavior of the TCP congestion avoidance algorithm, *ACM Computer Communication Review*, vol. 27, no. 3, pp. 67–82, July 1997.

[52] T. Lakshman, and U. Madhow, The performance of TCP/IP for networks with high bandwidth-delay products and random loss, *IEEE/ACM Transaction on Networking*, vol. 5, no. 3, pp. 336–350, 1997.

[53] A. Kumar, Comparative performance analysis of versions of TCP in a local network with a lossy link, *IEEE/ACM Transaction on Networking*, vol. 6, pp. 485–498, 1998.

[54] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, Modeling TCP Reno performance: a simple model and its empirical validation, *ACM Computer Communication Review*, vol. 28, pp. 303–314, 1998.

[55] A. Kumar, and J. Holtzman, Comparative performance analysis of versions of TCP in a local network with a mobile radio link, *Sadhana: Indian Academy of Sciences proceedings in Engineering Sciences*, Vol. 23, pp. 113–129, 1998.

[56] M. Zorzi, and R. Rao, Effect of correlated errors on TCP, *in Proc. of CISS*, pp. 666–671, 1997.

[57] E. Altman, K. Avrachenkov, and C. Barakat. A stochastic model of TCP/IP with stationary random losses, *ACM SIGCOMM Computer Communication Review*, vol. 30, pp. 231–242, 2000.

[58] A. A. Abouzeid, S. Roy, and M. Azizoglu, Stochastic modeling of TCP over lossy links, *in Proc. of IEEE INFOCOM*, pp. 1724–1733, 2000.

[59] V. Misra, W. Gong, and D. Towsley, Stochastic differential equation modeling and analysis of TCP–windowsize behavior, *in Proc. of Performance*, 1999.

[60] B. Kim, and J. Lee, A simple model for TCP loss recovery performance over wireless networks, *Journal of Communications and Networks*, vol.6, no. 3, pp. 235–244, 2004.

[61] A. A. Kherani, and R. Shorey, Throughput analysis of TCP in multi-hop wireless networks with IEEE 802.11 MAC, *in Proc. of IEEE WCNC*, vol. 1, pp. 237–242, 2004.

[62] The dynamic source routing protocol for mobile ad hoc networks (DSR), *IETF draft-ietf-manet-dsr-10.txt*, 2004.

[63] Ad hoc On-Demand Distance Vector (AODV) Routing, *Internet IETF RFC3561*, 2003.

[64] S. Floyd, and T. Henderson, The NewReno modification to TCP's fast recovery algorithm, *Internet IETF RFC2582*, 1999.

[65] S. Floyd, T. Henderson, and A. Gurtov, The NewReno modification to TCP's fast recovery algorithm, *Internet IETF RFC3782*, 2004.

[66] A. Medina, M. Allman, and S. Floyd, Measuring the evolution of transport protocols in the Internet, *Computer Communication Review*, Vol. 35, Issue 2, pp. 37–52, 2005.

[67] Scalable Mobile Network Simulator, http://pcl.cs.ucla.edu/projects/glomosim/

[68] C. Kent, and J. Mogul, Fragmentation considered harmful, *in Proc. of SIGCOMM*, vol. 17, No. 5, 1987.

[69] 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.

[70] Y. Xu, Y. Wang, J. Lui, and D. Chiu, Balancing Throughput and Fairness for TCP Flows in Multihop Ad–Hoc Networks, *in Proc. of IEEE WiOpt*, 2007.

[71] X. Li, P.Y. Kong, and K.C. Chua, Analysis of TCP Throughput in IEEE 802.11 based Multi-hop Ad Hoc Networks, *in Proc. of IEEE ICCCN*, pp. 297–302, 2005.

[72] X. Li, K.C. Chua, and P.Y. Kong, The Study of False Route Breakage in IEEE 802.11 based Ad Hoc Networks, *in Proc. of IEEE MASS*, 2006.

[73] X. Li, P.Y. Kong, and K.C. Chua, TCP Performance in IEEE 802.11 based Ad Hoc Networks with Multiple Wireless Lossy Links, To be published in *IEEE Transaction on Mobile Computing*, 2007.

[74] X. Li, K.C. Chua, P.Y. Kong, and S.M. Jiang, The Impact of Lossy Links on TCP performance in IEEE 802.11 based Ad Hoc Networks, *in Proc. of IEEE WCNC*, vol. 3, pp. 1545–1550, 2005.

[75] X. Li, P.Y. Kong, and K.C. Chua, DTPA: A Reliable Datagram Transport Protocol over Ad Hoc Networks, Submitted for the 2nd review, *IEEE Transaction on Mobile Computing*, September, 2006.

Author's Publications

[1] X. Li, P.Y. Kong, and K.C. Chua, TCP Performance in IEEE 802.11 based Ad Hoc Networks with Multiple Wireless Lossy Links, To be published in *IEEE Transaction on Mobile Computing*, 2007.

[2] X. Li, P.Y. Kong, and K.C. Chua, DTPA: A Reliable Datagram Transport Protocol over Ad Hoc Networks, Submitted for review, *IEEE Transaction on Mobile Computing*, September, 2006.

[3] X. Li, P.Y. Kong, and K.C. Chua, Finding an Optimum Maximum Congestion Window for TCP Reno over 802.11 based Ad Hoc Networks, *in Proc. of IEEE WCNC*, 2007.

[4] X. Li, K.C. Chua, and P.Y. Kong, The Study of False Route Breakage in IEEE 802.11 based Ad Hoc Networks, *in Proc. of IEEE MASS*, pp. 493-496, 2006.

[5] X. Li, P.Y. Kong, and K.C. Chua, Analysis of TCP Throughput in IEEE 802.11 based Multi-hop Ad Hoc Networks, *in Proc. of IEEE ICCCN*, pp. 297–302, 2005.

[6] X. Li, K.C. Chua, P.Y. Kong, and S.M. Jiang, The Impact of Lossy Links on TCP performance in IEEE 802.11 based Ad Hoc Networks, *in Proc. of IEEE WCNC*, vol. 3, pp. 1545–1550, 2005.

Appendix: Fast-Recovery Analysis

In this section, we adopt accurate and realistic analysis to fast-recovery process for TCP Reno and TCP NewReno. Given a loss window i and the number of detected lost packets l via the receipt of ACKs, we have obtained accurate formulae for the probabilities $\bar{\gamma}_{il}$ and γ_{il} which are used in the main text in Chapter 4. To facilitate the analysis, we define each round during the fast-recovery process starting from a retransmission. As shown in Fig. 4.1, the start of round k is denoted by R_k. Let Φ_k denote the number of packets successfully transmitted in the k-th round of fast-recovery period. It is obvious that only one lost packet can be detected and retransmitted within a round.

I. Reno

The condition that round $R_k (k > 1)$ can start only if TCP Reno can receive 3 duplicate ACKs and the retransmission within round R_{k-1} is successful. Given that there are L lost packets in the loss window i, Φ_1 is always equal to $i - L$. For $k \geq 2$, Φ_k comprises three parts: one retransmission, new transmissions by temporarily inflating the window and new transmissions triggered by the successful delivery of the retransmission. We denote them as $\Phi_{k,1}$, $\Phi_{k,2}$ and $\Phi_{k,3}$ respectively and they meet $\Phi_k = \Phi_{k,1} + \Phi_{k,2} + \Phi_{k,3}$. Obviously, $\Phi_{k,1} = 1$.

A. Calculate Φ_2

With $\Phi_1 \geq K$, where K represents three duplicate ACKs, during the second round, the congestion window is cut in half to $\lfloor i/2 \rfloor$ when the first packet loss is detected by the receipt

of three duplicate ACKs, and all the duplicate ACKs for the first retransmitted packet inflates the window by Φ_1. Since the TCP source continuously receives the duplicate ACKs with the same sequence number requesting for the lost packet, it will still regard the number of packets outstanding in the pipe as i packets. Hence, the number of new packets transmitted due to window inflation is $\Phi_{2,2} = \lfloor \frac{i}{2} \rfloor + \Phi_1 - i = \lfloor \frac{i}{2} \rfloor - L$.

Let m_2 denote the m_2-th packet of the i packets, which is also the second lost packet in loss window i. Since the first lost packet is retransmitted successfully, the source will figure that $m_2 - 1$ packets have been acknowledged. Therefore, the source regards the number of outstanding packets as $i + \Phi_{2,2} - (m_2 - 1)$, where $2 \leq m_2 \leq i - L + 2$. Then, the number of packets allowed to be transmitted after receiving the ACK for the lost packet is $\Phi_{2,3} = \lfloor \frac{i}{2} \rfloor - (i + \Phi_{2,2} - (m_2 - 1)) = L + m_2 - i - 1$, where $L + m_2 - i - 1 > 0$ only if the last $i - m_2$ packets within the loss window are also lost. Under this condition, $\Phi_{2,3} = 1$ and 0 otherwise. Φ_2 is given by:

$$\Phi_2 = \begin{cases} \lfloor \frac{i}{2} \rfloor + 1 - L - a_2 & L < i - m_2 + 2 \\ \lfloor \frac{i}{2} \rfloor + 2 - L - a_2 & L = i - m_2 + 2, \end{cases} \tag{6.1}$$

where a_2 denotes the number of lost packets in the second round and meets $0 \leq a_2 \leq \Phi_2$.

B. Calculation Φ_3

If the retransmission is successful and $\Phi_2 \geq K$, there is the third round under the condition of $L > 1$. In the third round, the congestion window becomes $\lfloor i/4 \rfloor$.

The window inflation is Φ_2 due to the receipt of Φ_2 duplicate ACKs requesting for the second lost packet. The source regards the number of outstanding packets as $i + \Phi_2 + a_2 - 1 - (m_2 - 1)$. Then, due to the window inflation, the number of new packets allowed to be transmitted in the third round is $\Phi_{3,2} = \lfloor \frac{i}{4} \rfloor + \Phi_2 - (i + \Phi_2 + a_2 - 1 - (m_2 - 1)) = \lfloor \frac{i}{4} \rfloor - i + m_2 - a_2$.

With the successful retransmission of the second lost packet, the further number of new packets transmitted is dependent on the position of the third packet loss within loss window

i. Let m_3 denotes the m_3-th packet in i packets which is also the third lost packet in loss window i. Then, the number of packets allowed to be transmitted after receiving the ACK for the lost packet is $\Phi_{3,3} = \lfloor \frac{i}{4} \rfloor - (i + \Phi_2 + a_2 - 1 + \Phi_{3,2} - (m_3 - 1)) = m_3 - m_2 - \Phi_2$. However, it is calculated that $\Phi_{3,3}$ can never be greater than zero under any circumstances. Hence, Φ_3 is then given by:

$$\Phi_3 = \lfloor \frac{i}{4} \rfloor - i + m_2 - a_2 - a_3 + 1, \tag{6.2}$$

where a_3 is the number of lost packets in the third round.

C. Calculate Φ_4

Similarly, in the fourth round of the fast-recovery process, we can get: $\Phi_{4,1} = 1$, $\Phi_{4,2} = \lfloor \frac{i}{8} \rfloor - i - \Phi_2 + m_3 - a_2 - a_3 + 1$ and $\Phi_{4,3} = \lfloor \frac{i}{8} \rfloor - i - \Phi_2 - \Phi_3 - \Phi_{4,2} + m_4 - a_2 - a_3 + 1$, where m_4 is the m_4-th packet in i packets which is also the fourth lost packet in loss window i. It is calculated that $\Phi_{4,2}$ and $\Phi_{4,3}$ can never be greater than zero. Hence, there is only one packet transmitted in the fourth round.

From the derivation above, we know that $\Phi_k (1 \leq k \leq 3)$ can be greater than zero under certain conditions. It indicates that TCP can recover at most three packet losses within a loss window via Three-Duplicate-ACK events during fast-recovery and more than three losses in a window will always invoke a timeout. This is consistent with the result in [60].

Given a loss window i, the conditions of the probabilities $\bar{\gamma}_{il}$ and γ_{il} $(0 \leq l \leq L)$ are:

$$\bar{\gamma}_{il} = \begin{cases} \bar{\gamma}_{i1} & L = 1, \Phi_1 \geq K, \Phi_{2,1} \text{ success} \\ \bar{\gamma}_{i2} & L = 2, \Phi_2 > K, \Phi_{3,1} \text{ success} \\ \bar{\gamma}_{i3} & L = 3, \Phi_2 > K, \Phi_3 > K, \Phi_{4,1} \text{ success} \\ 0 & \text{elsewhere} \end{cases} \tag{6.3}$$

$$\gamma_{il} = \begin{cases} \gamma_{i0} & L \geq 1, 0 \leq \Phi_1 < K \\ \gamma_{i1} & \{L \geq 1, \Phi_1 \geq K, \Phi_{2,1} \text{ lost}\} \\ & \bigcup \{L \geq 2, \Phi_1 \geq K, 1 \leq \Phi_2 \leq K, \Phi_{2,1} \text{ success}\} \\ \gamma_{i2} & \{L \geq 2, \Phi_2 > K, \Phi_{3,1} \text{ lost}\} \\ & \bigcup \{L \geq 3, \Phi_2 > K, 1 \leq \Phi_3 \leq K, \Phi_{3,1} \text{ success}\} \\ \gamma_{i3} & \{L \geq 3, \Phi_2 > K, \Phi_3 > K, \Phi_{4,1} \text{ lost}\} \\ & \bigcup \{L \geq 4, \Phi_2 > K, \Phi_3 > K, \Phi_{4,1} \text{ success}\} \\ 0 & \text{elsewhere,} \end{cases} \tag{6.4}$$

Base on these conditions, we can list the equations of $\bar{\gamma}_{il}$ and γ_{il} as follows:

$$\bar{\gamma}_{il} = \begin{cases} \bar{p}^{\Phi_1+1} & l = 1 \\ \sum_{a_2} \binom{\Phi_2+a_2-1}{a_2} (\Phi_1+1) p^{a_2+1} \bar{p}^{\Phi_1+\Phi_2} & l = 2 \\ \sum_{m_2} \sum_{a_3} \sum_{a_2} p(m_2) \binom{\Phi_3+a_3-1}{a_3} \binom{\Phi_2+a_2-1}{a_2} \binom{\Phi_1+2}{2} p^{a_2+a_3+2} \bar{p}^{\Phi_1+\Phi_2+\Phi_3} & l = 3 \\ 0 & \text{elsewhere,} \end{cases}$$

$$\gamma_{il} = \begin{cases} \sum_{L=i+1-K}^{i} \binom{\Phi_1+L-1}{L-1} p^{L-1} \bar{p}^{\Phi_1} & l = 0 \\ \sum_L \binom{\Phi_1+L-1}{L-1} p^L \bar{p}^{\Phi_1} + \sum_L \sum_{a_2} \binom{\Phi_2+a_2-1}{a_2} \binom{\Phi_1+L-1}{L-1} p^{L+a_2-1} \bar{p}^{\Phi_1+\Phi_2} & l = 1 \\ \sum_L \sum_{a_2} \binom{\Phi_2+a_2-1}{a_2} \binom{\Phi_1+L-1}{L-1} p^{L+a_2} \bar{p}^{\Phi_1+\Phi_2} & \\ + \sum_L \sum_{m_2} \sum_{a_3} \sum_{a_2} p(m_2) \binom{\Phi_3+a_3-1}{a_3} \binom{\Phi_2+a_2-1}{a_2} \binom{\Phi_1+L-1}{2} p^{L+a_2+a_3-1} \bar{p}^{\Phi_1+\Phi_2+\Phi_3} & l = 2 \\ \sum_L \sum_{m_2} \sum_{a_3} \sum_{a_2} p(m_2) \binom{\Phi_3+a_3-1}{a_3} \binom{\Phi_2+a_2-1}{a_2} \binom{\Phi_1+L-1}{2} p^{L+a_2+a_3-1} \bar{p}^{\Phi_1+\Phi_2+\Phi_3} & \\ + \sum_L \sum_{m_2} \sum_{a_3} \sum_{a_2} p(m_2) \binom{\Phi_3+a_3-1}{a_3} \binom{\Phi_2+a_2-1}{a_2} \binom{\Phi_1+L-1}{2} p^{L+a_2+a_3} \bar{p}^{\Phi_1+\Phi_2+\Phi_3} & l = 3 \\ 0 & \text{elsewhere,} \end{cases}$$

where $P(m_2)$ is the probability density for m_2 and is given by:

$$P(m_2) = \frac{\binom{i-m_2}{L-2}}{\binom{i-1}{L-1}}.$$

II. NewReno

NewReno differs from Reno in that only the first packet loss within the loss window i is detected by the receipt of three duplicate ACKs. The detection and recovery of the sequent lost packets is independent of new packets transmitted within the fast-recovery process. The lost packets can be retransmitted only when the TCP source receives a partial ACK which is triggered by the successful retransmission of the previously lost packet. Obviously, if all retransmissions are successful, the number of rounds during fast-recovery reaches its maximum value of $L + 1$ and the congestion avoidance phase starts with congestion window $\lfloor \frac{i}{2} \rfloor$. Otherwise, a timeout is invoked immediately when a retransmitted packet is lost, and the subsequent lost packets within the loss window i cannot be detected and recovered. The number of rounds within the fast-recovery process depends on the number of detected packet losses l and is given by $l + 1$.

We first analyze the fast-recovery process for the Slow-but-Steady variant of NewReno, which is the basis for further study of Impatient variant of NewReno. We use the same parameter definitions as those in Reno. Obviously, $\Phi_1 = i - L$ and $\Phi_{k,1} = 1$ for $k > 1$. In the second round, $\Phi_{2,2}$ is the same as that in the derivation of Reno. For $\Phi_{2,3}$, which differs from Reno, the NewReno source retransmits a packet as soon as it receives an ACK for any new packet. Hence, $\Phi_{2,3} = 0$. As a result, we have:

$$\Phi_2 = \lfloor \frac{i}{2} \rfloor - L + 1 - a_2. \tag{6.5}$$

In the third round, the congestion window is inflated to $\lfloor \frac{i}{2} \rfloor + i - L$ before the partial ACK for the first retransmission arrives. As soon as the source receives the partial ACK, the second lost packet within window i is transmitted immediately. The source deflates the congestion window by the amount of new data acknowledged by the cumulative acknowledgement field. If the partial ACK acknowledges at least one new data packet, the source then increments the congestion window by one packet. Hence, the congestion window becomes $\lfloor \frac{i}{2} \rfloor + i - L - m_2 + 2$. Since the source regards the number of outstanding packet as $w + \Phi_2 + a_2 - 1 - (m_2 - 1)$, the new transmission $\Phi_{3,2}$ is one packet with the permission of the new congestion window size. Similarly, we derive $\Phi_{3,3} = \Phi_2 - 1$. Hence, we have:

$$\Phi_3 = \lfloor \frac{i}{2} \rfloor - L + 2 - a_2 - a_3. \tag{6.6}$$

Following this logic, we derive and obtain:

$$\Phi_k = \lfloor \frac{i}{2} \rfloor - L + k - 1 - a_2 - a_3 - \cdots - a_k \qquad k > 1. \tag{6.7}$$

For the Slow-but-Steay variant, given a loss window i, the conditions of the probabilities $\bar{\gamma}_{il}$ and γ_{il} $(0 \le l \le L)$ are:

$$\bar{\gamma}_{il} = \bar{\gamma}_{iL}, \qquad \Phi_1 \ge K, \Phi_{L1} \text{ success}, \tag{6.8}$$

$$\gamma_{il} = \begin{cases} \gamma_{i0} & L \ge 1, 0 \le \Phi_1 < K \\ \gamma_{il} & L \ge 1, 0 < l \le L, \Phi_1 \ge K, \Phi_{l1} \text{ lost} \end{cases} \tag{6.9}$$

The equations of $\bar{\gamma}_{il}$ and γ_{il} $(0 \leq l \leq L)$ are given by:

$$\bar{\gamma}_{il} = \begin{cases} \binom{i-1}{l-1} p^{l-1} \bar{p}^{i} & l = L \\ 0 & \text{elsewhere,} \end{cases} \tag{6.10}$$

$$\gamma_{il} = \begin{cases} \sum_{L=i+1-K}^{i} \binom{i-1}{L-1} p^{L-1} \bar{p}^{i-L} & l = 0 \\ \sum_{L=l}^{i-K} \binom{i-1}{L-1} p^{L} \bar{p}^{i-L+l-1} & 1 \leq l \leq L \\ 0 & \text{elsewhere.} \end{cases} \tag{6.11}$$

Contrary to the Slow-but-Steady variant which resets the retransmit timer each time it receives a partial ACK, the Impatient variant resets the retransmit timer only after the first partial ACK. As a result, the retransmit timer expires and slow-start is invoked when a large number of packets have been dropped. If a full ACK is received before the timeout, the Impatient variant and the Slow-but-Steady variant behave identically. Hence, we define a threshold value c for the maximum number of lost packets which can be recovered before timeout occurs. For $L \leq c$, the Impatient variant has the same equations of $\bar{\gamma}_{il}$ and γ_{il} as the Slow-but-Steady variant. The two variants differ in the case of $L > c$. Hence, the equations of $\bar{\gamma}_{il}$ and γ_{il} $(0 \leq l \leq L)$ for Impatient NewReno are given by:

$$\bar{\gamma}_{il} = \begin{cases} \binom{i-1}{l-1} p^{l-1} \bar{p}^{i} & l = L \leq c \\ 0 & \text{elsewhere,} \end{cases} \tag{6.12}$$

$$\gamma_{il} = \begin{cases} \sum_{L=i+1-K}^{i} \binom{i-1}{L-1} p^{L-1} \bar{p}^{i-L} & l = 0 \\ \sum_{L=l}^{\min(i-K,c)} \binom{i-1}{L-1} p^{L} \bar{p}^{i-L+l-1} & 1 \leq l \leq L \leq c \\ \sum_{L=c+1}^{i-K} \binom{i-1}{L-1} p^{L} \bar{p}^{i-L+l-1} & 1 \leq l < c < L \\ \sum_{L=c+1}^{i-K} \binom{i-1}{L-1} p^{L-1} \bar{p}^{i-L+l-1} & 1 \leq l = c < L \\ 0 & \text{elsewhere.} \end{cases} \tag{6.13}$$

To derive c, let T_c denote the time to transmit $\sum_{j=3}^{c} \Phi_j$ packets. Using the same approach to calculate $E(Y)$ and $E(Z)$ in the main text in Chapter 4, the total time T_c to transmit these amount of packets can be derived. Hence, we can find the minimum value c which satisfies $T_c \geq RTO$.