TCP SINTOK: TRANSMISSION CONTROL PROTOCOL WITH DELAY-BASED LOSS DETECTION AND CONTENTION AVOIDANCE MECHANISMS FOR MOBILE AD HOC NETWORKS

ADIB HABBAL

DOCTOR OF PHILOSOPHY
UNIVERSITI UTARA MALAYSIA
2014

Permission to Use

In presenting this thesis in fulfilment of the requirements for a postgraduate degree from Universiti Utara Malaysia, I agree that the Universiti Library may make it freely available for inspection. I further agree that permission for the copying of this thesis in any manner, in whole or in part, for scholarly purpose may be granted by my supervisor(s) or, in their absence, by the Dean of Awang Had Salleh Graduate School of Arts and Sciences. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to Universiti Utara Malaysia for any scholarly use which may be made of any material from my thesis.

Requests for permission to copy or to make other use of materials in this thesis, in whole or in part, should be addressed to:

Dean of Awang Had Salleh Graduate School of Arts and Sciences
UUM College of Arts and Sciences
Universiti Utara Malaysia
06010 UUM Sintok

Abstrak

Rangkaian *Ad hoc* Mudah Alih (MANET) terdiri daripada peranti mudah alih yang bersambung antara satu sama lain dengan menggunakan saluran tanpa wayar. Ia membentuk satu rangkaian sementara tanpa bantuan infrastruktur tetap; yang mana hos adalah bebas untuk bergerak secara rawak dan juga bebas untuk menyertai atau meninggalkan sesuatu rangkaian. Sifat berpusat MANET tampil dengan cabaran baru yang melanggar konsep reka bentuk Protokol Kawalan Penghantaran (TCP); sejenis protokol dominan untuk Internet pada masa kini. TCP sentiasa merumuskan kehilangan paket sebagai petunjuk kesesakan rangkaian dan menyebabkan ia melaksanakan pengurangan mendadak kepada kadar penghantaran data. MANET mengalami beberapa jenis kehilangan paket disebabkan oleh ciri mobiliti dan pertelagahan capaian saluran tanpa wayar dan ini akan melemahkan prestasi TCP. Oleh itu, kajian eksperimental ini menyiasat satu protokol yang dikenali sebagai TCP Sintok. Protokol ini mempuntai dua mekanisme: Mekanisme Pengesanan Kehilangan berasaskan kelewatan (LDM) dan Mekanisme Pengelakan Perebutan (CAM). LDM diperkenalkan untuk menentukan punca kehilangan paket dengan memantau trend sampel kelewatan hujung-ke-akhir. CAM telah dibangunkan untuk penyesuaian pada kadar penghantaran (tetingkap kesesakan) mengikut keadaan rangkaian semasa. Kajian eksperimen telah dijalankan bagi mengesahkan keberkesanan TCP Sintok dalam mengenal pasti punca kehilangan paket dan menyesuaikan kadar penghantaran yang bersesuaian. Dua varian protokol TCP yang dikenali sebagai TCP NewReno dan ADTCP telah dipilih untuk menilai prestasi TCP Sintok melalui simulasi. Keputusan menunjukkan bahawa TCP Sintok memperbaiki ketaran, kelewatan dan daya pemprosesan berbanding dengan dua varian tersebut. Hasil penemuan penyelidikan ini mempunyai implikasi penting dalam menyediakan pemindahan data yang boleh dipercayai dalam MANET dan menyokong penempatan pada komunikasi peranti mudah alih.

Kata kunci: Protokol kawalan penghantaran, Rangkaian *ad hoc* mudah alih, Pengelakan pertelagahan, Teori penyesuaian komunikasi

Abstract

Mobile Ad hoc Network (MANET) consists of mobile devices that are connected to each other using a wireless channel, forming a temporary network without the aid of fixed infrastructure; in which hosts are free to move randomly as well as free to join or leave. This decentralized nature of MANET comes with new challenges that violate the design concepts of Transmission Control Protocol (TCP); the current dominant protocol of the Internet. TCP always infers packet loss as an indicator of network congestion and causes it to perform a sharp reduction to its sending rate. MANET suffers from several types of packet losses due to its mobility feature and contention on wireless channel access and these would lead to poor TCP performance. This experimental study investigates mobility and contention issues by proposing a protocol named TCP Sintok. This protocol comprises two mechanisms: Delay-based Loss Detection Mechanism (LDM), and Contention Avoidance Mechanism (CAM). LDM was introduced to determine the cause of the packet loss by monitoring the trend of end-to-end delay samples. CAM was developed to adapt the sending rate (congestion window) according to the current network condition. A series of experimental studies were conducted to validate the effectiveness of TCP Sintok in identifying the cause of packet loss and adapting the sending rate appropriately. Two variants of TCP protocol known as TCP NewReno and ADTCP were chosen to evaluate the performance of TCP Sintok through simulation. The results demonstrate that TCP Sintok improves jitter, delay and throughput as compared to the two variants. The findings have significant implication in providing reliable data transfer within MANET and supporting its deployment on mobile device communications.

Keywords: Transmission control protocol, Mobile ad hoc network, Contention avoidance, Communication accommodation theory

Declaration Associated with This Thesis

Some of the works presented in this thesis have been published or submitted as listed below.

[1] **Adib M. Monzer Habbal** and Suhaidi Hassan, “A Model for Congestion Control of Transmission Control Protocol in Mobile Wireless Ad hoc Networks ”, Journal of Computer Science (JCS), Vol. 9(3), pp. 468-473 (2013), ISSN: 1549-3636. [Citation indexed by SCOPUS]

[2] Suhaidi Hassan, **Adib M. Monzer Habbal**, Suki Arif " End-To-End Loss Discrimination Mechanism for TCP over MANET" in the Proceedings of LEADS SEMINAR 2013, Convention Centre, Universiti Utara Malaysia, 10-11 June 2013. [Chapter in Book]

[3] Suhaidi Hassan, **Adib M. Monzer Habbal**, “Modeling TCP Delay in IEEE 802.11 multi-hop Wireless Ad hoc Networks”, in the *Proceedings of the 3rd* European Conference for the Applied Mathematics and Informatics (AMATHI ’12), Montreux, Switzerland, 29-31 December 2012. ISBN 978-1-61804-148-7. [Citation indexed by SCOPUS]

[4] **Adib M. Monzer Habbal** and Suhaidi Hassan, “Contention Avoidance Mechanism for TCP in Mobile Ad hoc Network ”, in the *Proceedings of the 4th*Global Information Infrastructure and Networking Symposium (GIIS 2012), Choroni, Venezuela, 17-19 December 2012. ISBN 156-9-67736-1. [Citation indexed by SCOPUS]

[5] Khuzairi Mohd Zaini, **Adib M. Monzer Habbal**, Fazli Azzali, Suhaidi Hassan and Mohamad Rizal, “An Interaction between Congestion-Control Based Transport Protocols and MANET Routing Protocols”, Journal of Computer Science (JCS), Vol. 8(4), pp. 468-473 (2012), ISSN: 1549-3636. [Citation indexed by SCOPUS]

[6] Haniza N., Md Khambari, M. N, Shahrin S., **Adib M. Monzer Habbal** and Suhaidi Hassan, "Topology Influence on TCP Congestion Control Performance in Multi-hop Ad hoc Wireless", in the *Proceedings of the nternational Conference on Wireless Communication and Sensor Network (ICWCSN2012)*, United Arab Emirates, 29-31 January 2012. [Citation indexed by ISI/SCOPUS]

[7] **Adib M. Monzer Habbal** and Suhaidi Hassan, "Delay-Based Loss Discrimination Mechanism for Congestion Control in Wireless Ad-hoc Network", in the *Proceedings of the International Conference on Informatics Engineering & Information Science (ICIEIS2011),* Malaysia, on 14-16 November 2011. Communications in Computer and Information Science" (CCIS) Series of Springer LNCS. ISBN: 978-3-642-25461-1 [Citation indexed by ISI/SCOPUS]

[8] **Adib M. Monzer Habbal** and Suhaidi Hassan, "A Reference Model for TCP over MANET", in the *Proceedings of the 3rd IEEE International Conference on Computer Modeling ad Simulation (ICCMS2011)*, India, 7-9 January 2011. ISBN: 978-1-4244-9241-1. [Citation indexed by ISI]

[9] **Adib M. Monzer Habbal** and Suhaidi Hassan, "Loss Detection and Recovery Techniques for TCP in MANET", in the *Proceedings of the 2nd International Conference on Network Applications, Protocols and Services 2010 (NetApps2010)*, Malaysia, 22-23 September 2010. ISBN 978-1-4244-8048-7. [Citation indexed by SCOPUS]

Acknowledgements

In the name of ALLAH, Most Gracious, Most Merciful:

"*Work; so Allah will see your work and (so will) His Messenger and the believers*;"

(The Holy Quran - AtTawbah 9:105)

Conducting this research marks the end of an interesting and eventful journey. The completion of this thesis took a longer time than I expected to signify the fulfillment of a long-awaited goal. It could not have been achieved without the academic professional and personal support of the following wonderful and talented people.

I will start by extending my deep and sincere gratitude to my supervisor Professor Dr. Suhaidi Hassan (School of Computing, Universiti Utara Malaysia) for his tireless encouragement, wisdom and experience, who provided me with constant guidance and constructive criticism throughout all stages of my research. I must say a huge thank you to the current and past members of InterNetWorks Research Lab whom I enjoyed working with; especially, Assoc. Prof. Dr. Osman Ghazali, Dr. Ahmad Suki Che Mohamed Arif, Dr. Mohd. Hasbullah Omar, Dr. Yaser Miaji, Mr. Khuzairi Mohd Zaini, Dr. Massudi Mahmuddin, and others.

My grateful thanks are also extended to Dr. Peter W. Staecker (2013 IEEE President and CEO) for his encouragement and motivation, Prof. Dr. Srinivasan Keshav (Chair of ACM SIGCOMM) and Prof. Dr. Rahmat Budiarto (Surya University, Indonesia) for useful and insight discussion, and to Prof. Dr. Zulikha Jamaludin (UUM) and Prof. Dr. Giovanna Di Marzo Serugendo (University of Geneva, Switzerland) for their excellent and moral support during my research. Further, my truthful acknowledgement goes to Dr. Zhenghua Fu, (University of California, Los Angeles) for his advice on the implementation of TCP in NS-2 as well as making the source code of ADTCP available for research community; and to Mr. Sachin Gajjar, (Nirma University, India)

for sharing his experience in implementing I-ADTCP in NS-2 environment.

My deepest gratitude goes to network research community. In particular, I would like to thank the team of the Internet Society's Next Generation Leaders (NGL) program for their kind support and guidance given to me while serving as ISOC Fellow to the Internet Engineering Task Force (IETF) and especially to Steve Conte and my mentors: David Black, Mirja Kuehlwind, and Richard Scheffenegger for their scientific advice, knowledge and many insightful discussions. Further, special thanks to Asia Pacific Advanced Networking (APAN) fellowship committee for their guidance and good humor offered to me while participating as APAN fellow to the Techs in Paradise meeting. Also, I would like to thank those researchers who reviewed my papers and commented on my publications. Special thanks to Mr. Faisal Zulhumadi for editing this manuscript. Additionally, I would like to take this opportunity to present my gratitude to Universiti Utara Malaysia for their support and having trust in me to complete this study.

Finally, my heartiest gratitude goes to my family, to my late father M. Monzer, to my mother Faten who always has faith in me and prays for my success, to my brothers Amjad and Ayman, who are willing to extend a helping hand, to my beloved wife Rawaa for her understanding, support, and love, also deepest thanks to my daughter Faten for being so sweet and loving.

Dedication

For my family . . .

in memory of my father M. Monzer;

my mother Faten; and

my brothers Amjad, and Ayman

my wife Rawaa; and

our daughter Faten

Table of Contents

List of Tables

List of Figures

List of Appendices

List of Abbreviations

ABSE	-	Adaptive Bandwidth Share Estimation
ACK	-	Acknowledgement
ADSN	-	ACK Duplication Sequence Number
ADTCP	-	TCP-friendly Transport Protocol for Ad hoc Networks
AIMD	-	Additive Increase, Multiplicative Decrease
AODV	-	Ad Hoc On-Demand Distance Vector
ASP	-	Adaptive Packet Size
ASP-FeW	-	Adaptive Packet Size on Top of FeW
BEB	-	Binary Exponential Backoff
BER	-	Bit Error Rate
BDP	-	Bandwidth Delay Product
CAM	-	Contention Avoidance Mechanism
CAT	-	Communication Accommodation Theory
CR	-	Contention Ratio
CSMA/CA	-	Carrier Sensing Multiple Access with Collision Avoidance
CWA-CD	-	Congestion Window Adaptation through Contention Detection
CWL	-	Congestion Window Limit
CWND	-	Congestion Window
DACK	-	Delay ACKnoweledgment
DCF	-	Distributed Coordination Function
DIFS	-	Distributed Inter Frame Space
DRM	-	Design Research Methodology
DS-I	-	Descriptive Study-I
DS-II	-	Descriptive Study-II
DSR	-	Dynamic Source Routing
DUPACK	-	DUPlicate ACKnowledgements
ECN	-	Explicit Congestion Notification
ELFN	-	Explicit Link Fail Notification
ELU	-	Efficient Link Utilization
FEDM	-	Fuzzy-based Error Detection Mechanism

Few	-	Fractional Window increment
FIFO	-	First In First Out
FTP	-	File Transfer Protocol
GM	-	Gauss Markov
GUI	-	Graphical User Interface
HTTP	-	HyperText Transfer Protocol
IADTCP	-	Improved-ADTCP
IETF	-	Internet Engineering Task Force
IDD	-	Inter-packet Delay Difference
IW	-	Initial value of cwnd
LDA	-	Loss Differentiation Algorithm
LRA	-	Loss Recovery Algorithm
LW	-	Loss Window
LDM	-	Loss Detection Mechanism
LRL	-	Long Retry Limit
M-ADTCP	-	Modified AD-hoc Transmission Control Protocol
MAC	-	Media Access Control
MANET	-	Mobile Ad hoc NETwork
MME-TCP	-	Multi-metric Measurement based Enhancement of TCP
MATLAB	-	MATrix LABoratory
NS-2	-	Network Simulator ver2
OLSR	-	Optimized Link State Routing
OOO	-	Out-Of-Order
PCT	-	Pair wise Comparison Test
PDA	-	Personal Digital Assistant
PHY	-	PHYsical layer
PLR	-	Packet Loss Ratio
POR	-	Packet Out-of-order Arrival
RPGM	-	Reference Point Group Mobility
PS	-	Perspective Study
RC	-	Research Clarification
RFC	-	Request For Comments

RSD	-	Relative Sample Density
RTO	-	Retransmission Time Out
RTHC	-	Round-Trip Hop-Count
RTS/CTS	-	Request To Send / Clear To Send
RTT	-	Round Trip Time
RW	-	Random Waypoint
RWND	-	Receiver's Advertised Window
SACK	-	Selective ACKnoweledment
SANET	-	Static Ad hoc NETwork
SIFS	-	Short InterFrame Space
SMTP	-	Simple Mail Transfer Protocol
SMSS	-	Sender Maximum Segment Size
SRL	-	Short Retry Limit
SRTT	-	Smooth RTT
SSTHRESH	-	Slow Start THRESHold
STG	-	Short Term Goodput
STT	-	Short Term Throughput
TCP	-	Transmission Control Protocol
TCP/IP	-	Transmission Control Protocol/Internet Protocol
TCP ADA	-	TCP with Adaptive Delayed Acknowledgement
TCP AR	-	TCP Adaptive RTO
TCP-AP	-	TCP with Adaptive Pacing
TCPCC	-	TCP-Channel utilization and Contention Ratio
TCP DAA	-	Dynamic Adaptive Acknowledgement
TCP DCA	-	TCP Delayed Cumulative Ack
TCP DCR	-	TCP Delayed Congestion Response
TCP DOOR	-	TCP Detection of Out-of-Order and Response
TCP-MEDX	-	TCP-Mobile Error Detection eXtension
TCP-R	-	Protocol for Mobility-induced Packet Reordering
TCPW	-	TCP Westwood
TPSN	-	TCP Packet Sequence Number
PAT	-	Partition-Aware TCP

P2P	-	Peer-to-Peer
PDA	-	Personal Digital Assistant
PLR	-	Packet Loss Ratio
POR	-	Packet Out of order Delivery Ratio
Us	-	Sender's Utilization
Un	-	Neighbors' Utilization
VANET	-	Vehicular Ad hoc NETwork
VCRH	-	Variance of Contention RTT per Hop
WWW	-	World Wide Web
WLAN	-	Wireless Local Area Network

CHAPTER ONE
INTRODUCTION

The Internet success has contributed to the adaptation of the Transmission Control Protocol/Internet Protocol (TCP/IP) suite to build different types of communication networks including ad hoc network [2]. Transmission Control Protocol (TCP), the predominant transport protocol, is used in the TCP/IP stack to support the multitude of Internet services. This thesis presents a new Transmission Control Protocol, named TCP Sintok, and its verified performance in IEEE 802.11 ad hoc networks. This chapter aims to place the thesis of this work within its context, where the general background of the research is described briefly. This chapter begins with an introductory overview of TCP and ad hoc networks, followed by a brief description of the popular applications of ad hoc networks. Characteristics of mobile ad hoc networks are deliberated in Section 1.2, while Section 1.3 discusses the motivating factors that drive the need for studying the design concept of TCP congestion control. The problem statement is stated in Section 1.4 where the current issues and challenges of TCP are addressed. In Section 1.5, the research questions are presented, so as to frame the research objectives and scope of which are presented in Section 1.6 and 1.7, respectively. Meanwhile, the research significance is highlighted in Section 1.8, and finally, the thesis organization is outlined in Section 1.9.

1.1 TCP and Ad hoc Networks

The need for wireless computing devices such as tablets, Personal Digital Assistants (PDAs), and notebooks has accompanied the increasing interest in the usage of ad hoc networks. An ad hoc network is a set of wireless mobile or static devices that connect to each other using wireless links, forming a temporary network without depending on fixed infrastructure [12]. In contrast to infrastructure based wireless networks, nodes

(hosts) in ad hoc networks can communicate directly if they are within transmission coverage of each other. Otherwise, data will be sent through intermediate nodes which replace infrastructure devices, such as routers or access point, and directly forward data to the desired destination [13], as shown in Figure 1.1. Thus, normal nodes act as end hosts and intermediary nodes. With recent performance advancements in wireless communication technologies, such as IEEE 802.11 [14] and Bluetooth [15, 16], ad hoc networks are expected to experience widespread use and deployment for various commercial purposes.

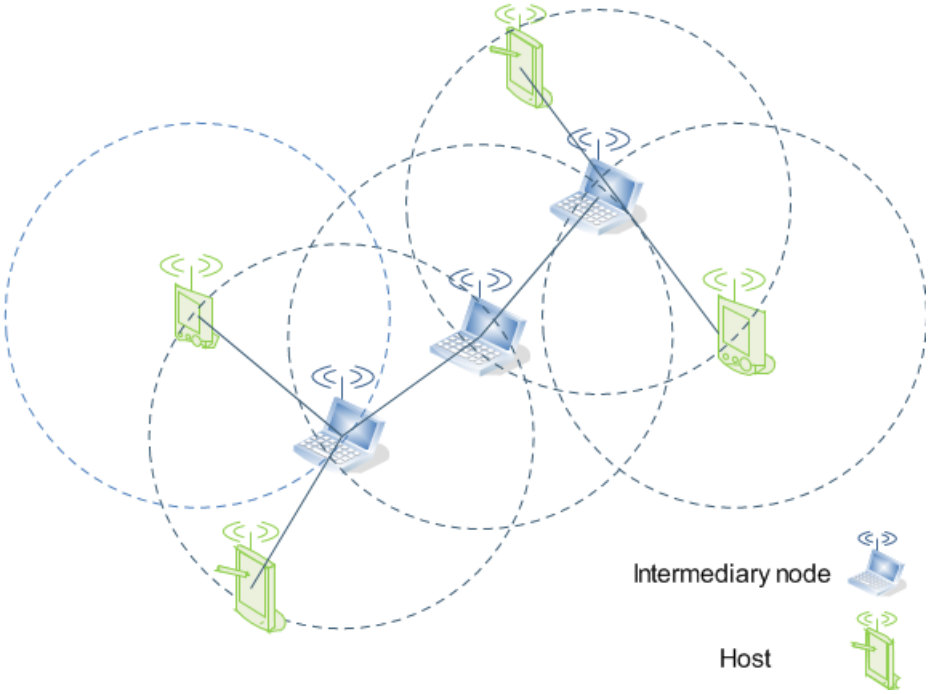


Figure 1.1: Ad hoc Network

Ad hoc networks can be realized through a variety of networks such as Mobile Ad hoc NETworks (MANETs), mesh networks, Static Ad hoc NETworks (SANET), Vehicular Ad hoc NETworks (VANET), home networks, and wireless sensor networks. Ad hoc networks can run in a stand-alone mode or it could be connected to the Internet. This flexibility and ease of building a network without any existing infrastructure would allow users to freely share information anytime and anywhere. Other scenar-

ios where ad hoc networks can be used are in education, such as information sharing among educators on campus; in homes, such as information and document exchange; in emergency disaster relief efforts, such as personal coordination efforts after a natural disaster like an earthquake or flooding; and in the military, used in battlefields and training exercises.

Transmission Control Protocol is the de facto standard protocol that provides end-to-end, reliable, and connection-oriented data delivery over unreliable networks [17, 18]. This protocol is the dominant protocol implemented in the Internet today, since a vast number of applications utilize TCP, such as World Wide Web (WWW), Email (SMTP), File Transfer Protocol (FTP), and Peer-to-Peer (P2P) file sharing. In other words, around 90 percent of Internet traffic is transmitted using TCP [19]. Since TCP is well tuned and due to its wide acceptance for use in the Internet, using TCP over ad hoc networks will make Internet applications portable to and compatible with ad hoc networks, which would enable wireless devices to connect with the wired network and the Internet easily. Additionally, TCP plays a crucial role in data transmission over ad hoc networks due to its reliability [20]. These features and others make TCP use over ad hoc networks a virtual certainty [21].

1.2 Mobile Ad hoc Networks Challenges

Mobile ad hoc networks can be considered as complex distributed systems that allow wireless devices to communicate with each other in an area that does not provide pre-existing network infrastructure [12]. These networks are characterized by self-organization (infrastructure-less), self-configuration (no authority), free movement, scalability, robustness, and easy maintenance. Furthermore, ad hoc networks are not only easy to build and implement but they are also less time consuming and cost effective. However, mobile ad hoc networks inherit several issues from wireless commu-

nications, such as high bit error rate, interference, hidden and exposed terminals, and path asymmetry [22]. Furthermore, mobile ad hoc networks pose new challenges that accompany node mobility, such as frequent route change, route failures, and network partitions [4].

For instance, in wired networks, buffer overflow or congestion at the bottle-neck router is the main cause of packet loss; while in contrast, ad hoc networks suffer from different types of packet losses that are not related to buffer overflow (congestion). Therefore, protocols that anticipate buffer overflow as the sole contributor to packet loss may make a wrong assumption in MANET, thus reacting badly in this situation [23]. As a result, many TCP/IP protocols, specifically TCP which were all well designed and developed for wired networks in the first place, are not directly usable in MANET.

The following subsections shall discuss the main challenges of mobile ad hoc networks in line with the direction of this thesis.

1.2.1 Mobility

All nodes in MANET are free to move independently and randomly in any direction, at any time. This leads to frequent changes in network topology [24]. Therefore, to support mobility and keep ongoing connection alive, each individual node is responsible to discover topology changes of its network [25]. More specifically, once a node moves, the following situations may occur and the associated action triggered:

Route Failures: Since nodes can move in any direction randomly, route failures may occur frequently in ad hoc networks. Furthermore, it may be associated with frequent route changes, route failures, packet reordering, and also some packet loss at intermediate nodes [26, 27].

Network Partition: This event occurs when a node within the ad hoc network moves away from other nearby nodes causing an isolation of some part of the network by dividing it into isolated parts [4], called partitions, as shown in Figure 1.2. From this figure, Node (D) moves away from node (C) causing a route failure between Node (D) and Node (C), because the distance becomes more than the allowed transmission range. Furthermore, this movement will break the network into two partitions, for example in Figure 1.2, the first partition contains the sender (A) and two other nodes (B) and (C), while the second partition consists of the receiver (F) in addition to node (D) and node (E) .

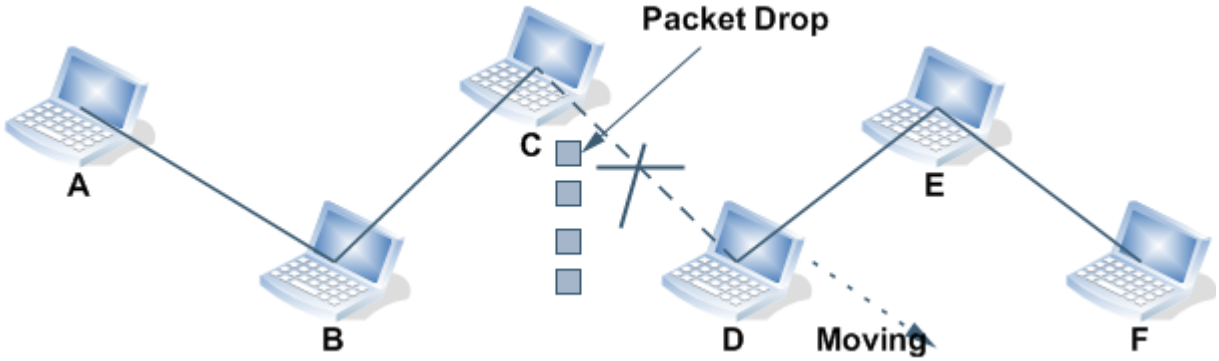


Figure 1.2: Network Partition (Adopted from [4])

1.2.2 Wireless Channel

Wireless channel is used by nodes as a shared medium to transmit and receive data. Wireless channel is well known as being unreliable and unprotected from interfering external signals. Additionally, wireless channel is prone to the following challenges:

High Bit Error Rate: Wireless channel is exposed to errors due to interference, obstacles, and signal attenuation and fading [13]. These errors may produce packet drop within a very short period or receive corrupted data packet/ACK at the receiver/sender. In case of small number of hops, link layer retransmission mechanism is capable of recovering from such loss type, but when the number of hops increases, this

type of loss may generate three DUPACKs that requires TCP to handle the situation differently[28].

Contention: The usage of wireless channel as a shared medium limits the nodes' ability to send packets. Once a node successfully obtains permission to access the wireless channel and performs its transmission, other nodes that are within the sender's transmission range should refrain from transmitting and schedule it for a later time. Therefore, in ad hoc networks, channel contention is considered as a sign of network overload that affects the entire area, rather than a single router [13, 29]. In particular, packet loss due to link-layer contention dominates in ad hoc networks, while buffer overflow-induced packet loss is rare [30].

There are two types of contention, namely interflow and intraflow.

(i) Intraflow contention refers to the contention between data packets and the ACKs within the same flow [31], as shown in Figure 1.3.

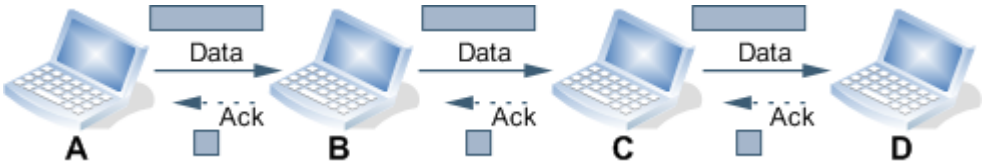


Figure 1.3: Intraflow Contention

(ii) Interflow contention is experienced by a node due to transmission by a nearby node [32], as illustrated in Figure 1.4.

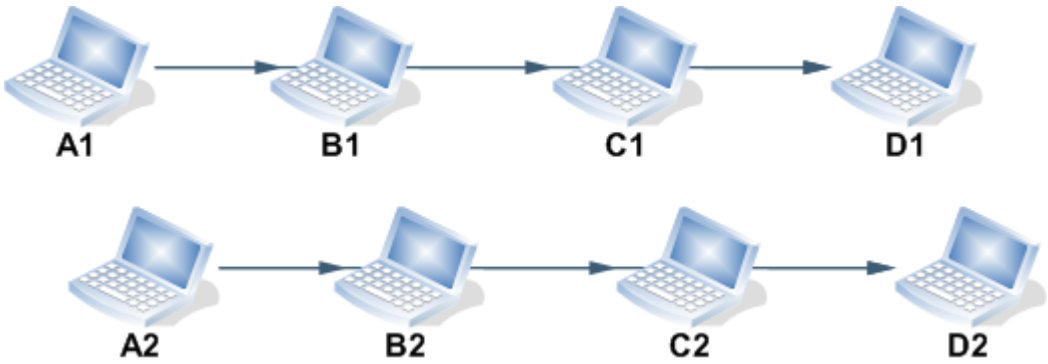


Figure 1.4: Interflow Contention

Hidden and Exposed Station: In ad hoc networks, nodes rely on Carrier Sensing Multiple Access with Collision Avoidance (CSMA/CA) mechanism to determine an idle channel, such as the IEEE 802.11 Distributed Coordination Function (DCF). However, such hidden and exposed terminal problems could not be solved completely using a sensing mechanism.

A hidden terminal is when two nodes located not within the transmission range of each other transmit to the same node. A typical hidden terminal problem is shown in Figure 1.5. Node A (resp. C) is located outside the transmission range of node C (resp. A), but both nodes have data to be transmitted to node B. Meanwhile, node B is located between node A and node C, thus node A cannot detect the transmission of node C because it is located outside the transmission range of node C, therefore Node C (resp. A) is thus "hidden" to node A (resp. C). Therefore, the data frames or packets will collide at node B if they are transmitted at the same time. These collisions make the transmission from node A and node C to node B problematic. Other methods, such as using Request To Send / Clear To Send (RTS/CTS), can be used to solve this problem via the hand-shaking event that proceeds data transmission. However, in situations with multihops, this problem will appear again and again, as the RTS/CTS scheme would not be able to handle hidden terminals.

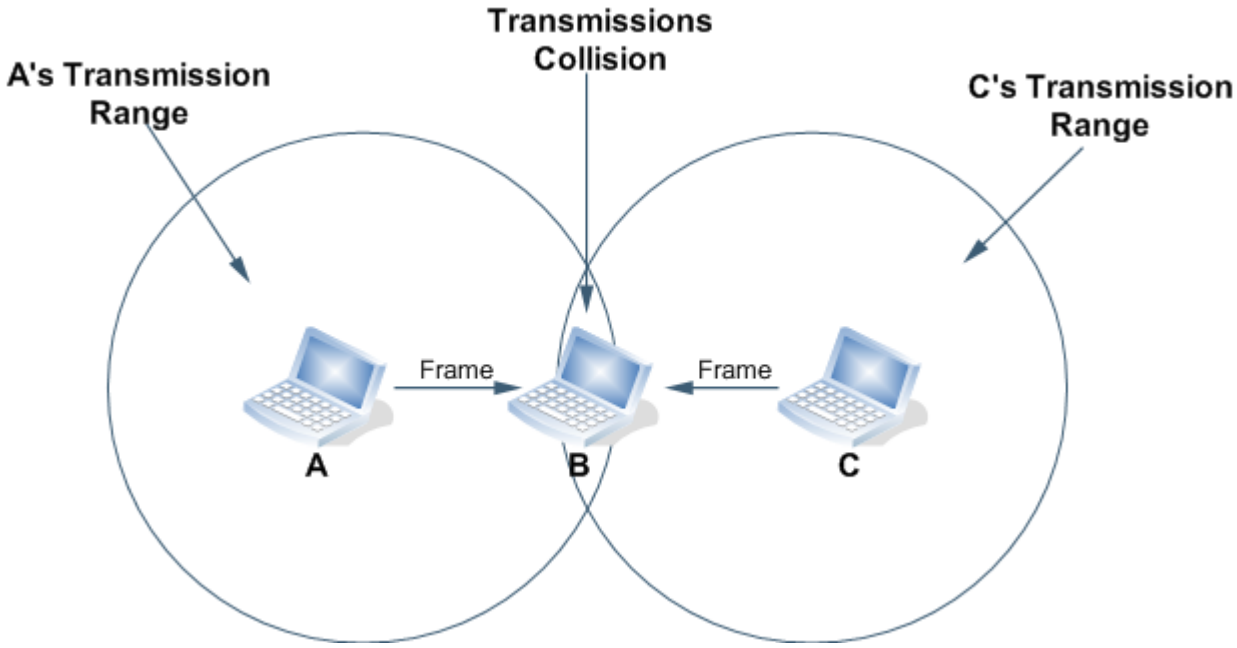


Figure 1.5: Hidden Terminal Problem

The exposed terminal problem happens when a transmission has to be deferred due to the transmission of another node within the sender's transmission range. Figure 1.6 shows a scenario of the exposed terminal problem. Nodes A and C are within node B's transmission range and node A is outside node C's transmission range. Node B has a frame to be transmitted to node A, while node C has data to be transmitted to node D. Node C needs to sense the channel status before transmission, but it detects a busy channel due to node B transmitting to node A. Therefore, node C will defer its transmission, even though this transmission would not cause any interference at node A; thus leading to a reduction of channel utilization.

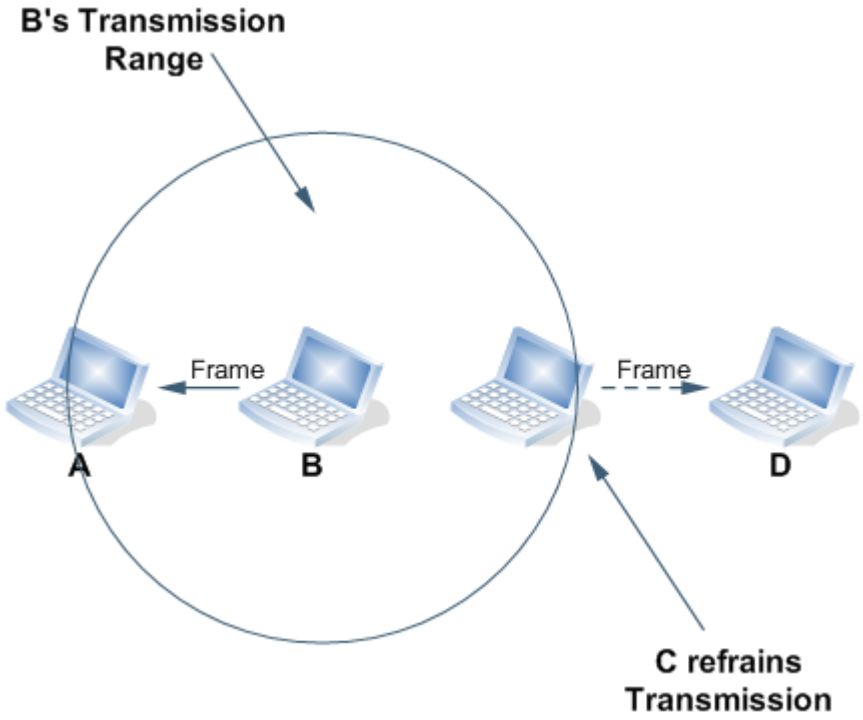


Figure 1.6: Exposed Terminal Problem

1.3 Research Motivation

To exploit the potential use of mobile ad hoc networks, an efficient TCP congestion control is extremely important to support a reliable transport service and make ad hoc networks viable for many applications [23]. Most TCP variants were designed based on the assumption that packet loss occurs mainly due to network congestion at the bottle-neck router [33, 34, 35, 36, 37, 38]. In other words, congestion is the main reason of network instability (an example of typical packet loss is shown in Figure 1.7). However, mobile ad hoc networks are characterized by low bandwidth, dynamic changing network topologies, high bit error rate, and shared wireless channel [4]. These features violate some design principles of TCP that was initially intended for use in wired networks, and thus impose on TCP several technical challenges to resolve this issue, therefore if no modifications are made, TCP performance would degrade [39, 40].

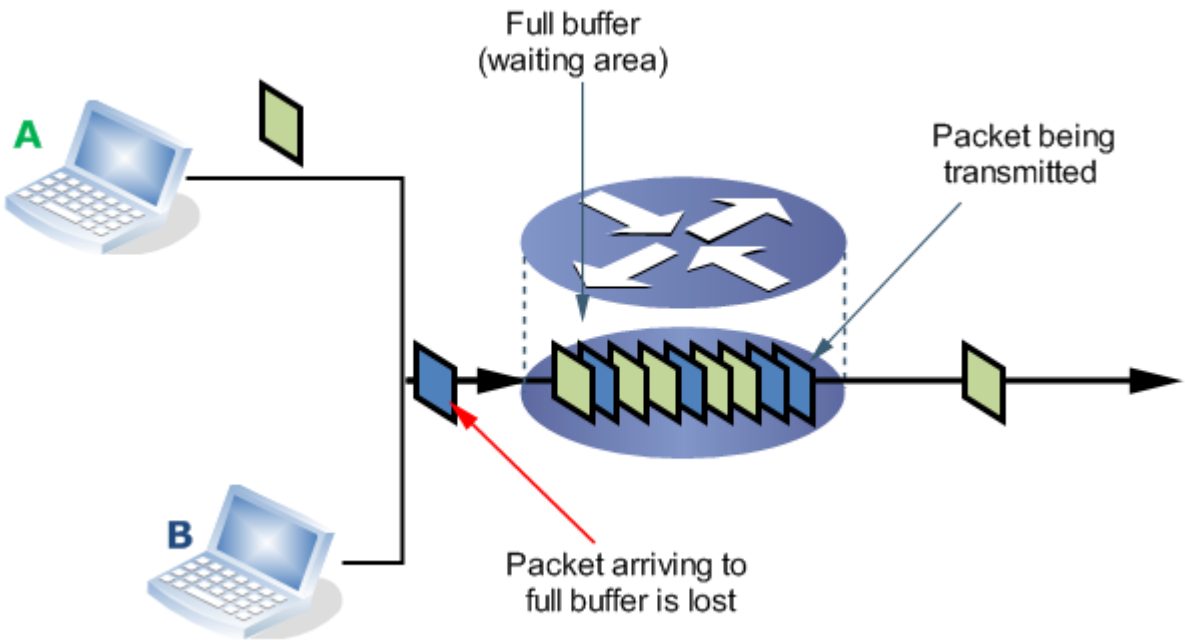


Figure 1.7: Buffer Overflow at the Bottle-neck Router (Adopted from [5])

Accurate insight about the impact of ignoring ad hoc networks' special characteristics on TCP performance is discussed in the following subsections.

1.3.1 Misinterpretation of Mobility Induced Loss as Congestion Loss

Nodes in mobile ad hoc networks may have different relative mobility that will eventually generate route failure (or route change) within the network. Once route failure occurs, it is the responsibility of routing protocol to detect the link break and start looking for a new route to the destination [24]. Meanwhile, all packets in the intermediate node will drop, so no packets will be delivered and no acknowledgements will be received. Sometimes, discovering and establishing such an alternative route may take time, longer than the Retransmission Time Out (RTO) value. In this case, TCP will misinterpret the loss as congestion and reduce its transmission rate, while the RTO value will double due to the BACK-OFF mechanism call [41]. During the process of finding a new route, a sequence of route changes and failures may occur, as well

as packet loss. As a result, TCP will again erroneously trigger a congestion control mechanism to decrease the transmission rate and increase RTO timer value. After a new stable route has been established, the small congestion window *(cwnd)* and Slow Start THRESHold *(ssthresh)* values would reduce the initial sending rate and the large RTO value would reduce the responsiveness of TCP as well as link utilization, thus resulting in poor performance [42].

1.3.2 Contention on the Wireless Channel Access

In IEEE 802.11 Distributed Coordination Function (DCF) multi-hop wireless ad hoc networks, a single channel is shared by all nodes within the network. The shared wireless channel allows a single sender to transmit at any given time, so the number of packets that can be in flight concurrently is limited from a source to a destination. Although nodes cooperate to forward other packets, they compete within a local neighborhood for wireless channel access before transmitting. Therefore, a control handshake of RTS/CTS message precede each packet transmission. Nodes within the sender or the receiver transmission range will defer their transmission upon overhearing these handshaking messages. However, signals are broadcasted in the wireless medium and may interfere with each other. When there exist concurrent transmissions within the interference range of either sender, a collision will be sensed and transmissions may fail. Therefore, the use of RTS/CTS in IEEE 802.11 MAC is a good solution to avoid interference, but interference can still be observed in the mutli-hop topology due to channel contention. Hence, a medium access protocol is required to coordinate any wireless channel access [13]. However, contention-based medium access control schemes, such as IEEE 802.11 MAC DCF protocol, have been shown to significantly affect TCP performance [28, 43, 44].

In addition, the large congestion window size of the sender can cause an excessive

number of medium accesses [45]. Moreover, the correlated arrivals of data packets (also retransmitted packets due to loss or RTO) and their ACKs lead to contention in obtaining access to the wireless channel, thus causing extreme collisions and packet loses [46]. This would thereby reduce achievable throughput of TCP while at the same time increase delay. This is considered the main reason of poor TCP performance over 802.11 MAC protocol, because the packet drops in ad hoc networks often occur because of link layer contentions, while buffer overflows are rare [28].

1.4 Problem Statement

In theory, TCP should run independently regardless of the underlying networks, since it should not care whether it is running over wired networks or even mobile ad hoc networks. In practice, it does matter because most TCP flavors have been carefully designed based on the assumption that all packet losses occur due to network congestion at the bottle-necked router which is inspired from wired network feature, as specified in a large number of RFCs [35, 36, 47, 37, 38]. TCP always infers packet loss as an indicator of network congestion and thus performs a sharp reduction to its sending rate. However, MANET suffers from several types of packet losses due to its node mobility and contention on wireless channel access [4, 26, 40, 48, 49, 50, 51]. To continue ignoring these special characteristics, as discussed in Sections 1.2, would lead to poor TCP performance [4, 21, 50]. The fundamental problem of using TCP in MANET is its misinterpretation of packet loss due to mobility induced route change/-failure as merely network congestion. Furthermore, TCP congestion control does not have a proper technique to handle and avoid contention in MANET [4, 21, 22, 39, 40], resulting in heavy underutilization of the available network resources.

Many researchers had clearly identified TCP problems in mobile ad hoc networks and suggested a number of solutions. The two protocols, namely TCP-friendly Transport

Protocol for Ad hoc Networks (ADTCP) and Explicit Link Fail Notification (ELFN), are particularly relevant here since they were proposed for specific MANET challenges. On the one hand, ELFN has advantages of providing the TCP sender with a more accurate information about network conditions, since the intermediate node can detect route failures faster than the end node. On the other hand, ADTCP does not need intermediate nodes to identify network status and maintain layer-based end-to-end semantics for supporting compatibility with TCP. However, most proposed schemes do not represent complete and ready-to-use protocols, but rather solutions for a small set of the identified problems [52]. Therefore, developing a reliable transport protocol for mobile ad hoc networks is still an open research issue [23, 29]. Thus, the overall goal of this thesis is to propose a new Transmission Control Protocol, named TCP Sintok, for mobile ad hoc networks.

1.5 Research Questions

In addressing TCP performance issues over ad hoc networks, the following research questions were raised up:

i. What are the requirements to enhance the performance of TCP over mobile ad hoc networks?
ii. How could TCP distinguish between mobility induced packet loss and congestion loss within mobile ad hoc networks to avoid unnecessary reduction of transmission rate?
iii. How should TCP adapt the transmission rate or the growth of congestion window to avoid network contention?
iv. What is the impact of the proposed mechanisms on TCP Sintok performance?

1.6 Research Objectives

The aim of this research was to design a new Transmission Control Protocol to detect the cause of packet loss in the IEEE 802.11 mobile ad hoc networks and to adapt the sending rate of TCP based on network conditions. This aim can be further explained by the following specific research objectives:

i. To develop a performance model for TCP congestion control over ad hoc networks which could serve as a benchmark for any future intended enhancement and improvement of TCP in this dynamic environment.

ii. To design a new delay-based Loss Detection Mechanism (LDM) for TCP congestion control to improve the accuracy of packet loss detection in mobile ad hoc networks:

    a. To develop a mathematical model of end-to-end delay in studying the impact of congestion and contention on delay trend.

    b. To validate the proposed delay model by comparing the model results with results obtained from a valid network simulator.

    c. To design delay-based LDM that is capable of accurately distinguishing mobility loss from congestion loss within mobile ad hoc networks.

    d. To verify and validate the proposed LDM via implementing LDM in NS-2 simulation environment; then comparing the identified network state by LDM to the actual network state collected in the trace file.

iii. To design a new Contention Avoidance Mechanism (CAM) to adapt TCP's sending rate (i.e., congestion window size) based on the current network condition in an ad hoc network environment:

    a. To explore the applicability of Communication Accommodation Theory (CAT) in computer communication, specifically in ad hoc networks.

b. To design Contention Avoidance Mechanism (CAM) based on CAT features and properties that is able to accurately adapt congestion window size according to the network condition leading to optimal resource utilization.

c. To verify and validate the proposed CAM by implementing it in NS-2 and comparing the obtained results with other results produced by real test bed and simulation.

iv. To develop TCP Sintok for mobile ad hoc networks based on the standard TCP NewReno:

a. To incorporate the proposed LDM and CAM into TCP Sintok.

b. To modify TCP congestion control action according to network status.

c. To measure the performance enhancement of TCP Sintok and compare it with the standard TCP NewReno to reveal the strengths and weaknesses of the proposed TCP Sintok.

d. To evaluate the performance of TCP Sintok by comparing it with the current TCP proposals for mobile ad hoc networks, namely ADTCP and ELFN.

1.7 Research Scope

The overall goal of this research was initially to develop TCP Sintok for mobile ad hoc networks. More specifically, TCP Sintok is proposed to support any application that requires reliable data delivery service over medium-sized IEEE 802.11 ad hoc networks. Techniques aimed to detect the cause of packet loss and avoid contention would be the point of focus for this study. Furthermore, the study concentrated on maintaining an end-to-end semantic of TCP and looked at the sender side only, as this is recommended to support deployability in such a dynamic environment where a wide variety of mobile devices exists. As such, security and power consumption (energy) issues in mobile ad hoc networks are excluded from the scope of this research.

1.8 Significance of the Research

This research proposed a new TCP Sintok that can detect the cause of packet loss within the scope of IEEE 802.11 ad hoc networks. Furthermore, the proposed TCP Sintok is capable of adapting the transmission rates in line with the current dynamic network conditions. Output of this research can provide reliable delivery service to assist ad hoc network applications that require reliable data transfer features, leading to a set of acceptable performance levels that is required by the user. Additionally, the findings have significant implications for both the usage and the real deployment of mobile ad hoc networks, as well as supporting MANET in asserting a greater impact on mobile device communications. Finally, TCP Sintok would be able to maximize mobile ad hoc networks resource utilization by increasing throughput, and minimizing delay and jitter.

1.9 Organization of the Thesis

This thesis is organized into seven chapters, where the following is a summary of key chapter highlights:

Chapter One presents an overview about the thesis as a whole. Specifically, it presents an introduction to the importance of TCP for ad hoc networks and the need for designing a new transmission control protocol. In addition, this chapter discusses the problem statement, motivation, objectives, and research significance.

Chapter Two critically evaluates the related work and the literature review in the areas related to the research scope. In addition, a theoretical model is proposed as a benchmark for future intended enhancement of TCP.

Chapter Three presents the Design Research Methodology (DRM) as the research

framework to conduct this study and combines several methods adopted to propose and implement TCP Sintok.

Chapter Four is concerned about the modeling of end-to-end delay and proposing Loss Detection Mechanism (LDM) for TCP Sintok. The chapter discusses the design motivation of LDM and its verification and validation.

Chapter Five describes CAT theory and investigates the aplicability of CAT in computer networks. In addition, it introduces and thoroughly discusses Contention Avoidance Mechanism (CAM), including the verification and validation of this mechanism.

Chapter Six introduces TCP Sintok and presents in detail its performance evaluation through simulation.

Chapter Seven states the conclusion as well as the contributions of the reseach work presented in this thesis, then suggests future directions for further studies.

CHAPTER TWO
LITERATURE REVIEW

While Chapter One had introduced and described the overall research plan, this chapter shall delve into greater detail the background and several important past research related to Transmission Control Protocol (TCP) that is being implemented in mobile ad hoc networks, which would assist in defining the general framework of this research. In this chapter, the role of TCP is presented in Section 2.1 with special emphasis on the revealed problem of congestion control, while the performance model of TCP in ad hoc networks is introduced in Section 2.2. Next, TCP proposals that have been classified are described in Section 2.3 to reveal some viable research directions. Additionally, this chapter also provides a range of TCP proposals where some re-active mechanisms are covered in Section 2.3.1; while a number of pro-active mechanisms are covered in Section 2.3.2. Finally, this chapter concludes with theories pertinent to congestion control in Section 2.4.

2.1 Transmission Control Protocol (TCP)

TCP is the accepted de facto standard protocol for commercial communication networks that have been implemented in the Internet and thus it has been universally adopted as the norm. This widely used protocol provides many useful services for numerous applications, such as summarized by [53], connection-oriented, reliability, byte stream data transfer, full duplex, and end-to-end semantic services (as shown in Figure 2.1). It characteristically guarantees that a byte stream sent from a program or application software on the originating computer (known as the sender) would be reliably delivered and in the same order to another program on another distant but linked computer (known as the receiver). Historically, the first specification of TCP was described in RFC793, which later on was modified and documented through a

large number of RFCs, such as [35, 36, 37, 38, 47]. To put things into perspective so as to assist the research movement in this field, a roadmap to all documents that specify and extend TCP was created and it is presented in [54]. Meanwhile, a list of open research issues in Internet congestion control is also provided in RFC6077 [55].

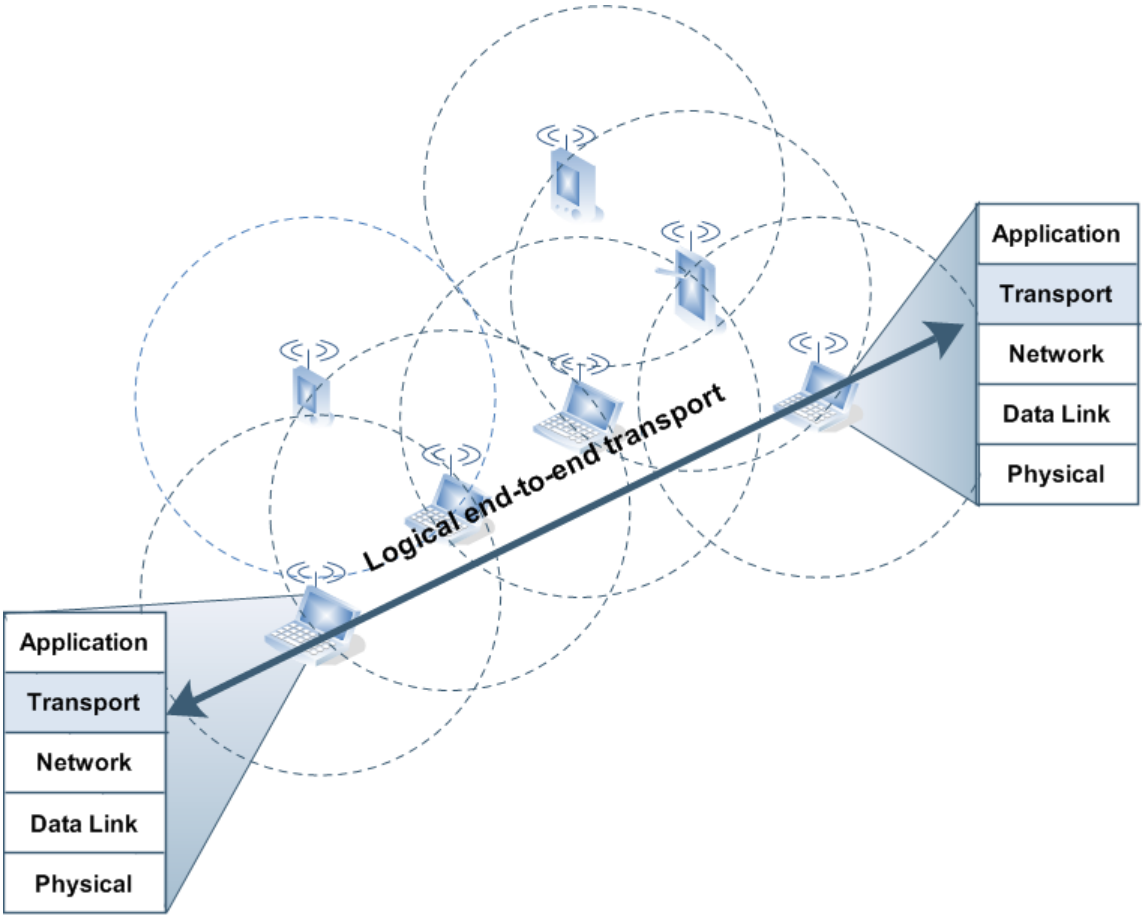


Figure 2.1: Transport Layer Provide Logical End-to-End Communication

Generally, TCP's role can take many forms, but the most common are classified into four different types [5], namely flow control, connection management, retransmission mechanism, and congestion control. Due to the importance of congestion control in the context of this study, the following sub-sections shall provide a full, in-depth description of the congestion control concept, while a brief explanation is presented related to the other tasks of TCP. If more information is required, readers can obtain more details from several sources, for instance RFCs that formally present the functions of

TCP and additionally from Kurose and Ross [5].

2.1.1 Flow Control

Jacobson explained how TCP implements a window based flow control mechanism to avoid overwhelming the buffer space located at the destination [56]. The destination/receiver node advertises the size of its buffer (using the advertised window field) to the related sender/source that would indicate the data byte number the sender can send to the receiver. In other words, the sender's buffer or window size can be determined by the receiver's advertised window size. Each TCP (data or ACK) packet header will contain this information.

2.1.2 Connection Management

In practice, TCP is a transport protocol that is connection-oriented, which would require the establishment of a logical connection between two applications or application processes before any data exchange can even begin. In order to fulfil all the requirements of this part of the overall service, TCP utilizes port numbers and provides a Three-Way Handshaking procedure for connection establishment, as well as a Four-Way Handshaking method for connection termination.

2.1.3 Retransmission Mechanism

In order to ensure reliability of data delivery, which is one of the main responsibilities of TCP, TCP uses acknowledgement (the acronym ACK is used to represent this) to confirm (acknowledge) a successful reception of every segment that has been sent [9]. In the event of a segment arrival being intact at its destination, the receiver will respond by sending an ACK back to the sender to acknowledge the sequence number of the received segment, and in turn, the sender will send a new segment. The use of sequence numbers is for ensuring the in-sequence delivery of segments, as well as to

assist in identifying lost or corrupted segments. However, instead of acknowledging every individual segment as it arrives (which would be time consuming), this protocol employs the procedure of cumulative acknowledgments. Using this concept, an ACK for a segment with sequence number, say (X), will acknowledge all segments correctly received in the order with sequence number up to (X). Then, the receiver will request the sender to dispatch a new segment with sequence number ($X+1$) .

If for one reason or another the segments that have been received are discovered not to be in the correct sequence, i.e., Out-Of-Order (OOO) where some segments might be missing, then the last ACK will be sent again (duplicate ACK). When the duplicate ACK is received by the sender, an assumption would be some reordering of the segments has occurred, but once a third duplicate ACK arrives (four acknowledgments carrying the same sequence number), the sender will then resend the segment with the sequence number mentioned within the duplicate ACK (predicting that this segment has been lost). Additionally, TCP uses a timer called the Retransmission Time Out (RTO), which is used to detect the loss of a segment. If this RTO elapses without receiving any ACK, the sender would then retransmit the unacknowledged segment again. Because of its important role, the RTO value should be set accordingly in order to avoid long delay and premature retransmissions. Details on how to set RTO value are available in [57].

2.1.4 Congestion Control

This section will describe in great detail the main focus of this study. While TCP performs all the other important roles, TCP's main responsibility and top priority is to reduce congestion within the network by adjusting its transmission rate to the current available bandwidth [5]. Congestion can occur when the resource demand exceeds the capacity [53]. In the current modern TCP congestion control implementations, there

could exist three mechanisms [35, 38, 56], namely:

- slow start,
- congestion avoidance,
- and fast retransmit and fast recovery.

TCP congestion control is not so much a provided service solely for the invoking application in a particular instance, because it is a provided service for the whole Internet [58], thus it is a factor that is pertinent for preserving the Internet stability and robustness. Types, details, and flavors of TCP protocol can be found in [36, 37, 38, 59, 60].

Generally, the network is a combination of end systems and connecting devices, for example routers, which can be called nodes. The transmitted packets would pass through many routers before they reach their respective destinations, where each router has a buffer which has limited capacity that stores incoming packets. If the router receives packets faster than it can process and transmit, then it would start to collect and store packets within its buffer. However, as mentioned earlier, this buffer has a limited capacity and thus the router would store incoming packets in its limited buffer until the buffer becomes full. When the buffer limit has been reached, congestion would occur and some packets would be dropped and lost, and thus causing delay in the entire data communication process (as illustrated in Figure 1.7).

In instances where a packet does not reach its intended destination and there has been no ACK sent for it, the sender would attempt to rescue the situation by retransmitting the lost packet based on what has been explained in section 2.1.3. Although this would be the logical thing to do, this action may create more congestion and more

dropping of packets if the filled buffers are still not cleared, which would lead to more retransmissions and greater congestion. Therefore, unless an appropriate mechanism has been implemented, the whole system may collapse and thus no more data can be sent through this particularly congested route or network. This would mean that TCP needs to find a way to avoid this from occurring, or in other words, the network itself should be another entity that plays a major role in the determination of the size of the sender's window in addition to the receiver window. Thus TCP would use congestion control mechanisms to avoid the network resources being overrun by senders.

Besides the receiver's advertised window, which is known as *(rwnd)*, TCP's congestion control has introduced for the connection two new variables, namely *ssthresh* (the slow start threshold) and also *cwnd* (the congestion window). The congestion window (*cwnd*) at a sender-side limits the amount of data the sender can transmit into the network before receiving an acknowledgment (ACK). Specifically, the *(cwnd)* function is to prevent the sender from sending more data than the network can accommodate in the current load conditions. While the receiver's advertised window (*rwnd*) is a receiver-side limit on the amount of outstanding data. Specifically, *(rwnd)* is used to prevent the sender from overrunning the resources (buffer) located at the receiver. The minimum value of the *(cwnd)* and *(rwnd)* would control the data transmission (W). The relationship of these variables is illustrated in the following Equation (2.1).

$$W = min(cwnd, rwnd) \tag{2.1}$$

Last but not least, the slow start threshold *(ssthresh*) determines whether the slow start or congestion avoidance algorithm is used for controlling the data transmission rate. These algorithms will now be explained in the following subsections.

2.1.4.1 Slow Start and Congestion Avoidance

Any TCP sender is required to use the slow start mechanism as well as congestion avoidance mechanism for controlling the outstanding data volume that is imminently to be injected into the network (the details of which are available in [47, 38]). The idea here is to adaptively and dynamically modify *cwnd* to reflect the current network load. Practically, this can be implemented using the process of detecting lost segments. Basically, a segment loss can be discovered using either a duplicate ACK mechanism or a time-out mechanism, as elaborated in section 2.1.3.

Initially, the unknown conditions of the network during the commencement of the data transmission would force the TCP to scan the network for determining the availability of load capacity. This is performed to prevent transmissions from overwhelming the network with large amounts of data. In order to combat this situation, a slow start mechanism can be used at the first stage of data transmission or after the Retransmission Time Out (RTO) has been triggered.

In mathematical terms, the initial *cwnd* (IW) value is required to be equal to or less than the Sender Maximum Segment Size (*SMSS*) multiplied by two ($2xSMSS$) bytes (as an extension, an initial window of three to four segments can be used by the TCP sender). The *ssthresh* starting value is set arbitrarily high. An example would be the advertised window size. On the one hand, the slow start algorithm can be used when the slow start threshold value is greater than the congestion window value (i.e., $ssthresh > cwnd$). On the other hand, the congestion avoidance algorithm can be utilized when slow start threshold value is less than the congestion window value (i.e., $ssthresh < cwnd$). If both the *cwnd* value and *ssthresh* value are equal, an arbitrary selection could be made by the sender to invoke either the slow start method or the congestion avoidance algorithm.

In the slow start period, a TCP sender would increase the *cwnd* value by at most (SMSS) bytes for each ACK received that acknowledges the new data. Slow start will end as soon as the *cwnd* value reaches the *ssthresh* value or when congestion is detected. Meanwhile for the congestion avoidance algorithm, the *cwnd* value is incremented by one full-sized segment per Round-Trip Time (RTT), thus implying a trend that is linear instead of exponential growth. During this congestion avoidance phase, a commonly used formula for updating the *cwnd* value is given in Equation (2.2):

$$cwnd = cwnd + \frac{SMSSxSMSS}{cwnd} \tag{2.2}$$

This congestion avoidance algorithm shall proceed up until congestion has been detected to occur. For every incoming non-duplicate ACK, this update or adjustment is executed. Equation (2.2) gives an acceptable approximation to the underlying principle of increasing *cwnd* by one full-sized segment per RTT. The value of *ssthresh* must then be adjusted when the TCP sender detects a loss of segment through the utilization of the retransmission timer. This value is set to become not more than the following calculated value (Equation 2.3):

$$ssthresh = max(\frac{FlightSize}{2}, 2xSMSS) \tag{2.3}$$

where the value of $FlightSize$ refers to the outstanding data amount within the network.

Upon the occurrence of a timeout, the value of *cwnd* shall then be modified to become not more than the value of the Loss Window (LW), which is equal to a single full-sized segment. Therefore after the dropped segment has been retransmitted, the sender of the TCP segment then uses the slow start algorithm to adjust the window value by incrementing it from the one segment that is full-sized to the new *ssthresh* value. At this point, the algorithm for congestion avoidance again takes over (as illustrated in Figure 2.2).

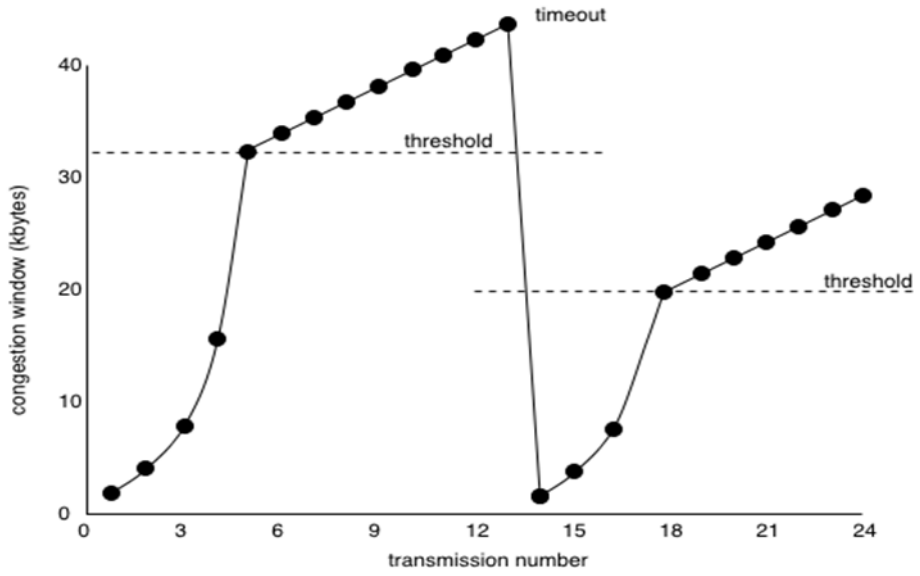


Figure 2.2: Congestion Control of TCP Tahoe (Adopted from [5])

2.1.4.2 Fast Retransmit

One method of detecting lost segments is using Duplicate ACKs, as shown in Figure 2.3. In an event where the sender receives one duplicate ACK, it cannot yet know whether the segment is actually dropped or lost, or it has not been received in sequence. However when several duplicate ACKs are received [61], it would then be reasonable for the assumption that a packet loss event has occurred, can be made. The main role of the fast retransmit mechanism is to speed up the retransmission process by allowing the sender to retransmit a segment as soon as it has enough evidence that a segment has been lost. This means that the sender shall proceed with the retransmission of

the missing segment immediately after three duplicate ACKs has been received, or in other words, four identical ACKs has arrived without any other intervening segments arriving, as an alternative to waiting for the Retransmit Timer (RTO) to expire.

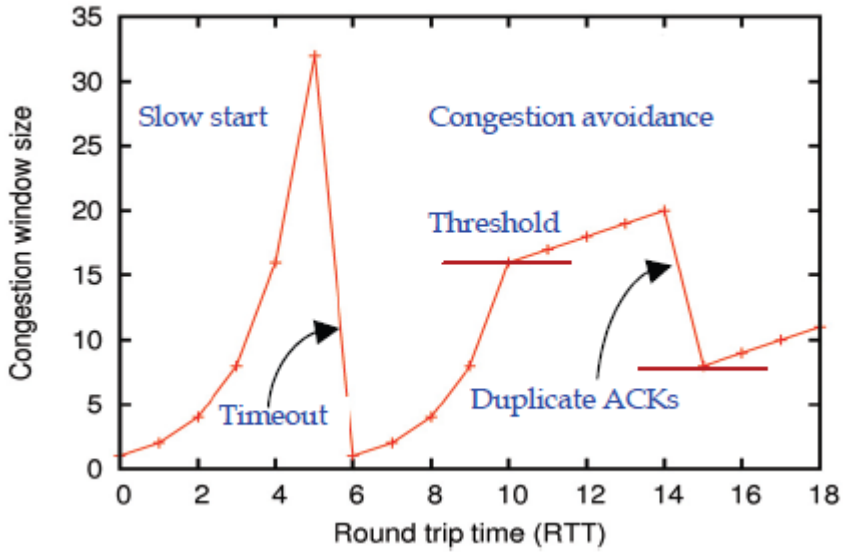


Figure 2.3: Congestion Control of TCP Reno (Adopted from [6])

2.1.4.3 Fast Recovery

In the TCP Tahoe [18] which is one of TCP variations available, the slow start algorithm is launched by the connection proceeding a packet loss detection, but if there exists a condition where packet losses are rare and the window size is large, it is more feasible for the connection to continue from the congestion avoidance phase. This is because it would take some time for increasing the window size from one to the *ssthresh* value. Meanwhile, the aim of the fast recovery algorithm in TCP Reno [61], yet another variant, is to be able to perform this type of behavior and process. In connection utilizing the fast retransmit method, the sender is able to use the duplicate ACK flow to establish a record to clock the transmission of segments. The fast retransmit algorithm and fast recovery algorithm are usually implemented together, and they are given in the following:

i. After the arrival of the third duplicate ACK, the *ssthresh* is adjusted to become one-half the current congestion window, *cwnd*, but no less than two segments.

ii. Retransmission of the lost segment will occur with the *cwnd* being set to *ssthresh* value plus 3*xSMSS*. This effectively "inflates" or artificially increases the congestion window value according to the number of segments that had left the network but the receiver has buffered and accounted for (the number of which is three segments).

iii. Next, any further duplicate ACKs that are received, this would cause an increase in *cwnd* by *SMSS*. This artificially increases the congestion window value and thus reflects the additional segment(s) that has/have left the network.

iv. A segment is then transmitted, depending on whether it is allowed by the new *cwnd* value and advertised window value on the receiver's end.

v. The arrival of the proceeding ACK which acknowledges new data segment will cause the *cwnd* value to be set according to the *ssthresh* value, which is the value that is set in step 1. This would reflect the decrease or "deflating" the window value.

This particular ACK forms the acknowledgment that was requested during the retransmission process from step 1, which is exactly one RTT value after the retransmission process. In addition with the condition that no segments were lost, this ACK would also form the acknowledgement of all the intermediate segments that were sent from the actual lost segment to the receipt of the third duplicate ACK.

NewReno Modifications to Fast Recovery

The TCP NewReno variant was proposed in 1995–1996 by Floyd et al. [37], which is a modification of TCP Reno to recover multiple losses that occurs within the same

window, thus improving retransmissions during the fast recovery phase (as illustrated in Figure 2.4).

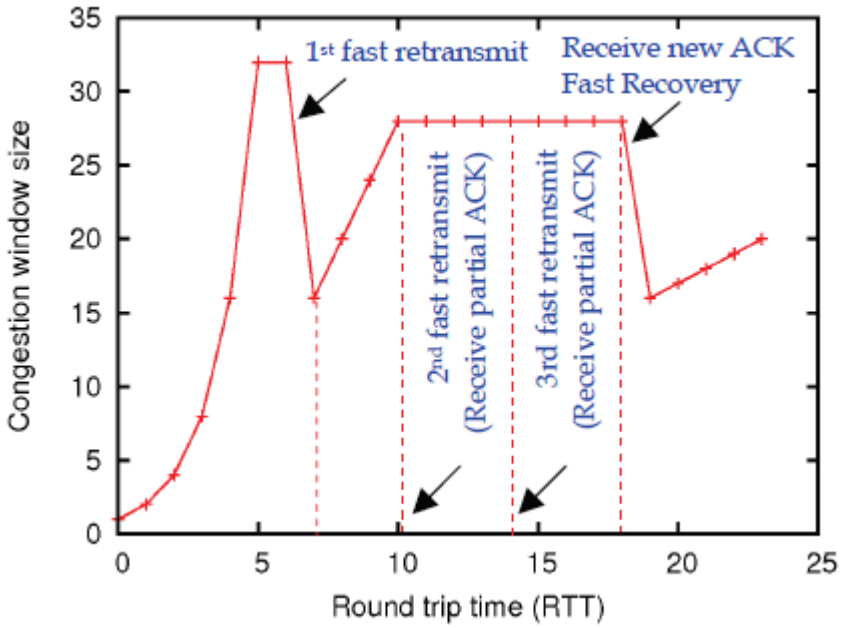


Figure 2.4: Congestion Control of TCP NewReno (Adopted from [6])

In this phase, a new unsent segment from the end of the congestion window is sent for every duplicate ACK that is returned, to keep the transmit window full. For every ACK that makes partial progress in the sequence space, the sender assumes that the ACK points to a new hole, and the next segment beyond the acknowledged sequence number is sent. The progress in the transmit buffer resets the timeout timer, and this allows TCP NewReno to fill large or multiple holes in the sequence space. High throughput is maintained during the hole-filling process, because NewReno can send new segments at the end of the congestion window during fast recovery. When entering fast recovery, TCP records the highest outstanding unacknowledged segment sequence number. Upon the acknowledgment of this sequence number, TCP returns to the congestion avoidance state. TCP NewReno will misinterpret the situation if there are no losses, but instead reordering of segments by more than three segment sequence numbers. In such a case, NewReno mistakenly enters fast recovery, but when the re-

ordered segment is delivered, ACK sequence-number progress occurs and from there until the end of fast recovery, every bit of sequence number progress produces a duplicate and needless retransmission that is immediately acknowledged. TCP NewReno protocol variant can substantially outperform the TCP Reno at high error rates. Because of its able performance, the TCP NewReno variant has been implemented as the default TCP variant for MS Windows XP.

2.2 Performance Model of TCP Congestion Control

As briefly described in Chapter One, this study shall introduce a novel model to present the state-of-the-art TCP congestion control over ad hoc networks. The model is capable of guiding researchers' steps toward achieving greater enhancement of the TCP congestion control mechanism. This model is also known as the performance model (Figure 2.5). It comprises a collection of ad hoc network related factors impacting TCP. It also exhibits the important measurable factors that could be used for studying the performance of TCP under various conditions. The factors/links that require greater attention in the model were identified using experimental studies conducted by the researchers as a basis. Furthermore, a review and analysis of the current literature, data were obtained to assist in supporting, verifying, and validating this new model. The discussion has been classified according the respective phenomena, which are: high bit error rate, mobility, and contention [4, 62, 40, 63].

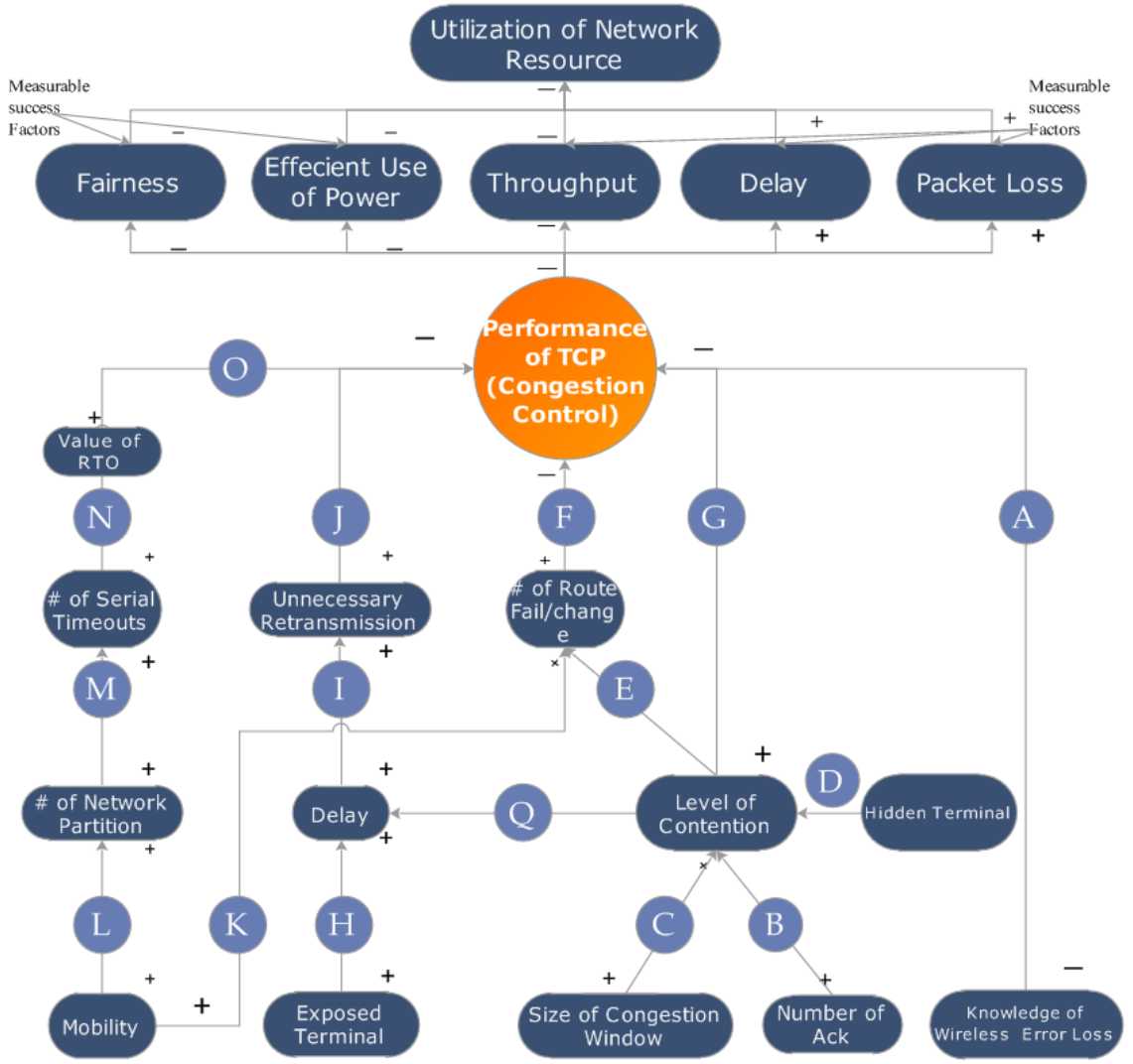
2.2.1 High Bit Error Rate

Characteristically, it has been widely accepted as a well-known fact that the current wireless channel is inherently unreliable and weak as well as being unprotected and exposed to outside signals and interference. Thus, technologies in the link layer have to be able to handle situations involving packet corruption. However, according to [64], occasionally these corrupted packets may not be able to be repaired and provided

on time. Link layer retransmission techniques are capable of recovering from channel error loss in situations with low number hops, thus this would be low error rates can be achieved. However, increased number of hops would lead to greater difficulty in recovery [65]. If and when the link layer ceases all attempts executed at the link-level retransmission, TCP would consider the actual loss as an indication of congestion. According to [21, 52, 66], this misinterpretation of packet loss as congestion rather than due to the actual wireless channel error, has badly degraded the overall TCP performance over the network. Therefore, the gap identified here is that a better and improved detection mechanism is required that correctly detects and interprets the packet loss not as a congestion occurrence, but rather as a corruption event.

Therefore, from the elaboration in the previous discussion above, the conclusion reached is represented by Link A in Figure 2.5 and the following statement:

- Statement A – Low knowledge of wireless error loss has negative impact on TCP performance [21, 66, 67, 68] (i.e., poor throughput).



A	Low Knowledge of Wireless Loss has negative effect on the performance of TCP.	I	High delay may increase the unnecessary retransmission
B	Large number of generated Ack will increas the level of Contention	J	The large number of unnecessary retransmission has negative effect on the performance of TCP
C	Large size of congestion window will increase the level on contention	K	Mobility may increase the number of route failure/change
D	Hidden Terminal will increase the level of contention	L	Mobility increases the probability of network partition
E	The high level of contention may increase the number of route fail/change	M	Long network partition produces a serial of timeouts
F	Frequent route failure/change has negative effect on the performance of TCP	N	Frequent timeouts increases the value of RTO
G	The high level of contention has negative effect on the performance of TCP	O	High RTO value has negative effect on the performance of TCP
H	Exposed Terminal will increase the delay	Q	High level of contention will increase the delay

Figure 2.5: Performance Model of TCP Congestion Control

2.2.2 Contention

Meanwhile, other researchers who want to achieve the same goals had looked at the problem from different perspectives and using various approaches. The nodes that are located within ad hoc networks would utilize a wireless channel for sending and receiving data, but this shared medium would only allow a single transmission at any given time by a single transmitter [45]. This would mean that all members within a local neighborhood that wish to transmit would have to compete with each other for access to the wireless channel before proceeding with any data transmission. Additionally, the packets numbers which can be concurrently flowing through the network would be restricted from a starting source to a given destination. Previous research by [28] had reported that packets may drop because of link-layer contention, a condition of which dominates in ad hoc networks, while common events in wired networks such as buffer overflow-induced packet loss, would be considered a rare event in ad hoc networks.

In order to overcome this problem, there are three identified factors that could contribute in increasing the level of contention, which are: (1) according to [69, 70, 71], contention of TCP data packets within the forward path would be increased by a large congestion window; (2) according to [72, 73], the contention between data and ACK packets within the forward and return paths would be increased by the ACK generation for every packet that arrives; and (3) the role of hidden and exposed terminals [73, 74]. Thus, as summarized by [74, 75], an excessive number of medium accesses by these factors would increase the level of contention. As a consequence, the following can occur because of the high level of contention: (1) overall TCP performance would degrade; (2) degradation of overall TCP performance (Link F in Figure 2.5) due to increase in route failure probability (Link E in Figure 2.5); or (3) and according to [21], further decrease in overall TCP performance due to high delay that leads to

unnecessary retransmission (Link I in Figure 2.5).

The following links and associated statements were produced from the previous discussion and shown in Figure 2.5:

- Statement B – Large congestion window size increases the level of contention [69, 71, 76, 77];
- Statement C – Large number of generated ACKs increases the level of contention [39, 72, 73, 78];
- Statement D – Hidden terminal increases the contention level [28, 74];
- Statement G – Large (high) contention level has negative impact on the TCP performance [28, 73] (i.e., poor throughput and fairness; and high delay and packet loss);
- Statement E – Large (high) contention level increases the probability of route failure/change [75];
- Statement F – Frequent route failure/change has negative impact on the TCP performance [21, 45, 51];
- Statement H – Exposed terminal increases the delay [21];
- Statement Q – High contention level increases the delay [21];
- Statement I – Large (high) delay increases the number of unnecessary retransmission [79, 80]; and
- Statement J – High number of unnecessary retransmission has negative impact on TCP performance [45, 79, 80] (i.e., poor throughput).

2.2.3 Mobility

Within the mobile ad hoc networks, constant connection is required to be kept active and alive in order to support this mobility, since all devices are free to move randomly.

However, the topology will undoubtedly undergo change as the host moves, thus leading to network partitioning and/or route failure/change. Authors in [4, 26, 48, 81, 82] observed that mobility-induced frequent route failures have a significant impact on the performance of TCP, due to TCP's inability to recognize the difference between route failure/change loss and congestion loss.

This fact has led to the following links and associated statements, as shown in Figure 2.5:

- Statement K – Mobility increases the number of route failure/change [26, 48, 82, 83, 84]; and
- Statement F – Frequent route failure/change has negative impact on TCP performance [4, 26, 48, 81, 82, 83] (i.e., poor throughput, high loss, and delay).

Meanwhile, the phenomenon of network partition occurs in MANET when a node exits (move away from) the network thus causing the MANET to break into two parts and an isolation emerges. These isolated parts are generally referred to as partitions. During this period, the TCP sender cannot receive any ACKs transmitted by the TCP receiver. The persistence of this disconnectivity greater than the Retransmission Time Out (RTO) of the sender node would trigger the exponential back-off algorithm. This algorithm consists of doubling the RTO value whenever timeout expires, therefore exaggerating the effects of network partition. Hence in the event of network partition, a series of expired RTOs will follow (Link M) and the RTO value will double (Link N), leading to a large RTO value which will reduce the responsiveness of TCP (Link O). In other words, as according to [85, 86], the key factor affecting TCP performance is consecutive timeout.

The following statements and links in Figure 2.5 were produced from the above discussion:

- Statement L – Mobility increases the probabilities of network partition [24, 84, 85];
- Statement M – Long network partition produces a series of timeouts [85, 86];
- Statement N – Series of timeouts increases the RTO value [85, 86]; and
- Statement O – Large RTO value has negative impact on TCP performance [85, 86].

2.2.4 Discussion

The A to O derived statements are combined to form the performance model for TCP Congestion Control, as shown in Figure 2.5. Researchers can now easily determine, based on this proposed model, the key factor that they wish to address in their respective studies in order to contribute toward improving TCP performance in ad hoc networks. As an example, a research focus maybe on the reduction and elimination wireless error loss influence on TCP congestion control performance. Accordingly, a potential research output would be a mechanism for transforming the "low knowledge of wireless error loss" to "high knowledge of wireless error loss", thus reflecting the relationship between TCP congestion control performance and wireless error loss knowledge. Also as a consequence of using the proposed detection mechanism to address the specific TCP congestion control problem, the researcher can design a conceptual model illustrating the expected desired situation.

2.3 TCP Proposal Classification for Mobile Ad hoc Networks

To date, several schemes were proposed to improve TCP performance over MANET. Al Hanbali et al. [4] classified proposals into layered proposal and cross layer pro-

posal, and then reported that cross layer proposals may report better performance. However, layered proposals maintain the TCP end-to-end semantics, which is designing protocols in isolation, so they are considered to be a long term solution rather than short term. Furthermore according to [1, 87], end-to-end proposals provide flexibility for backward compatibility, require no network support, and are easy to implement and deploy. Therefore, this study focused on proposing an end-to-end TCP Sintok variant with the ultimate objective of improving TCP performance within the ad hoc networks.

Following the results and recommendations of previous research, this research has classified the end-to-end TCP proposals into two main categories, which are the reactive approach and proactive approach. On the one hand, reactive approach proposals use a loss differentiation algorithm to identify the network status only after the packet loss event has occurred, and then take appropriate action in recovering based on the network condition. On the other hand, proactive proposals aim to improve the spatial channel reuse from the start without waiting for any packet loss to occur before taking action, and reduce the effect of medium contention on TCP performance.

Thus, the popular reactive-proactive TCP proposal classification is described in the following sections.

2.3.1 Reactive Approach

This section discusses in greater detail the recent proposals that have been highlighted to cope with the problem of TCP being unable to differentiate the different MANET packet loss types. Out of all proposals, they can be characteristically classified with their capability in detecting the reason for packet loss, which are as follows: dealing with route failure/change loss, dealing with wireless error loss, and hybrid approach.

The following subsections describe the schemes according to the aforementioned taxonomy.

2.3.1.1 Dealing with Route Failure/Change

The following proposals focus more on the differentiating of congestion loss from route change/failure loss. These include works by [86] investigating fixed RTO, proposing the TCP Adaptive RTO (TCP AR) [88], looking into Partition-Aware TCP (PAT) [89], implementing TCP Detection of Out-of-Order and Response (TCP DOOR) [7], introducing Protocol for Mobility-induced Packet Reordering (TCP-R) [90], and examining Tuning Rules in TCP Congestion Control on MANET [91].

a. Fixed RTO: Dye and Boppana in [86] compared the different TCP flavor performances under three different routing protocols within the MANET environment. The results of that study had shown that the exponentially growth of retransmission timeout is problematic in MANET because of TCP being unable to differentiate between congestion and route failure, which are packets loss caused by two completely different reasons. As a possible solution to distinguish between congestion and route failures, the authors proposed employing a heuristic approach while fixing the RTO value as a constant after the first retransmission. When the second RTO expires and still an ACK is not received, in other words when two timeouts expire in sequence, route failure has occurred during transmission rather than network congestion, can be concluded by the sender. Therefore, to overcome this problem after the packet loss type identification, the unacknowledged packet can be retransmitted again without doubling the RTO value, which remains a constant value until the retransmitted packet is acknowledged and the route is reestablished.

Using NS2-based simulations, a comparison of this particular mechanism with em-

bedded selective acknowledgment extensions (SACK) and delayed acknowledgment extensions (DACK) in TCP executed using three different reactive and proactive routing protocols was performed. The results of their NS-2 simulations showed that both DACK and SACK yielded minor improvements, but a performance increase comparable to ELFN [13] would be possible with fixed RTO value. Similar to ELFN's mechanism of sending probe packets, fixed RTO would lead to sending of packets periodically, but it is simpler and no feedback is required. Nevertheless in cases of interoperation wireless and wired environments, the assumption that two consecutive timeouts being related to route failures would pose as a limitation. Furthermore in cases of heavy contention, this matter would require further analysis.

b. TCP Adaptive RTO (TCP AR): Meanwhile, Touati et al. in [88] got similar results to [86] that consecutive timeout is a key factor that affects TCP performance. However, in contrast to the Fixed RTO factor, a new mechanism called TCP Adaptive RTO (TCP AR) was proposed [88], comprising the adaptation of RTO value into the network conditions. In order to distinguish network congestion from route failure, these authors deployed a throughput filter, which was already proposed previously in research associated with the Adaptive Bandwidth Share Estimation (ABSE). The level of path congestion can be determined by the comparison of instantaneous sending rate obtained from *cwnd* and the estimated rate, which is illustrated in the following TCP AR Pseudo Code.

Algorithm 2.1 TCP AR Pseudo Code

// indication of temporary route loss

Step 1. If (thk * RTT >cwnd)

Step 2. Then maintain RTO value (fixed)

Step 3. Otherwise set RTO = RTO*2. // Indication of congestion loss

Using combinations of various network conditions in NS-2 and proactive and reactive routing protocols, the authors then compared TCP AR throughput with TCP NewReno and TCPW ABSE. The results of the simulations showed that in terms of efficiency, TCP AR was able to achieve the best performance by approximately a 161% gain in throughput when NewReno is implemented using OLSR, and up to approximately 277% when using DSR, all of which showed more outstanding performance improvements as the node's mobility increases.

c. Partition-Aware TCP (PAT): Another approach named Partition-Aware TCP (PAT) Likes TCP Fixed and TCP AR. Because of the RTO value rapid increase phenomenon, much focus has been given on the current exponential back off mechanism of TCP being unable to effectively respond to reconnection after a long disconnection period. In order to cope with this problem, Qianwen et al. proposed several modifications to the RTO back off mechanism of the current TCP [89]. Their study looked at the duration of disconnection focusing more on the density and speed for four models of mobility, which were Gauss Markov (GM), Reference Point Group Mobility (RPGM), Random Waypoint (RW), and Manhattan Grid (MG). The researchers used MATLAB to derive an equation that described the relationship between the studied factors. A relatively similar trend was observed in the results for the duration of disconnection related to the four models. The exponential back off mechanism was proposed for modification, which is adjustment of the increment method from exponentially to arithmetically.

The maximum possible dead time would be incurred when RTO is large. Thus for each mobility model, a capping of the maximum RTO value at the average duration of disconnection would be done in order to avoid from RTO approaching a value that is too large.

They then compared the potential improvement of PAT with Fixed-RTO and legacy TCP using NS-2 under the same environment of four mobility models. Especially at low node densities and high speeds, their results supported the PAT scheme being capable of offering much more throughput improvement compared to the original TCP. Thus, significantly less number of retransmissions could prove useful and appropriate in real networks, even though the throughput improvement may not exceed the Fixed-RTO for some scenarios with single TCP flow. Nevertheless as stated by the authors themselves, the outcome of their study as derived from simulation results was solely based on partition behavior of various mobility models. However, previous mathematicians using mathematical models had successfully described the behavior and movement of these mobility models, so deriving from mathematical equations a model to explain partition behavior under different mobility models would be more precise and appropriate.

d. TCP DOOR: Meanwhile as an indicator for route change and to avoid unnecessary invocation of congestion control, Wang and Zhang used an out-of-order (OOO) data packet delivery and/or ACK [7]. By definition, when an earlier sent packet arrives later than the next packet sent after it, OOO would occur (as shown in Figure 2.6).

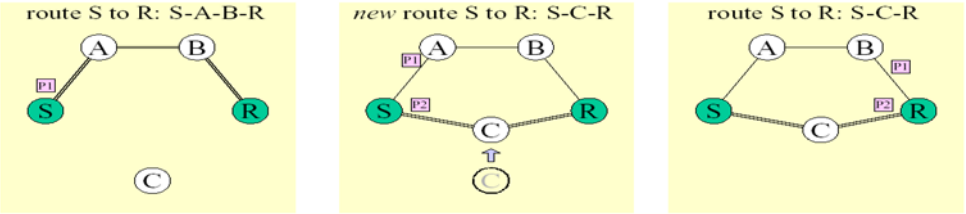


Figure 2.6: A Possible Case of Route Change (Adopted from [7])

Both the sender and receiver sides can perform the OOO event detection where OOO events would be detected by the sender using the non-decreasing property of ACK sequence numbers. Each time an ACK packet arrives at the sender, a comparison is

made between the sequence number contained in the ACK and the one in the previously sent ACK. If the previous sequence number is greater, OOO is immediately declared by the sender. In cases where ACKs carry the same sequence number (duplicate ACK packets exist), the authors had proposed a modification of the TCP by the addition of a TCP option of a single byte in the ACK header, which they named as the Acknowledgement Duplication Sequence Number (ADSN). When the first ACK for a data packet transmitted by the receiver, the ADSN variable would be set to zero. This ADSN number would be incremented every occasion that the receiver transmits a duplicate ACK containing the same sequence number. Therefore, the TCP sender would be capable of detecting a delivery of OOO since every ACK duplicate shall contain a different ADSN number. However, an additional two more bytes are required for the receiver to hold the TCP options, named the TCP Packet Sequence Number (TPSN), in order to successfully detect the OOO event. With each TCP packet including the retransmitted packets, the TPSN can be incremented and transmitted. For every OOO event that the receiver detects, the sender can then set a specific option bit located within the ACK packet header as a notification.

When the TCP sender is made aware of the OOO event, it can then take action according to two mechanisms that have been suggested by the authors. Firstly, the sender may temporarily keep its state variables constant to disable congestion control mechanisms in TCP for a specific period of time (T1). In addition, if the sender can invoke Instant Recovery. This is when the sender has entered a congestion avoidance state (by halving its window size value) and suffered from congestion symptoms during the previous period (T2). The sender should then revert immediately to the state before any action for congestion avoidance was initiated.

The TCP Detection of Out-of-Order and Response (TCP DOOR) was implemented

by the authors as a TCP-SACK derivative, and they further investigated its capability by conducting simulations as an evaluation of TCP DOOR performance vis-a-vis the standard TCP-SACK. TCP DOOR can significantly enhance TCP throughput by an average of approximately 50%, as shown by their simulation results. The authors stated that it would be sufficient for the OOO event to be detected by either sender or receiver since both performing the detection do not significantly yield better results. This approach is recommended for further investigation in situations where it would be difficult to adopt a feedback-based approach, such as mixed wired and ad hoc wireless network environments. However, especially in multi-path routing cases, the issue that arises is that what if route changes did not cause the OOO event. More analysis is required to investigate the assumption that OOO events are the exclusive results of route change.

e. TCP-R: From another perspective, Wei et al. [90] proposed TCP-R to address the issue of route change and mobility-induced link failure causing poor TCP performance, which may lead to the persistent reordering of packets under the MANET environment. TCP-R delays the triggering of congestion response algorithm for a short period of time in order to enable the receiver in accepting packets that travel using different routes, which in contrast to TCP DOOR which attempts to detect OOO events prior to taking any action. Therefore, unnecessary retransmission and reduction of congestion window are avoided when TCP-R proceeds as if the packet was never lost in the first place, if and when the packet is received before the end of delay period. However, if by the end of the set delay period the packet does not arrive, a trigger of the fast retransmission congestion recovery algorithm occurs.

Extensive simulation by the researchers showed that under persistent packet reordering, TCP-R achieves higher throughput than the existing mechanisms. Similarly, they

found that in cases where reordering does not occur, the TCP-R protocol would maintain equivalent throughput rates as in the NewReno variant, and thus fair in sharing network resources. However, especially in cases of frequent route failure and change that are very common in MANET, it was observed to be very difficult in obtaining the desired delay value, which has two implications if an inappropriate value is set. Firstly, the protocol is not fast enough to respond to congestion when too large a delay is set, and too small delay would not allow the receiver to receive the "lost" packet. So, the focus here would be on the accuracy of the delay period calculation.

f. Tuning Rules in TCP Congestion Control: Yang and Lin discussed several TCP parameters that can affect TCP performance in the MANET environment [91]. They found that TCP performance is affected by congestion control, which involves two preliminary parameters of TCP congestion control, namely RTO and congestion window.

The RTO is used to check whether the time for the ACK to arrive is greater than the RTO value for detecting congestion state (loss). Therefore, RTO not only becomes a key to indicate packet loss, but it is also a key for adjusting the congestion window. The Round Trip Time (RTT) samples between any two end-points is used to calculate the RTO value. However since RTT is not very stable, calculating RTO is not an easy task, especially when unstable RTT is characteristically produced from the free movement of mobile node, such as the case in the MANET environment. The authors suggested improving the efficiency of computing the RTO in the Jacobson Karels algorithm by calculating the EstimateRTT and Deviation from the bit shift instead of floating point, which illustrated as follows:

Algorithm 2.2 Using bit Shift to Calculate Deviation and EstimateRTT

Step 1. SampleRTT - = (EstimateRTT >> RTT_BITS) ;

Step 2. Then { EstimateRTT + = SampleRTT ; }

Step 3. if (SampleRTT < 0) SampleRTT = - SampleRTT ;

Step 4. SampleRTT - = (Deviation >> VAR_BITS); Deviation + = SampleRTT ;

Step 5. Deviation + = SampleRTT ;

Step 6. TimeOut = (EstimateRTT + (Deviation << VAR_EXP) * TICK);

The values of TICK, VAR_EXP, VAR_BITS, and RTT_BITS variables would directly affect the RTO value. In order to find out the better or optimal values and combinations of the parameters, the researchers had compared throughput, transmission time, and packet delivery ratio of each of the parameter's combinations under different mobility rate models. They then proposed fine-tuning strategies for these RTO component parameters. In addition, they stated that different RTO parameters in different mobility rates had yielded various TCP performance results. Hence, there is a greater need to select the most appropriate or optimal RTO parameters according to the discernible constraints and applications of the real-world environment. However, adjusting or modifying RTO parameters certainly is not an easy task. Moreover, if the application requirements are known, the issue that arises is that how the mobility models can be identified.

2.3.1.2 Dealing with Wireless Error Loss

Moving on to the next concept of distinguishing congestion loss from wireless channel loss, the following sections will elaborate further on this topic with proposals and previous work performed by various active researchers in this particular field.

a. TCP-MEDX: In order to address this distinguishing issue, Xiong et al. in [92]

developed the TCP-Mobile Error Detection eXtension (TCP-MEDX) through their research to address the non-congestive loss issue, distinguishing loss between congestion and transmission error in MANET. Two parameters, namely propagation delay and Differences between propagation delays, are used by TCP-MDEX as a predictor for the determination of network congestion level. The following equation, similar to RTT calculation in TCP, can be used to calculate the average propagation delay:

$$P = \alpha P + (1 - \alpha) P_{new} \tag{2.4}$$

where α is the parameter that holds a value ranging between 0 and 1,

(P_{new}) is the most recent value received for propagation delay, and

(P) is the threshold, which is also the average propagation delay.

The propagation calculation involves both sender and receiver. When the receiver accepts a packet, the $(Trecv–Tsnd)$ calculation for propagation delay begins, where the time when the source sends the packet is represented by *Tsnd* and the receiving time is *Trecv*. The delayed acknowledgment strategy is not adopted by the TCP-MEDX for constantly informing the sender about the state of the network. Meanwhile, the sender updates its own propagation delay while at the same time stores the propagation delay value obtained from the ACK header in its propagation buffer. The TCP-MEDX performance is heavily influenced by the two (α) and (β) variables, so the authors put forward a recommendation of values, namely proposing (α) to be set between 0.2 and 0.8, while (β) is at 1.2. If two conditions are satisfied, the state of the network is considered congested by the TCP-MDEX method. The conditions are the delay of

propagation value increases by a certain number of packets, and the current delay of propagation value is far beyond a threshold equaling $(\beta * P)$, where β is specified by the application and P is an average propagation delay.

TCP-MEDX will investigate any event related to the detection of packet loss according to the above two criteria. A decision will be made as to whether a congestion has indeed occurred or otherwise. The congestion window will be halved in the event of congestion occurring, otherwise an assumption will be made by TCP-MEDX that a transmission error has caused the packet loss and the congestion window will remain unchanged.

From the simulation results, when compared to standard established TCP, TCP-MEDX achieves significant improvement in throughput and is more capable of properly identifying cause of packet loss. However, it was not clear how the detection of the loss was performed, either due to retransmission timer expiration or 3DUPACKs occurring. However, since multiple packet losses within the same congestion window may have occurred, halving the congestion window upon the expiration of retransmission timer for resolving a burst loss event can be considered ineffective. Besides, because it is only initiated after a loss event, TCP-MEDX is labelled energy efficient, but it would require both the receiver and sender to modify the TCP header by inserting and processing timestamps.

b. TCP-DCR: Meanwhile in the presence of channel error loss, Bhandarkar et al. in [93] improved the performance of TCP by developing a TCP Delayed Congestion Response (TCP-DCR) at the sender. After the first duplicate ACK is received, this approach implements a congestion response algorithm that is delayed for a period equaling one RTT. This limits the TCP response to occur mostly during congestion losses

by allowing the mechanism at the link-level to recover the channel error lost packets. On the one hand, TCP-DCR would avoid congestion window reduction during this induced delay period by cancelling unnecessary retransmission during the occasion when a missing packet is recovered. On the other hand, the packet would be treated as a packet lost due to congestion if the packet is not recovered by the time the delay period expires.

The TCP SACK in NS-2 was modified by implementing the TCP-DCR, and then the modification was compared with the original in respect of their performance. When the primary cause for packet loss is congestion, TCP-DCR is capable of offering significantly better performance levels in the face of channel errors, but at the same time having minimal impact on performance. Nevertheless, TCP-R used a similar idea by delaying the trigger action when receiving the first ACK duplicate, but also allows packets that followed a different route to be accepted by the receiver. However This case does not take transmitted segments following different routes in TCP-DCR into account which may required more than one RTT to recover. Hence, further analysis is required regarding the choice of a single RTT.

2.3.1.3 Hybrid Approach

In contrast to the other approaches, the following proposals and possibly viable solutions have taken a mixed-type or hybrid approach and thus capable of detecting two packets loss types or more. The following sections shall elaborate further upon the previous works related to the field of interest.

a. ADTCP: In order to improve the accuracy of detecting packet loss, instead of relying on a single metric, the work by [1] combined multiple metrics, called TCP-friendly Transport Protocol for Ad hoc Networks (ADTCP). Various network states

or conditions can be identified, such as congestion, channel error, route change, and disconnection, by this pioneering approach that uses multi-metric end-to-end measurements.

In this method, every time the receiver accepts a packet, the receiver would calculate four metric values, namely Short-Term Throughput (STT), Inter-packet Delay Difference (IDD), Packet Loss Ratio (PLR), and Packet out-of-order Arrival (POR). Based on a technique called Relative Sample Density (RSD), this would define a value to be High or Low. The rules listed in Table 2.1 shows how the network state is estimated.

Table 2.1: Identification Rules of Network State (Adopted from[1])

#	**IDD and STT**	**POR**	**PLR**
Congestion	(High, Low)	*	
Route Change	Not (High, Low)	High	*
Channel Error	Not (High, Low)		High
Disconnection	Not (High, Low)	*	*
Normal	Default		

High: Top 30%; Low: Bottom 30%; '*': do not care

Next for every outgoing ACK packet, the receiver would pass information of the network state straight back to the sender. The latest of the most current state feedback can be obtained from these ACK packets, which the sender would maintain and proceed with normal TCP operations. This would continue up until the occurrence of a packet loss by either a third duplicate ACK or retransmission timeout. Control action will be taken by the sender in accordance with the estimation of the network state by the receiver. In any cases of congestion, the same congestion control action (such as conventional TCP) can be adopted by the sender. However, when the detection

of channel error occurs, a retransmission of the identified lost packet by the sender would commence without slowing the performance down. Meanwhile, in the event of a route change occurring, an estimated bandwidth of the new route can be calculated by the sender via setting current sending rate to the slow start threshold, and proceed with congestion avoidance initiation. Lastly, rather than invoking the mechanism for exponential back off, the current state will be frozen by the sender if and when disconnection is detected, and periodic probing will be carried out until a reconnection is established, while repeating the same steps in case of route changes occur.

The ADTCP method has become very famous and served as the basic platform for many later approaches, since it is TCP friendly and provides flexibility for backward compability. Furthermore, simulation results showed that ADTCP can achieve close to ELFN performance, with an improvement in the throughput from 100% to 800% vis-a-vis the TCP NewReno variant. Meanwhile in real testbed settings for weak channel case scenarios, ADTCP shows 30% throughput improvement over Reno and in mobility cases, 100% performance gain. However, due to its slightly high computational cost, a 5% decrease in ADTCP throughput was observed in clear channel the testbed settings. Nevertheless, a sample space is maintained by the ADTCP to set the metric value to be high or low. By using a weight, high value would be given to the recent sample as compared to the older sample, but sometimes a negative influence occurs from the old value. This is especially true for route change cases, in which it is more efficient for the sample space to be rebuilt for the elimination of old route influence.

b. TCP Enhanced Scheme for ADTCP: Previous researchers, like Li et al., used ADTCP as a base for their new enhanced scheme [94]. They attempted to reduce the calculation complexity in ADTCP by designing a much simpler and more effective one. This new proposed scheme is capable of detecting three network states, namely

congestion, channel error, and route change, based on receiver measurements for IDD, RTT, and Hop Limit (HOP). The variable IDD shows the actual buffer queue changes and RTT describes the length of the buffer queue. The congestion state can be identified by both IDD and RTT. In cases where the network state is not congested, then the HOP is used to detect whether a route change event had occurred or had not. Otherwise, if the TCP sender detected a packet loss and it is not due to congestion or route change, the assumption would be that the loss is due to wireless-related error. Meanwhile, RSD is used to decide the high or low value of each metric. However, the weight of each history record is characteristically uniform, and the sample space would be requested to be rebuilt after a route change.

In order to evaluate the performance of this new scheme, a comparison using NS-2 was conducted among NewReno, ELFN, ADTCP, and this particular proposed scheme. The simulation result showed a significantly better performance by the new scheme as compared to the NewReno method, while it achieve close to ADTCP and ELFN performances. However, the new scheme can detect congestion state more accurately than none-congestion state, thus making its performance degrades in cases of mobility and channel errors.

c. I-ADTCP: As a positive extension of previous work, Gajjar and Gupta proposed to improve ADTCP by applying Chen et al. [71] adaptive CWL strategy on top of it to ensure sufficient bandwidth utilization of the sender-receiver path [95]. The TCP's congestion window would be limited below the upper bound of bandwidth delay product of the path, which will avoid overloading the network. Furthermore, through a calculation of STT and IDD using RSD, an incipient congestion check can be performed. An avoidance of congestion build up by limiting the packets sent by the sender occurs during the incipient congestion through the reduction of Congestion Window Limit

(CWL) by half.

From the output results of simulations, at all levels of traffic intensity, I-ADTCP can be clearly observed to outperform ADTCP. Through the detection and reaction of incipient congestion, I-ADTCP attempts to constantly remain in the congestion avoidance phase at all times. Although better performance can be gained in most cases by comparing a small fixed CWL setting with setting the CWL in I-ADTCP, the small fixed CWL perhaps cannot be outperformed in every single scenario. This comes from the notion that certain assumptions under chain topology was the reason for the design and development of adaptive CWL. Nevertheless, the I-ADTCP is neither capable of handling the issue of sample space nor reducing the complexity of computation.

d. Imp-ADTCP: Next, Teng et al. in [96] proposed an improved scheme (Imp-ADTCP) for the original end-to-end original ADTCP scheme. The Imp_ADTCP method focuses on decreasing its aggressiveness and controlling the growth of the congestion window. The congestion widow growth factor is set by the this scheme which adaptively adjusts the threshold of congestion window in accordance with the forward path hop counts. If the number of hop counts is less or equal to eight, the growth factor is set to 0.01, otherwise it would be set to 0.1.

Simulation results showed that the Imp-ADTCP method can improve TCP throughput in dynamic networks, but there is nothing reported on the performance of Imp-ADTCP in static topology. Additionally, the value of the forward path hop count is obtained with the support of lower layer, thus making this approach not a purely layered-based method.

e. MME-TCP: Following a similar line of investigation, Kai et al. in [97] applied the

same ideas of ADTCP to the TCP NewReno variant, generally similar mechanisms and identical combinations of metrics were used to differentiate the type of packet loss. Thus, the corresponding action algorithms followed similar steps. However, the differences between MME-TCP and ADTCP are that MME-TCP is sender based and instead of using Short Term Throughput (STT) like in ADTCP, the MME-TCP method uses Short Term Goodput (STG). Also as a network state, wireless error is completely ignored by MME-TCP. Last but not least, in order to determine whether the state of the metrics is high or low, MME-TCP uses a mean deviation-based algorithm.

Through simulation, the output results revealed the TCP NewReno variant was outperformed by MME-TCP, by as much as 25% to 80% improvement in the throughput. However, the study did not report any performance result for noisy channel which contains high bit error rates. Also, each ACK is counted as one during the calculation of STG, but sometimes more than one packet arrives and it is acknowledged by a single ACK.

f. TCP WELCOME: On a different approach for enhancing TCP performance, TCP-WELCOME was designed by [8] as a sender-based solution for resolving errors by using the RTT variable to address issues in the MANET environment, such as congestion, wireless error, and route failure. The TCP-WELCOME approach distinguishs between packet loss causes and then triggers the most appropriate packet loss recovery according to the identified loss cause. Through the observation of RTT sample evolution history over the connection, this approach is capable of realizing the differing loss types and the trigger for data packet loss (i.e., RTO and 3DuplACK). On the one hand, wireless-related packet loss would be detected when RTT values are stable and 3DuplACK is detected, otherwise it would be considered as congestion type loss. On the other hand, route failure related loss would be detected when RTT values are sta-

ble and RTO is detected, otherwise the loss would be treated as congestion. The loss differentiation idea is illustrated by Figure 2.7.

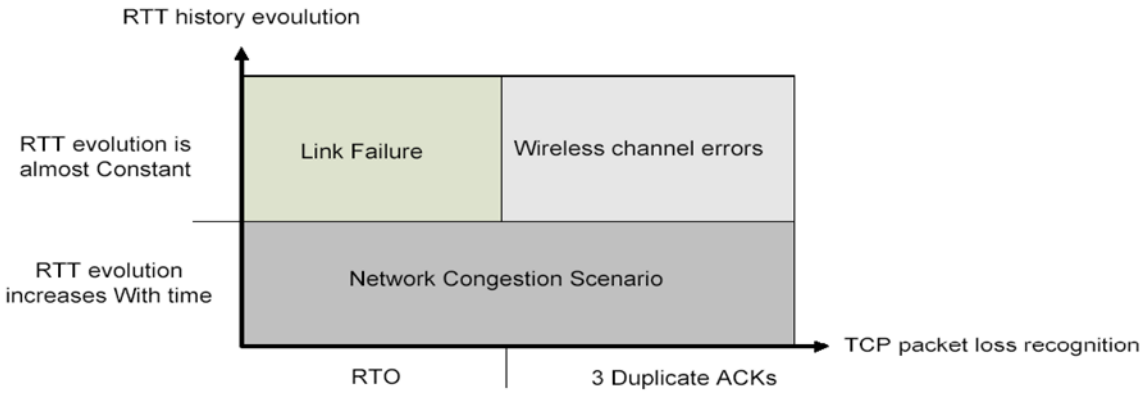


Figure 2.7: TCP WELCOME Loss Differentiation Algorithm (Adopted from [8])

The TCP-WELCOME reaction then focuses on data transmission rate and RTO calculation after using the proposed LDA to identify the data packet loss cause. No modification is needed to the standard TCP NewReno method in cases of congestion loss, as well as wireless error loss where the only remedy is the retransmission of the lost packet. However, the *cwnd* and RTO values would be updated, once route failure is detected, based on the new route characteristics of length, quality, and load, which is performed according to the following:

$$RTO_{new} = (RTT_{new}/RTT_{old})RTO_{old} \tag{2.5}$$

$$CWND_{new} = (RTT_{old}/RTT_{new})CWND_{old} \tag{2.6}$$

In terms of energy consumption and throughput, simulation output results showed that

SACK, TCP NewReno, Westwood, and Vegas methods were all outperformed by the TCP-WELCOME approach under various packet loss scenarios, such as interference, congestion, link failure, and signal loss. However, during the evaluation period in these simulations, the frequent route changes and network disconnection that may come into effect were not taken into account. Nevertheless, it was reported by Biaz and Vaidya in [98] that using RTT is not encouraging because of the large amount of noise associated with the measurement. Furthermore, the challenge in ad hoc network is that delay is no longer influenced by network queue length, but it is also susceptible to routing path oscillation, disconnection, MAC layer contention (very common). These make such measurement highly noisy.

2.3.1.4 Discussion

Several extensions and modifications have been proposed to mend the many challenges encountered by TCP in MANETs. Some of these solutions that handle more than one issue in MANET are presented in the following summary Table 2.2. This table aims to discuss further the main proposals to gain a better understanding and highlight the gap in the literature. In the table it is shown the based protocol, metrics used, approach, and implementation issues with the respective congestion control proposal.

In the column “Based on” off the table it indicates whether the proposed protocol will be based on TCP NewReno, Reno, SACK, or maybe other flavor. The “Used Metics” column indicates the type of performance metrics used to propose the new protocol. The column “Approach” identifies the issue(s) that been handled by the proposed protocol. Finally, the column “Implementation “ shows whether the approach can be implemented independent from the underlying layer(s) or the receiver side.

Table 2.2: Comparison among TCP Proposals

Solution		Based on	Used Metrics	Approach					Implementation							
				Adapt back-off	Dealing with route change/failure	Dealing with wireless error loss	Reducing Ack OverHead	Controlling congestion window size	Sender Based	Receiver Based	Intermediate Node independent	MAC independent	Routing independent	Ease of deployment		
TCP Sintok		NewReno	RTT+ELU	X	✓	X	X	✓	✓	X	✓	✓	✓	✓	IDD	inter packet delay
ADTCP		NewReno	STH+IDD+PL+OOO	X	✓	✓	X	X	✓	✓	✓	✓	✓	X	PL	packet loss
E-ADTCP		NewReno	IDD +RTT+hop limit	X	✓	✓	X	X	✓	✓	✓	✓	X	X	OOO	out of order
I-ADTCP		ADTCP	STH+IDD+PL+OOO+hopLimit	X	✓	✓	X	✓	✓	✓	✓	✓	X	X	STH	short thorughput
Imp-ADTCP		ADTCP	STH+IDD+PL+OOO	X	✓	✓	X	✓	✓	✓	✓	✓	X	X	SGT	goodput
MME-TCP		NewReno	SGT+PL+OOO	X	✓	X	X	X	✓	X	✓	✓	✓	O	CU	Channel Utilization
TCP-Welcor		NewReno	RTT	X	✓	✓	X	X	✓	X	✓	✓	✓	✓	CR	Contention Ratio
															ELU	Efficient Link Utilization

X	No
O	Intermidiate
✓	Yes

2.3.2 Proactive Approach

In this section, the recent proposals in ad hoc networks environment that have been highlighted were aimed to alleviate the spatial contention effects on the performance of TCP. They will be introduced, described, and analyzed. These proposed solutions can be divided based on the following aspects:

- the sender's perspective, and
- the receiver's perspective.

Some representative schemes are presented in the following sections, according to the above mentioned taxonomy.

2.3.2.1 Sender Perspective

The following viable proposed solutions are capable of reducing the excessive medium accesses due to a large congestion window by selecting the optimal maximum congestion window value or otherwise control the growth of the congestion window.

a. Congestion Window Limit (CWL): Previous research by Fu et al. in [28] revealed several interesting results when they studied the effect of multi-hop wireless channel on TCP congestion control and throughput. Their analysis using simulations revealed that there is a window size (W), at which TCP throughput is highest through improved spatial channel reuse. However, the average window size of standard TCP is much larger than (W) and does not operate close to (W). The implication here is that reduced spatial channel reuse would degrade throughput in TCP.

They did further studies in a h-hop chain of IEEE 802.11 wireless nodes and results

show that TCP's optimal throughput is achieved when its window size is $(h/4)$, where h is the length of the chain. If TCP window size is below this value, it under-utilizes the channel. Moreover, if it is larger, it will not further increase the channel utilization but it will reduce TCP throughput.

b. Adaptive Congestion Window Limit Strategy: Meanwhile, Chen et al. took up the observation and results by [28] in relation with TCP congestion window size [71]. They established a strategy for turning the CWL in TCP into identifying the bandwidth-delay product (BDP) of a path in MANET environment. They proved that the BDP of the path cannot exceed its round trip hop count (RTHC), independent of the respective MAC layer protocol being used. Furthermore where N is the path RTHC value, N/5 forms the optimized value of the upper bound based on IEEE 802.11 MAC layer protocol. They then dynamically adjust CWL in TCP according to the current path RTHC value, which is the proposed adaptive CWL setting strategy. This strategy is explained as follows:

Algorithm 2.3 Adaptive Congestion Window Limit Setting Strategy

Step 1. if (N <= 4) CWL = 2;

Step 2. else if (N <= 8) CWL = 1;

Step 3. else if (N <= 12) CWL = 2;

Step 4. else if (N <= 20) CWL = 3;

Step 5. else if (N <= 26) CWL = 4;

Step 6. else if (N <= 30) CWL = 5;

Their simulation, using NS-2, was conducted to demonstrate the performance gain of this CWL setting strategy. From the output of their results, a 8% to 16% improvement of TCP Reno performance was achieved using this simple strategy in comparison with the unbounded congestion window in TCP Reno. However, this scheme heavily de-

pends on being aware of the path length by the routing protocol, such as the DSR. Nevertheless, changing the maximum retransmission timeout of TCP into 2s in contrast to the 240s given in [99] might affect the results.

c. New Tuning Maximum Congestion Window: Kim et al. [70] proposed a new technique to overcome drawbacks of adaptive congestion window limit setting strategy presented in [71]. The basic idea in this method is that by collecting ECN and then send the value to the TCP sender with the advertised window, an estimation of the optimum window size can be performed by the TCP receiver. Whenever an ACK packet is received, the advertised window value is used to set the limit of congestion window at the TCP sender.

These researchers used the NS-2 simulator for evaluation of the proposed mechanism validity using random, grid, and chain topologies. In dynamic ad hoc networks, the simulation results showed that the proposed algorithm is capable of enhancing TCP performance by 10% to 60%. However, the implementation of fine-tuning the maximum congestion window requires ECN to be enabled and the adaptation of LRED to detect contention in the link layer. Nevertheless, the received information might be out dated.

d. TCP with Adaptive Pacing (TCP-AP): Work by Eirakabawy et al. had introduced a congestion control mechanism that is characteristically end-to-end [100]. The proposed approach is better known as TCP with Adaptive Pacing (TCP-AP), where within the current congestion window, it implements a scheduling that is rate-based. TCP-AP uses two factors to adapt its transmission rate, the former one is the delay of current out-of-interference estimation and latter one is the variation coefficient of recently measured RTT.

Both test bed as well as NS-2 were used to test and evaluate TCP-AP, and the results demonstrated that TCP with adaptive pacing can yield significant performance in terms of goodput and fairness with respect to TCP NewReno. However, TCP-AP is customized for use in multihop wireless networks only. Furthermore, TCP-AP achieves its best goodput when the number of hops is small and RTS/CTS is disabled. As the number of hops increases, the performance gap reduces. When RTC/CTS is enabled the performance of TCP-AP degrades significantly.

e. CWA-CD: Meanwhile for addressing the problem *of cwnd* overshooting, Zhang et al. proposed a TCP variant to alleviate this [77]. Firstly, the authors investigated the relation between BDP and congestion control algorithm, where they concluded that the actual BDP value in wired networks is much larger than multihop ad hoc networks. Using the original method to control the growth of congestion window would tend to increase *cwnd* way beyond the actual BDP, thus causing the entire network to overload, while operating with bad congestion. To address this issue, the said authors proposed to split RTT into two, which are contention RTT and congestion RTT. The former is the totaled contention delays that exists down the path, while the latter is made up of transfer delay that is characteristically end-to-end of all the links down the pathway. More specifically, congestion RTT does not contain contention delay, but comprises transmission delay, queuing delay, and processing delay. Therefore, the authors revealed that only congestion RTT can determine BDP and there is no relation between contention RTT and BDP. As a result, the overshooting problem of the TCP congestion window is caused by the inadequate use of contention RTT. Next, for the evaluation of the degree of link contentions, the authors defined Variance of Contention RTT per Hop (VCRH). Based on VCRH, the Congestion Window Adaptation through Contention Detection (CWA-CD) was then proposed as a viable solution. The *cwnd* value is adapted by the CWA-CD, based on VCRH in addition to RTO and

acknowledgment (ACK).

The researchers had claimed that the CWA-CD method is timely and accurate in limiting the congestion window size overshooting. Their simulation results showed that CWA-CD outperforms the conventional TCP within the static topology environment from the aspects of throughput and delay. However in dynamic scenarios, CWA-CD achieved similar throughput in comparison with the original TCP. The mobility factor is ignored in the CWA-CD approach that only focuses on link contentions, which in MANETs, is the main source of link unreliability. Therefore, a comprehensive investigation of link failure due to mobility should be considered to improve the performance of CWA-CD in dynamic topology.

f. FeW: This scheme that was proposed by Nahem et al. is a cross-layer approach [101]. They firstly investigated in IEEE 802.11 ad hoc networks, the effect of MAC contention and congestion window on the interaction between on-demand ad hoc routing protocol and TCP. The authors observed that the TCP's window mechanism is one of the main factors for throughput degradation. TCP generally sends at a high rate and increase *cwnd* aggressively, which would cause network congestion and channel contention to occur. These losses would have a negative impact on the routing protocol, because on-demand routing protocols would initiate a new route discovery process since they treat such loss as a route failure. Therefore, the authors proposed a Fractional Window increment (FeW) scheme to restrict the growth of the congestion window and thus reduce the aggressiveness of the TCP implementation by using a window update that is characteristically fractional and not exponential.

Also their simulation results showed dramatic improvement in the TCP FeW performance. However, they had not made it clear as to how far does the slower conges-

tion window growth may adversely effect short connections with only relatively small amounts of data. Furthermore depending on network conditions, TCP FeW performance will vary. Therefore for a given network condition, TCP FeW needs some fine-tuning to achieve the performance requirements.

g. ASP-FeW: Then, Wang et al. proposed a solution for FeW problems by introducing a new Adaptive Packet Size (ASP) [102]. The authors further analyzed this proposed solution and found that for transmission, it could not make appropriate use of its accurate predicted window. This method limits the amount of data transmission and can be considered to be too strict, so an Adaptive Packet Size (ASP) method on top of FeW is therefore recommended. While FeW contains fixed packet size, the ASP-FeW method fills the current predicted window by adapting packet size. The ASP-FeW method calculates the current packet size using the following equation:

$$packetsize\_ = ((cwnd\_XinitPacketsize\_)/cwnd\_) \tag{2.7}$$

where *packetsize_* is the current packet size, *cwnd_* represents the congestion window size, and *initPacketsize_* contains the initial packet size which has a fixed value.

Every time there is a change in the *cwnd*, this equation is used to compute the new packet size by the TCP sender. This is even more pertinent when the retransmission timeout occurs. The TCP sender enters the slow-start phase by resetting the *cwnd* to 1. It then repacks in its buffer the data with the given initial packet size and retransmits it. There is then no need to repack the lost data packet, when TCP enters fast retransmit when three duplicate ACKs occur, so a re-transmission of packets occurs.

h. TCPCC: Zhang et al. studied the effect of conventional window mechanism on TCP performance [76], and highlighted that network congestion is caused by medium contentions, and in the case of over-injection of current TCP window mechanism, it would result in severe contentions. Then the authors introduced Contention Ratio (CR) and Channel Utilization (CU) as the two metrics for network status characterization. In relation to these new metric measurements, a new TCP transmission rate control mechanism (TCPCC) was proposed, where the CU and CR are accordingly estimated with each intermediate node collecting information about the network status. The returned ACK will contain values of CU and CR. Then, the TCP sender is able to make an adjustment in its transmission rate while considering the feedback information, as shown in the pseudo code below:

Algorithm 2.4 TCPCC Mechanism Window Adaptation Scheme

Step 1. if (CU < CU*max* and CR > CR*min*) {

Step 2. cwnd = cwnd - (CU*max* - UC)*cwnd

Step 3. } else {

Step 4. cwnd increase as TCP does

Step 5. CU*max* = CU

Step 6. CR*min* = CR }

The results of their simulations revealed that in terms of end-to-end delay and throughput, conventional TCP is outperformed by TCPCC in static topology. However, there is still the obvious problem in this mechanism when it comes to wireless networks, because the status of bottleneck link defines these two parameters without considering the whole path. Furthermore, evaluating TCPCC over mobile ad hoc networks and studying its performance over channel with different BER is required.

2.3.2.2 Receiver Perspective

These viable solutions can reduce spatial contention by introducing fewer ACKs, in other words, by sending cumulative ACK instead of sending separate ACKs for each packet received successfully.

a. Dynamic DelAck: Altman et al. had extended the idea beyond two consecutive ACKs in standard DACK's combination [99]. They proposed a Dynamic DelACK to reduce channel contention among ACKs and data, as a solution on the receiver-side of the same TCP connection [103]. The Altman scheme serves as a basis for many later approaches. The idea is to generate an ACK for every d data packets or after a certain, fixed timeout. The authors observed in NS-2 simulation, significant improvements in performance for $(d = 2)$, and increased further for d equaling (3) and (4). However when TCP operates at a small window size, this value may be problematic. Therefore, they proposed a dynamic delayed ACK where d increases with the number of the packet sequence, up to the maximum level of $(d = 4)$.

Their simulation focused only on static multi-hop networks, but demonstrated that performance gain is possible with this approach. Nevertheless, the receiver initially delays four packets except for the startup, where it neither fills in the gap in the receiver's buffer nor reacts to the case of out-of-order and uses fixed interval of $100ms$ for timing out. In other words, in conditions of medium level change, there is a lack of adaptability. In addition, the available bandwidth of a connection would be depended upon by the congestion window size, which in turn is depended upon by the value of d. In relation to the value of d, since information about the network status is not provided by the sequence number itself, the sequence number does not effectively reduce channel contention of the intra-flow nature [13]. Furthermore every time a delayed ACK is received by a source, network congestion may occur due to the burst of TCP

segments that may be injected into the network [58].

b. TCP-ADA: Meanwhile, Singh and Kankipati [78] studied in the mobile ad hoc network environment the effect of the ACK number generated for a single window on TCP throughput. When one ACK acknowledges a full window, maximum throughput can be achieved as shown through mathematical analysis. Due to this analysis, they proposed TCP with Adaptive Delayed Acknowledgement (TCP-ADA) as a receiver-side solution, which is an attempt to reduce the number of ACKs to one per window. The ACK generation time is adaptively determined by TCP-ADA, which is in contrast to Dynamic DelAck that defers the ACK generation until a fixed number of data packets have arrived. In the event of every data packet arrival, before transmitting the cumulative ACK, the receiver pauses for a duration equal to wait $factorXaverage$ inter-arrival time. If another data packet is received during that pause period, the deferment period can be calculated as per previous ACK. The assumption that all DATA packets that are sent in the congestion window have been received is made by TCP-ADA once the wait period expires, and for all received DATA, it would send the cumulative ACK.

In a comparison exercise using NS-2, with the wait factor equaling 1.2 and averaging constant equaling 0.8, the authors compared the differences betweenTCP and TCP delAck to TCP-ADA. It was observed over TCP that TCP DelAck exhibited less throughput improvement than TCP-ADA. This is caused by the adaptive delay during the transmission of cumulative acknowledgement. However, like Dynamic DelAck, the scheme was not able to explain out-of-order packet reception and packet loss. Furthermore, the sender woul be idle and may not send any new packets during the deferment period, and the case could be even worse if the ACK got lost.

c. TCP-DAA: Following the same line, De Oliveira and Braun proposed the receiver-

side method to overcome some drawbacks in the Altman scheme [103]. They combined the premise of higher numbers of delayed ACKs (like TCP-ADA) together with the Altman et al. recommendation in [47] (i.e., direct reaction to filling in a gap or packet out-of-order) to design the new scheme called Dynamic Adaptive Acknowledgment (TCP-DAA) [104] . Furthermore, TCP-DAA is capable of adjusting itself to the channel conditions by timeout interval computation for the receiver with the incoming packet inter-arrival time serving as the basis.

In TCP-DAA, when the wireless channel is in good condition, the receiver combines up to four ACK packets where the four packet limit is restricted by the congestion window limit of the sender. The delay management is performed using the delay window (*dwin*) that restricts the number of delayed ACKs. An ACK is immediately send and a reduction of *dwin* by the size of two packets is performed whenever the receiver timer expires or when it gets a packet that is either filling a gap or out-of-order in the receiver's buffer. There are two adjustments at the sender to minimize the unnecessary retransmission, which are 1) a fivefold increase of the regular RTO for compensation of the four combined regular ACK maximum and, 2) the duplicate ACK number for fast retransmit trigger is reduced to two.

Simulation evaluation showed that TCP-DAA can outperform not only conventional TCP but also similar techniques like large Delay ACK (LDA) in terms of throughput and energy consumption. Yet in scenarios with no problem of hidden nodes, the results reported that TCP-DAA does not seem to be very friendly. Nevertheless, setting congestion window limit to four seems to be appropriate in wireless environment with up to 10 hops, as stated by the authors, but not appropriate for hybrid networks where wired systems and wireless systems coexist together with the existence of high bandwidth delay product. Finally, one issue that may arise here is what if the TCP sender

congestion window is not limited to four.

d. Delayed Cumulative Ack (TCP-DCA): The authors [73] proposed an adaptive delayed ACK method based on the TCP path hop number. Unlike TCP-DAA, there is no limit value expected for the *cwnd* sender. However, the sender has to send its current *cwnd* in the TCP header to the receiver. This scheme is basically designed for short paths where the problem of interference is at a minimum, and to maximally increase TCP throughput, the receiver would delay its ACK as long as possible. In cases of long paths, for achieving optimal performance of TCP and avoid high packet loss, the receiver would use a suitable delay window size (*dwin*) restriction. The authors performed extensive simulation for identifying the proper value of delay window based on the path length. Table 2.3 lists these values.

Table 2.3: Delay Window at TCP-DCA Receiver

Parameter Path Length (h)	**Delay Window Limit**
$h \leq 3$	Congestion Window
$3 < h \leq 9$	5
$h \geq 10$	3

Similar to TCP-DAA, the receiver will generate one ACK to acknowledge one (*dwin*) in case the packets arrive in order, otherwise, the ACK will immediately be generated. In addition, the receiver keeps the fall-back delay ACK timer updated, which the estimated time for data packets to arrive (*dwin*).

TCP-DCA superior performance was revealed, in static networks producing up to 30% gain against the standard TCP and 20% better performance over different routing protocols in MANETs. However, the TCP-DCA receiver requires the information of the sender's congestion window size using information contained in the data packet header, namely the advertised window field. This is an apparent drawback to this approach. Moreover, the rate of congestion window increase is slowed down since the receiver does not send ACKs as often as in standard TCP, and that could be fine but there is nothing reported about RTT and RTO, so how can the sender appropriately estimate these values?

2.4 Theories Pertinent to Congestion Control

There are two major theories adopted in this research; Detection Theory and Communication Accommodation Theory. The following subsections will introduce these theories in the context of this research.

2.4.1 Detection Theory

Detection theory, or signal detection theory, is a means to quantify the ability to distinguish between information-bearing patterns (called signal in machines or stimulus in humans) and random patterns that distract from the information (called noise). It provides a general framework to study decisions that are made in ambiguous or uncertain situations [105]. According to the theory, there are a number of determiners of how a detecting system will detect a signal, and where its threshold levels will be. To apply detection theory to a data set where stimuli were either present or absent, and the observer categorized each trial as having the stimulus present or absent, the trials are sorted into one of four categories, as shown in the Table 2.4:

Table 2.4: Detection Table

	Respond "Absent"	**Respond "Present"**
Stimulus Present	Miss	Hit
Stimulus Absent	Correct Rejection	False Alarm

"Detection Theory is an introduction to one of the most important tools for the analysis of data where choices must be made and performance is not perfect" [106]. Interestingly Detection theory has applications in many fields such as telecommunications, biomedicine, quality control, radar and sonar, and psychology. Detection theory will provide the base for the proposed Loss Detection Mechanism (LDM) to be presented in Chapter 4.

2.4.2 Communication Accommodation Theory (CAT)

Communication Accommodation Theory (CAT) is a theory of human communication developed by Howard Giles [107, 108]. CAT is an interesting theory revolving largely around adjustment; it explains how individuals adjust to, or accommodate to, the speaking style, vocal patterns, and gestures of each other to gain greater communication efficiency [109]. CAT is a robust paradigm in the sense that it is able to attend to (1) social consequences, (2) ideological and macro social factors, (3) intergroup variables and processes, (4) discursive practices in naturalistic settings, and (5) individual life span and group language shifts. Furthermore, CAT can be applied in almost any situation that involves communication including computer communication [110].

The Scopus database was searched using the "Communication Accommodation The-

ory" keyword. There are a total (68) documents in Scopus database that include journal articles (55), review (8), article in press (2), short survey (2), and conference paper(1). The search was limited only to *articles* between years 1994 and 2012.

The results of the previous search showed that most of these articles came from different subject areas, with highest subject area being "SOCIAL SCIENCES". Furthermore, it was interesting to reveal that CAT has attracted researchers' attention from multi-disciplinary areas, such as economics, health care, and computer science; and has the potential to span across a wide range of new domains for future applications. Table 2.5 lists the top ten subject areas that these articles were published in.

Table 2.5: List of Top Ten Subject Areas of CAT Articles

#	**Subject Areas**	**Record Count**	**% number articles**
1	Social Sciences	42	76.363%
2	Psychology	23	41.818%
3	Arts and Humanities	15	27.272%
4	Medicine	10	18.181%
5	Business, Management and Accounting	7	12.727%
6	Biochemistry, Genetics and Molecular Biology	3	5.454%
7	Economics, Econometrics and Finance	3	5.454%
8	Computer Science [111, 112]	2	3.636%
9	Health Professions	2	3.636%
10	Nursing	2	3.636%

2.5 Summary

This chapter provides the detailed background on issues that are covered in this thesis. The background materials on loss detection and contention avoidance techniques were reviewed and critically analyzed for content and significance. Furthermore, it introduced the TCP congestion control performance model. This model forms the basis of the theoretical framework for this research. It reveals the necessity for a reliable loss detection mechanism that can effectively detect the cause of packet loss in mobile ad hoc networks. Additionally, it highlights the obligation for adapting TCP sending rate to avoid contention in the shared medium and support spatial channel reuse.

In the next Chapter, the research methodology for achieving the objectives of this research and evaluating TCP Sintok will be presented.

CHAPTER THREE
RESEARCH METHODOLOGY

This thesis aims at designing an new Transmission Control Protocol (TCP Sintok) for ad hoc networks named, specifically its loss detection and contention avoidance mechanisms. Furthermore, verifying and validating the proposed mechanisms as well as evaluating the performance of TCP Sintok are important tasks to be accomplished as stated in Chapter One. In order to achieve these objectives, it requires a rigorous methodology to follow, and this is the focus point of this chapter. For this purpose, this research employs the Design Research Methodology (DRM) and introduces its main stages according to the phenomena of this research. The chapter starts with the overall research approach as shown in Section 3.1. Section 3.2 introduces the first stage of DRM named Research Clarification (RC). It discusses the aims of RC stage, methods to support this stage, and main deliverables. Section 3.3 describes the second stage called Descriptive Study-I (DS-I). It discusses steps to obtain sufficient understanding of the current situation, designs a reference model, and proposes a conceptual model. Section 3.4 highlights the methods adopted in designing the proposed loss detection and contention avoidance mechanisms in line with the third stage of DRM, named Prescriptive Study (PS). Methods for performance evaluation and its metrics are discussed in Section 3.5 toward the end of the chapter, where a chapter summary rounds of this chapter.

3.1 Research Approach

The prime aim of this research is to design loss detection and contention avoidance mechanisms for Transmission Control Protocol, called TCP Sintok, with an intention to change existing situations into preferred ones (i.e., to improve TCP performance in ad hoc networks); this demands a careful mapping between understanding of the

traditional protocol and the development of a new one leading to an effective and efficient solution [113]. These requirements are fitted with the design research definition as proposed by Blessing [114], where "design research integrates the development of understanding and the development of [protocol]".

These aspects complement each other in order to produce an efficient and effective solution, which would result in a better/higher performance protocol. Blessing stated that design research must be scientific in acquiring valid results both in the theoretical and practical sense; and due to its unique features it needs a special methodology. Hence, Blessing proposed the Design Research Methodology (DRM).

DRM is an approach, guideline, and a set of supporting methods to be used as a framework for performing design research, where “it helps making design research more rigorous, effective and efficient and its outcomes academically and practically more worthwhile” [114]. Due to these attractive features, DRM has been adopted for conducting this research. DRM specific objectives are listed as follows:

- to provide a framework for individual researchers to conduct design research;
- to help identify research areas that are academically and practically worthwhile and realistic;
- to allow a variety of research methods in addition to help in selecting suitable methods and combinations of methods; and
- to providing guidelines for rigorous research and systematic planning.

DRM can be divided into four different stages, namely Research Clarification (RC), Descriptive Study-I (DS-I), Prescriptive Study (PS), and Descriptive Study-II (DS-II) stages. In the following sections, a brief explanation of DRM stages from the

perspective of this research area is presented. Additionally, the main methods and deliverables of each stage will be highlighted (for more details, refer to Blessing et al. [114]).

Figure 3.1 illustrates the DRM framework where it shows the links between design research methodology (DRM) stages, the methods used in each stage, and the main deliverables. The light arrows between the stages illustrate the main process flow, while the bold arrows to/from each stage demonstrate methods used and deliverables of that particular stage.

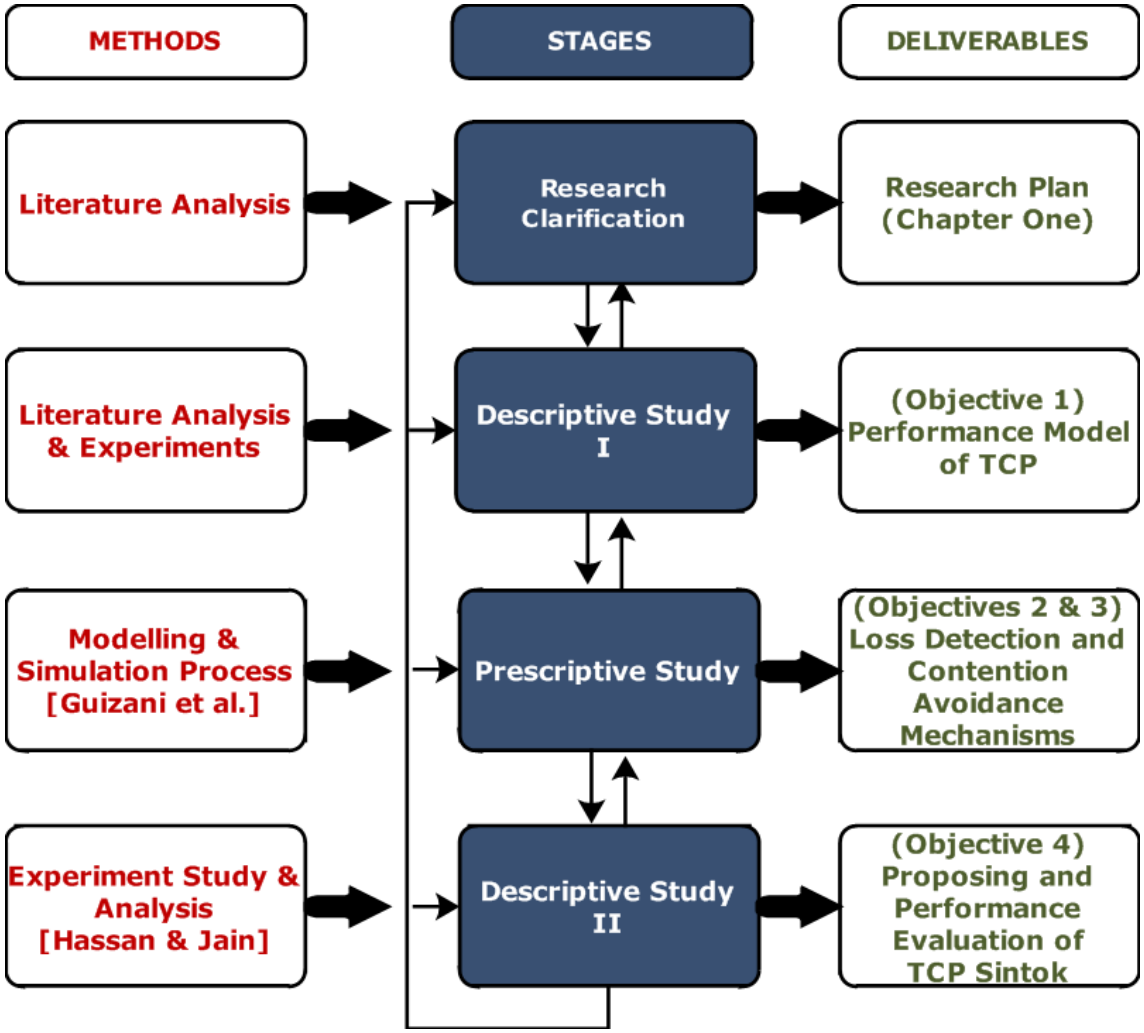


Figure 3.1: Research Approach

3.2 Research Clarification (RC)

This research started with the first stage of DRM called RC, which was used to obtain a precise understanding, as well as a challenging but realistic overall research plan. RC consists of six iterative steps, as shown in Figure 3.2:

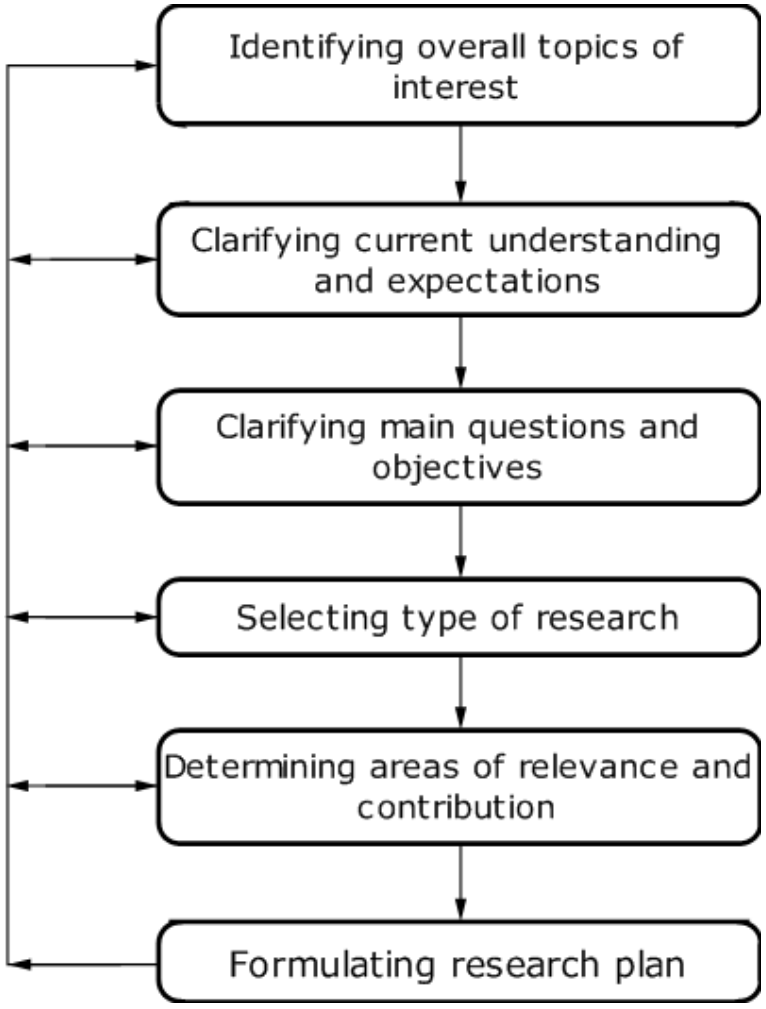


Figure 3.2: Main Steps in the Research Clarification Stage

In general, the deliverables of the RC stage is Chapter One. More specifically, the deliverable is the overall research plan which includes the following points:

- research focus and motivation,
- research problem and research questions,

- relevant areas to be consulted,
- approach (type of research, scope, main stages, and methods), and last but not least
- area of contribution and deliverables.

3.3 Descriptive Study-I (DS-I)

Upon the completion of the RC stage which formulated the overall research plan, the research moves to the second stage, known as DS-I. The DS-I was used to gain a deep understanding of the current situation and it involves a critical review of the literature in the research area as well as empirical studies. During the course of this research, a detailed review of the current proposals was discussed [68, 115] and many empirical studies were critically evaluated to gain deep understanding of the existing solutions and directions [116, 117]. The DS-I consists of five steps with many iterations, as illustrated in Figure 3.3, where every step aims to increase the understanding and may give rise to further empirical studies or literature reviews leading to refining and updating the performance and conceptual models.

The deliverables of the DS-I stage are:

- critical review of previous works as well as TCP Performance Model as presented in Chapter Two, and
- TCP Sintok Conceptual Model.

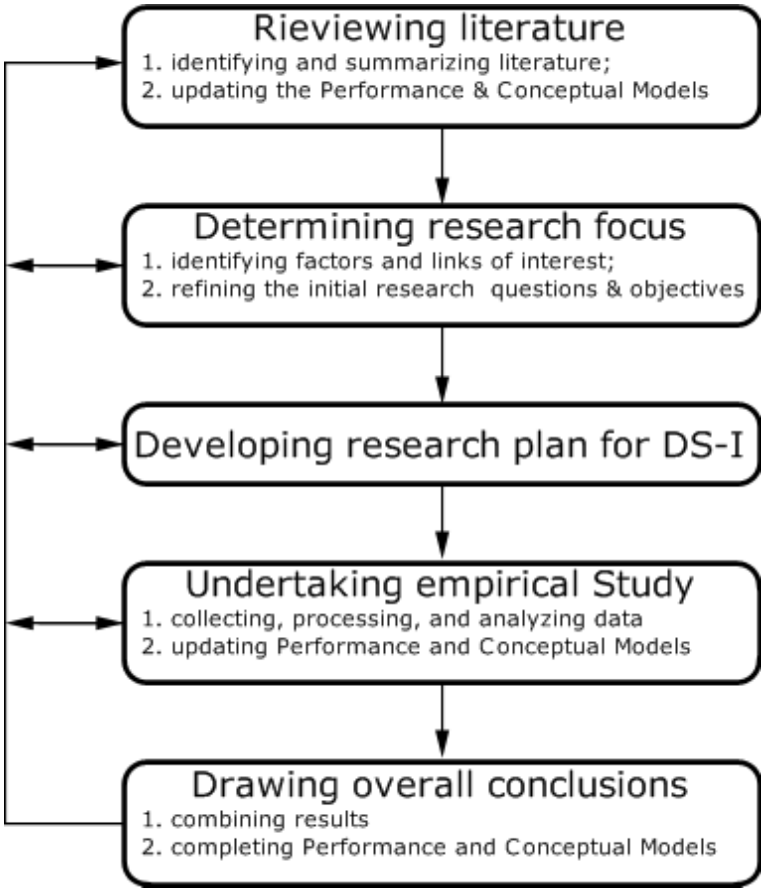


Figure 3.3: Main Steps in the Descriptive Study-I Stage

3.3.1 Conceptual Model of TCP Sintok

In order to improve TCP performance in ad hoc networks while basing on the proposed performance model and the critical review of much related works, congestion control was determined as the key factor to be addressed. Furthermore, the overall ultimate goal for this research was on the reduction and/or elimination route failure and route change loss influences in TCP and adjusting the congestion window size growth in order to avoid network contention. As a consequence of addressing TCP congestion control, a conceptual model of TCP Sintok was designed to describe the expected desired and improved situation using the proposed contention avoidance and loss detection mechanisms. The proposed conceptual model is shown in Figure 3.4

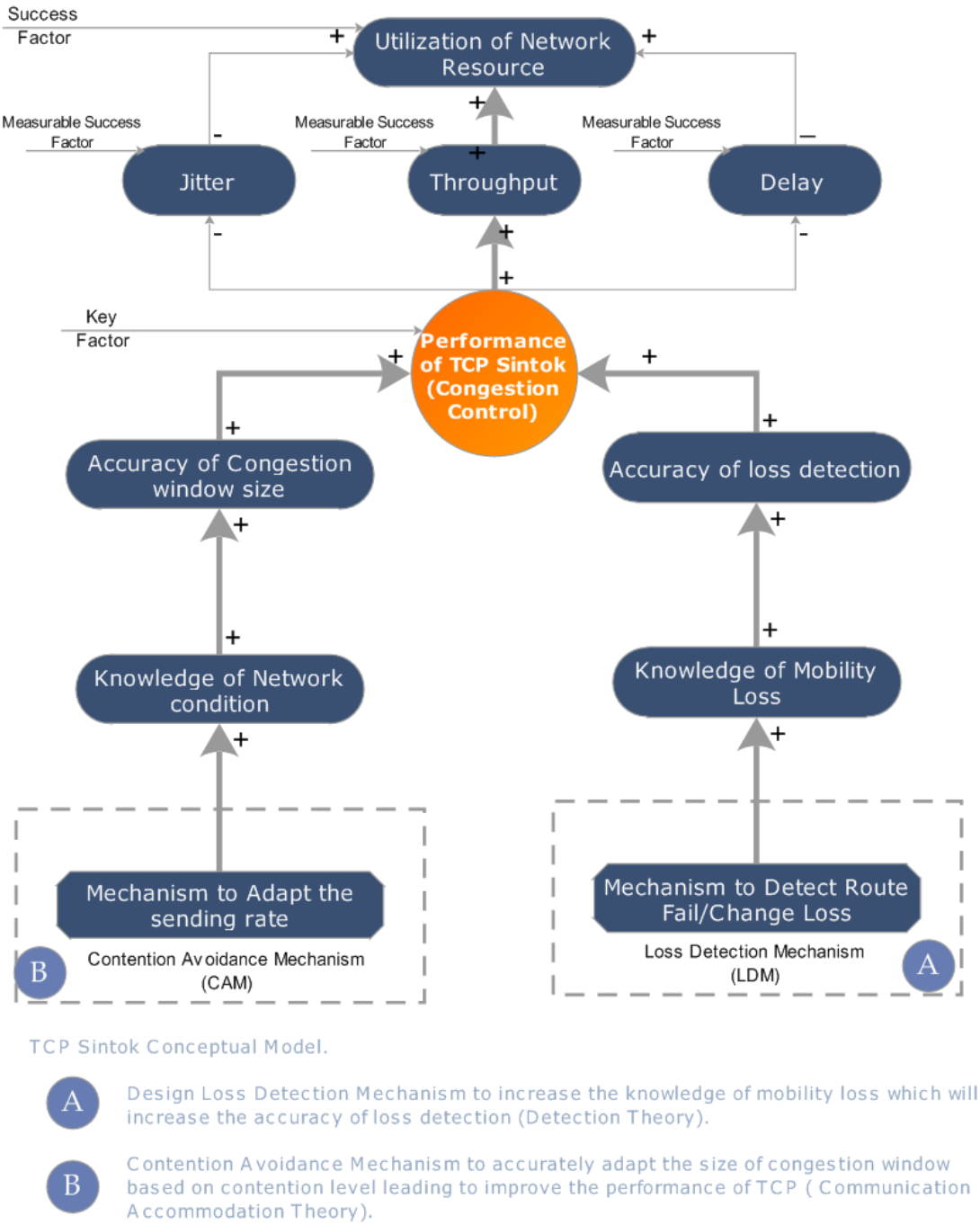


Figure 3.4: Conceptual Model of TCP Sintok

3.4 Prescriptive Study (PS)

The PS is the main stage in DRM, as it includes the design of the proposed mechanisms. For the purpose of this research, network modeling and simulation process proposed by Guizani et al. was followed [118]. Figure 3.5 shows a block diagram

describing the main steps of the prescriptive stage according to this research phenomena. The first block represents specifications of the proposed mechanism (protocol) to be modeled. The second and third blocks constitute modeling development which includes problem analysis, goals determination, and study of related theory. Furthermore, it frequently involves making assumptions and introducing a simplification to reduce the model's complexity. Blocks 4 and 5 illustrate model implementation and it depends very much on the choice of the simulation environment. Finally, validation will be covered in greater detail in the following subsection.

The deliverables of the PS stage are:

- Chapters Four and Five (Objectives 2 and 3) and part of Chapter Six (the development of TCP Sintok),
- design and implementation of the proposed mechanisms, and
- validation of the proposed mechanisms.

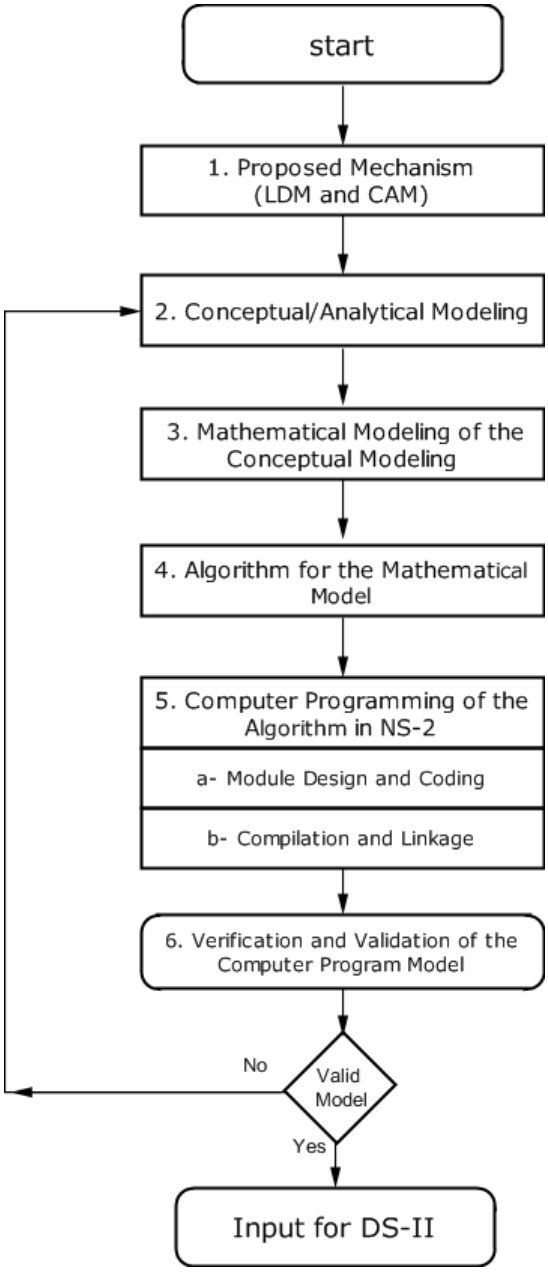


Figure 3.5: Main Steps in the Prescriptive Study Stage

3.4.1 Verification and Validation

Model verification is a determination that the model is transformed from one form to another with sufficient accuracy being maintained [119]. In other words, model verification evaluates the accuracy of transforming a model representation from a flowchart or pseudo code form into an executable computer program. In this research, all proposed mechanisms were transformed into C++ code since NS-2 requires C++ as the

base programming language. Furthermore, all mechanisms (models) must be verified to ensure that they have been coded correctly and free of bugs or errors [120]. Therefore, Eclipse C/C++ Development Tools (CDT) that runs on top of the Eclipse Platform was used for this purpose. The Eclipse IDE for C/C++ Developers provides advanced functionality for C/C++ developers, including an editor, debugger, launcher, parser, and makefile generator, as shown in Figure 3.6.

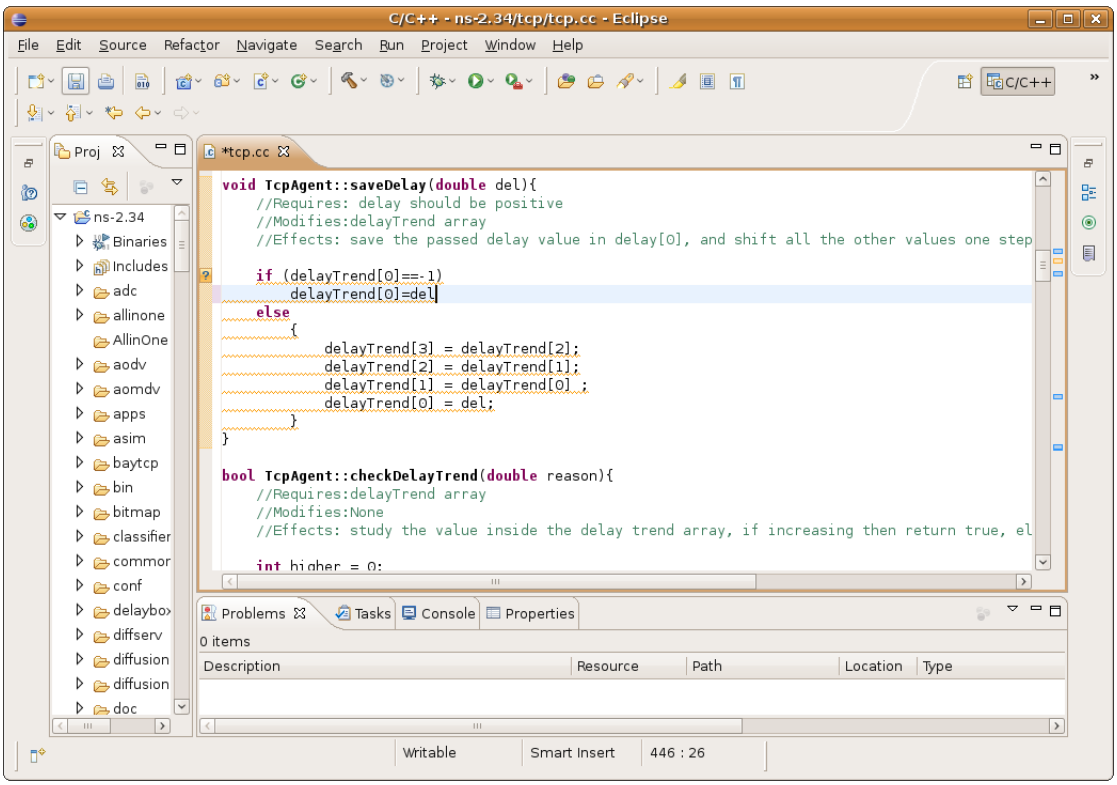


Figure 3.6: Eclipse C/C++ Development Tools

Furthermore, CDT's Code Analysis (Codan) integrated in Eclipse is capable of assisting the researcher by highlighting and indicating possible syntax errors to the researcher as he or she types the code, find bugs, and other issues as problems, warnings, or not at all. Codan works by scanning C++ code and checks for potential programming problems as well as syntax and sematic errors, as shown in Figure 3.7. CDT's Code Analysis is a very important feature and was used to ensure that,

- the mechanism is programmed and implemented correctly, and
- the mechanism does not contain any errors or bugs.

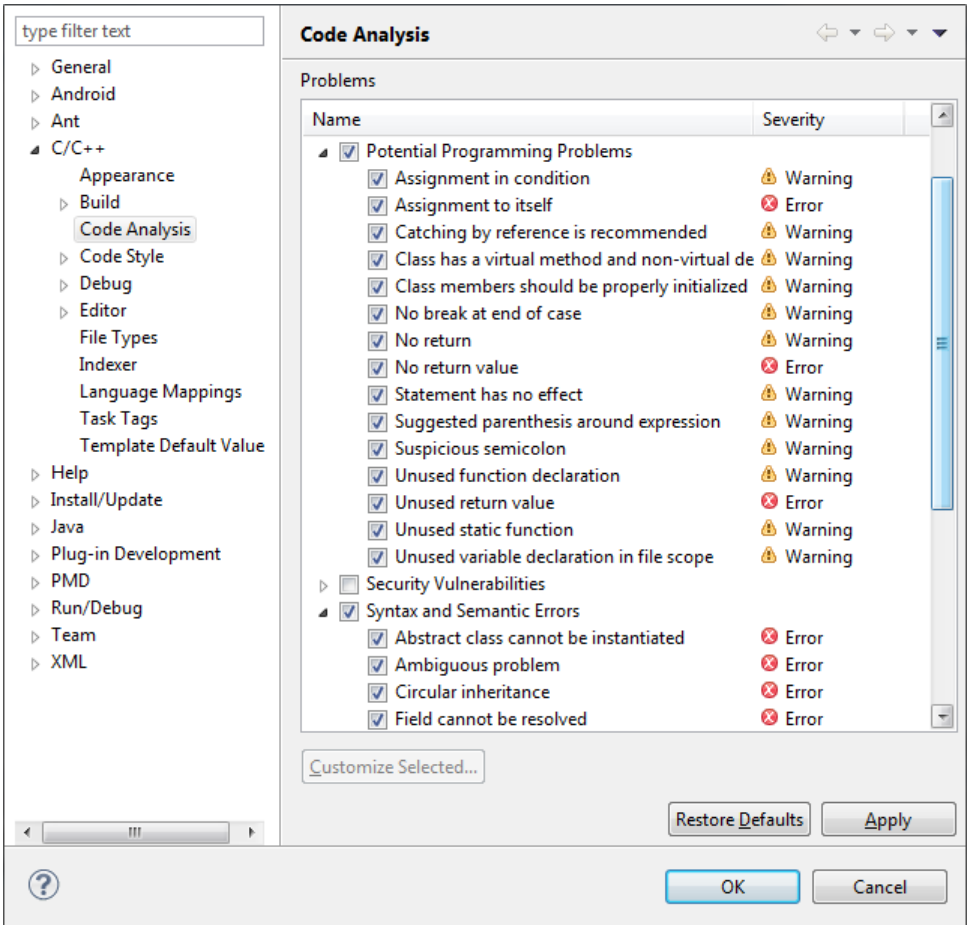


Figure 3.7: Code Analysis in Eclipse

Model Validation is usually defined as the "substantiation that the model, within its domain of applicability, behaves with satisfactory consistent with the study objectives" [121]. Model validation is also defined as the "substantiation that [is] a computerized model within its domain of applicability possesses a satisfactory range of accuracy" [122]. Validation needs to be performed to ensure that the mechanism meets its intended requirements in terms of the methods employed and the results obtained, which is all part of the process to build the correct model.

Meanwhile, Balci classified the verification and validation techniques into four main categories, namely Informal, Static, Dynamic, and Formal [119]. The dynamic technique is commonly used in model verifying and validating and thus it is the primary approach applied in this research. In particular, the comparison testing is used where the mechanism is executed under different conditions and the values obtained are used to determine whether the mechanism and its implementations are correct [123]. Furthermore, this would ensure that the mechanism operates in accordance with the specifications; for example, (1) model results are compared with the actual simulation output, (2) simulation model results are compared to results of analytic model, or (3) simulation model results are compared to results of real measurement.

The dynamic technique generally requires model execution and is intended to validate the model according to its execution behavior. Furthermore, dynamic techniques require model instrumentation and there are three steps that are followed in order to apply the dynamic technique for validating the proposed mechanisms in this research. Firstly, the executable model is instrumented, which refers to the insertion of additional code into the executable model for collecting information related to the model behavior during the actual execution. Secondly, the instrumented model is executed, and lastly the model output is analyzed and compared to results of other valid models (i.e., benchmarking).

As stated in Chapter One, this research concentrated on detecting the cause of packet loss in IEEE 802.11 ad hoc networks, as such the research has proposed an analytical model of end-to-end delay and used simulation to verify and validate the results. Then, the proposed model was used to design a loss detection mechanism which was also verified and validated by comparing the proposed mechanism results to the actual results obtained from the simulation. On the other hand, the research had focused on

adapting the sending rate of TCP to avoid contention and thus maximize the network resource utilization. As such, the contention avoidance model has been implemented in a simulation environment, and real measurement results were obtained from other studies that can be used to validate/benchmark the simulation results.

3.5 Descriptive Study-II (DS-II)

The DS-II focused on the evaluation of the designed mechanisms and protocol. Performance evaluation is a very important step in evaluating any research. The evaluation of TCP performance can be conducted using three possible traditional methods [124, 125, 126], namely analytical modeling, simulation, and measurement.

3.5.1 Evaluation Approach Consideration

Selecting the evaluation technique is a very crucial step in all performance evaluation projects [2]. Table 3.1 outlines the strength and weaknesses of performance evaluation techniques as stated in [2, 127, 128]. From Table 3.1 and according to [129, 130], it is very clear that simulation would be the most desired approach for the performance evaluation of this research.

Table 3.1: Comparison of Performance Evaluation Techniques (Adopted from [2])

Criteria	Analytical modeling	Simulation	Measurement
Time required	Low	Medium	High
Accuracy	Low	Moderate	High
Tools	Analysts	Computer languages	Instrumentation
Trade-off evaluation	Easy	Moderate	Difficult
Cost	Small	Medium	High

3.5.1.1 Analytical Modeling

Analytical (also called mathematical) modeling is a set of equations formulated using mathematical symbolism to describing the performance of an actual system [125]. A mathematical model can be investigated using computer programming, which translates the operations by using functional relationships within the system. The results of the mathematical model can be represented using a graphical representation drawn from the output of the running program. Users can adjust the conditions of the system by varying the input or parameters of the program. The technique is best suited in studying the system behavior of the very rare or unsafe in real life. This might help to better understand the initial view of a system before moving to the implementation process. This technique is often used to study simple systems; where an analytical model will be built and validated to explore and solve a specific problem in a system. Once the system complexity increases, this technique would require simplification and assumptions to focus on certain aspects of the system and fix the rest. According to Jain in [2], mathematical modeling has several benefits and advantages such as low cost, less time required, and easy in trade-off evaluation. However, analytical model-

ing has low accuracy as compared to other techniques in performance evaluation.

3.5.1.2 Measurement

Measurement based performance evaluation can be conducted using test-beds or implementation of an actual network. Also, this approach provides very accurate results, but it is costly due to the specialized equipments needed [131]. Furthermore, constructing a real ad hoc network test bed for a given scenario is typically expensive and remains limited in terms of working scenarios, mobility models and so on. Additionally, measurements are generally not repeatable.

3.5.1.3 Simulation

Simulation technique is widely used in representing dynamic responses and behaviors of real systems. It is a computer-based system model or generated using computer programming. Furthermore, simulation is a more flexible tool for studying the performance of various protocols. It enables analysis the protocol performance in scalable, controllable, and repeatable environments [129, 132]. For this reason, simulation was applied extensively for performance evaluation and validation of ad hoc networks protocols [29, 120, 133] and it is the main evaluation method in this research. The following are some of the advantages of using a network simulator to study TCP performance:

- simulators need a single computer to run the simulation experiments and analyze the obtained results,
- simulators allow network researchers to investigate a wide range of scenarios in a short period,
- complex topologies can be easily created via simulation environment, whereas such topologies would difficult to replicate in a testbed environment, and

- simulators provide access to data about all traffic transmitted during the simulation experiment.

3.5.2 Evaluation Environment

Simulation was chosen for the purpose of performance evaluation in this research and was used successfully for performance analysis, assessing network traffic loading, tuning of resources, and the prediction and optimization of the performance of protocols and architectures. There are many discrete event network simulators available for ad hoc networks researchers, such as NS-2 [134], NS3 [135], GloMoSim [136], J-Sim [137], OPNET [138, 139], Qualnet [140], and OMNeT++ [141]. Each one has its own unique advantages and disadvantages, and in order to get an overview about any network simulator, many studies were conducted to compare the performance of several simulators.

Koksal wrote in [142] that NS-2 and OMNeT++ would be the best choices for network researchers. While NS-2 is the most popular simulator for academic research, it is generally criticized for its complicated architecture. However, its large widespread use by the network research community makes up for it since there are many people helping each other with their problems through the use of mailing lists and forums. Meanwhile, OMNeT++ is gaining popularity in academic and industrial worlds. Unlike NS-2, OMNeT++ has a well-designed simulation engine and supports hierarchical modeling, so it is logically better for development. Also, OMNeT++'s powerful GUI gives it a certain edge. However, OMNeT++ lacks the abundance of external models and user base NS-2 has. OPNET Modeler is also a good, complete solution; but, it caters more to industrial researchers who need an extensive set of built-in reliable models for constructing credible simulations in a quick way, rather than academic researchers.

Table 3.2 provides the conclusions drawn in [3] from its comparison between NS-2, OPNET, and J-Sim simulators that are perhaps considered the leading ones.

Lucio et al. [143] stated that NS-2 and OPNET Modeler reported very similar results, but the freeware version of NS-2 makes it more attractive to a researcher. From the technical point of view, Lucio et al. showed similar performance for both simulators. Additionally, Garrido et al. [144] presented a comparison restricted to NS-2 vs OPNET. They concluded based on the simulation results for the different MANET scenarios that the trend of all the metrics in both simulators were rather consistent, although in certain experiments absolute values were quite different.

Xian at el. [145] compared the performance of OMNET++ with NS-2 and with OPNET in the area of wireless sensor networks. The paper showed that OMNET++ has better performance than both NS-2 and OPNET in terms of simulation time and memory. However, the paper does not address the reliability of the results obtained by the different simulators.

According to Kurkowski in [120], simulation is an often used tool to analyze research output in MANETs; 114 out of the 151 *MobiHoc* published papers (75.5%) used simulation to test their research. Furthermore, NS-2 is the most used simulator in MANET research; 35 of the 80 simulation papers that state the simulator used in their simulation study used NS-2 (43.8%). Based on the previous discussions and the attractive features of NS-2, it has been chosen as the evaluation environment for the purpose of this research. More details about NS-2 are provided in the following subsection.

Table 3.2: Comparisons Between Three Simulators (Adopted from [3])

Name/Version	OPNET Modeler 10.0.A	ns-2 2.27	J-Sim (formerly JavaSim) 1.3
Availability	Highly expensive, commercial software (no publicly available trial). Available with source code for simulation modules (except for restricted protocols)	Open-source software, available with full source code, validation tests and examples	Open-source software, available with full source code, and examples
Support	- excellent manual - excellent manual - source code and examples	- good manual - publicly available mailing list - source code and examples	- good manual - publicly available mailing list - source code and examples
Topology/ Scenario	- GUI, XML, import (e.g., HP OV) - "scenario" parameters - C/C++	- OTcl scripts (or C++)	- Tcl scripts (or Java) (as of 1.3) - OTcl (or Java) (future releases)
Extensions (components)	- C/C++	- OTcl (higher level) - C++ (lower level)	- Java (as of 1.3) - also OTcl for higher level (future releases)
Simulation mode	- synchronous, single threaded, discrete event queue based, with zero event processing time, fully deterministic - multithreaded, discrete event queue based, with zero event processing time - distributed simulation: HLA (High-Level Arch.)	- synchronous, single threaded, discrete event queue based, with zero event processing time, fully deterministic - parallel/distributed version available (Parallel /Distributed NS, PDNS)	- synchronous, single threaded, with zero event processing time, fully deterministic - multithreaded, "real-time process-based", with event processing time taken into account, non-deterministic
Brief summary (with subjective assessment)	- quite slow, "heavy weight" - expensive commercial software - ready, high-fidelity equipment and protocol models; a "reference" simulator - unique (e.g., military) features; widely used in NATO projects	- fast, quite modern, free - OTcl binding - simplified equipment models - many recent TCP mechanisms implemented for ns-2 - currently most popular in research projects	- scalable, modern, free - Tcl/Jacl binding (OTcl/Jacl) - simplified equipment models - new simulation paradigm (active components)

3.5.2.1 Network Simulator 2 (NS-2)

Network Simulator (version 2), widely known as NS-2, is an open source object-oriented simulator. It is a discrete event simulator targeted at network research [146]. It was developed as part of the Virtual Internet TestBed project (VINT) in 1989 and evolve based on a collaboration of many institutes and research centers. The NS-2 simulator has had a smooth transition from the NS-1 version, which had a similar architecture, and thus NS-2 was designed to be backward compatible with scripts written in NS-1. In contrast, the gap between the architectures of NS-2 and NS-3 is very large and NS-3 is not backward compatible. This would suggest that NS-2 will remain for many years as a useful tool, with an advantage of having a huge amount of accessible open source modules that had been developed during the last decade, all of which have not yet been ported to NS-3. The open source nature of NS-2 and the community-based development practices of NS-2 which are one of the main sources for its rapid development, are expected to continue with the NS-3 version.

NS-2 is one of the most popular used network simulators [147]. It provides libraries containing pre-defined modules for most communication protocols [148] besides an advanced environment to test and debug. NS-2 is a well-documented network simulator and provides a convenient environment for network researchers to work or extend any existing module. Furthermore, it is a well-validated tool. Hence all NS-2 modules of applications and protocols have been validated against detailed sets of tests. Additionally, whenever a new module is developed, a new evaluation will be held to confirm that the newly developed module works well and as specified with other modules of NS-2.

The validity of any simulation experiment is very crucial to assess any proposed protocol, so the validation process included in NS-2 could enhance confidence in using it,

as confirmed in [149]. Moreover, as an evidence that NS-2 has been used and accepted widely as a firm research tool in network research, a lot of studies conducted using NS-2 have been published in credible and high impact publications such as IETF RFCs, IEEE Transactions, IEEE/ACM Transactions on Networking, ACM SIGCOMM, IEEE INFOCOM, and others, as mentioned in [150, 151].

3.5.2.2 Experiment Steps

A decomposition of the simulation performance evaluation into steps is certainly helpful. Previous researchers had divided the simulation steps into 8 or 12 steps. For example, Hassan and Jain had divided the simulation task into eight steps [9], as shown in Figure 3.8. This research follows [9] simulation steps that consists of pre-software stage and software stage.

A. Pre-software stage includes four steps, as follows:

i. define the study objective precisely,
ii. design network model and select fixed parameters: this step is about design network topology on a piece of paper, and select appropriate network parameters to reflect a valid and real scenario,
iii. select performance metrics to be used for the evaluation study, and
iv. select variable parameters: in most TCP/IP simulations, the objective of the performance evaluation is to study the impact of certain variables on the selected performance metrics.

B. Software stage also includes four steps, as follows:

i. implement the topology designed in step A-2 into the software simulation program,

ii. configure or program simulation software to generate relevant performance metrics selected in step A-3,

iii. execute simulation software, and after the simulation finishes its runs successfully, the performance metrics data will be collected, and

iv. present the data collected in the previous step in a meaningful format then interpret the presented results.

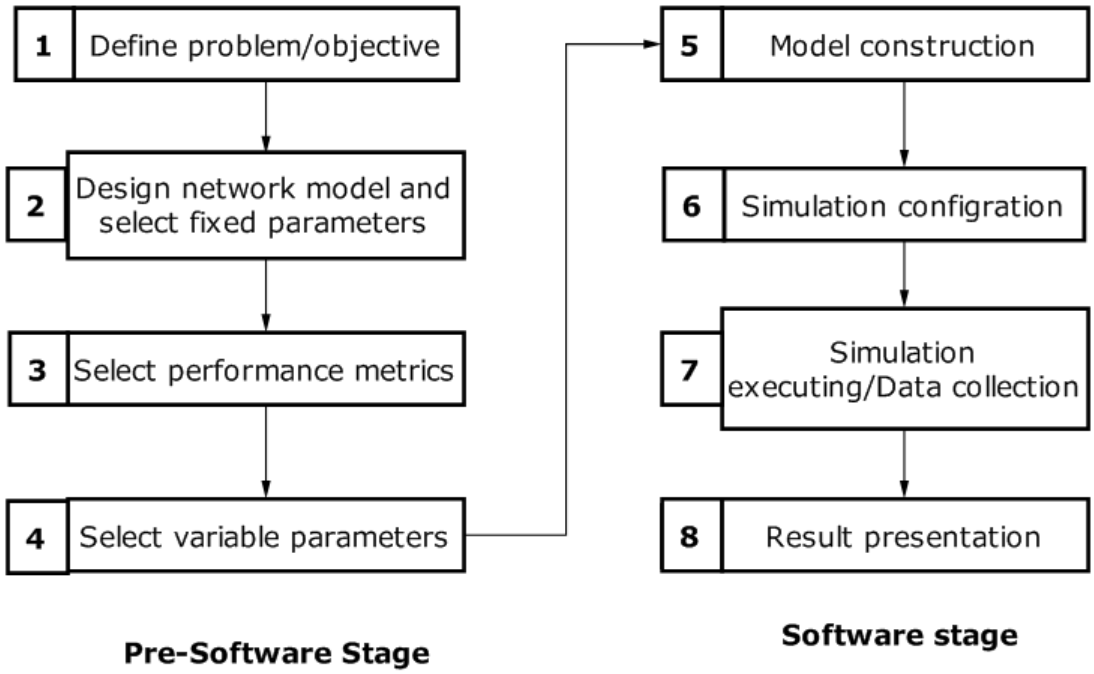


Figure 3.8: Simulation Steps (Adopted from [9])

3.5.2.3 Experiments Setup

All experiments that are presented in Chapters 4, 5, and 6 were performed using the network simulator (NS-2) version 2.34 on Ubuntu version 8 of the Linux operating system. Furthermore, all the experiments executed a termination simulation for 500 seconds, as performed by many other works [77]; following the same set up described

below, unless stated. The detailed setting of all the parameters are illustrated in Table 3.3.

Physical and MAC Model: Wireless LANs (WLAN) based on IEEE 802.11 family have recently become popular for allowing high data rates and relatively low cost [152]. Furthermore, IEEE 802.11 can run in infrastructure and ad hoc modes. Therefore, all nodes were configured with IEEE 802.11b PHY, IEEE 802.11b MAC Distributed Coordination Function (DCF) with RTS/CTS set ON [14, 10, 153, 154]. The nodes communicate with identical half-duplex wireless radios. The transmission range was set to 250 m and the sensing range was set to approximately 550 m. The data transmission was set to 2 Mbps.

Error Model: Wireless channel error can be generated from a wide variety of error models. Error model simulates link-level errors by marking the packet's error flag. For the purpose of this study and for creating a random uniformly distributed errors in the wireless channel, Uniform error model was used with Packet Error Rate (PER) between 1% and 10%.

Routing Protocol and Mobility Model: Two routing protocols, as recommended by IETF, was used during the performance evaluation, namely the Ad Hoc On-Demand Distance Vector (AODV) [155] as well as Dynamic Source Routing (DSR) [156]. DSR is a suitable mobile ad hoc network routing protocol where networks need to be established quickly [157]. However big overhead due to source-destination natural of DSR makes AODV a good alternative. Furthermore, AODV has good performance in different network sizes and it provides flexible and effective routing [158]. In the random topology, nodes are moving according to Random Waypoint Mobility Model [159]. This model is the most often used by researchers due to its properties in reflect-

ing the physical world movement, which is inherently unpredictable and unrepeatable. In this model, a node moves from its current position to a randomly chosen new position within the simulation area, using a random speed uniformly distributed between [v min ,v max], where *V min* refers to the minimum speed of the node and *V max* refers to the maximum speed. The Random Waypoint Mobility Model includes pause times when a new direction and speed is selected. As soon as a mobile node arrives at the new destination, it pauses for a selected time period (pause time) before starting to travel again.

Traffic Generation Model: FTP is frequently used to test TCP performance over ad hoc networks [157, 158]. FTP represents file transfer between two hosts and it has data to be sent continually though the experiment time. The size of TCP packet was set to 1000 Bytes and the length of ACK packet was 40 Bytes. Size of headers in greater detail are available in Table 3.3.

Topology and number of nodes: Several network topologies with different network sizes and varying number of nodes were used to test and evaluate the ability of TCP Sintok to react. More specifically, 6-hop and 7-hop chains; 5X5 nodes, 6X6 nodes, 7X7 nodes, and 8X8 nodes in grid topology; and random topology with 30 wireless nodes.

Table 3.3: Parameters Values

Parameter	**Value**
Simulation Topology	Dumbbell, Chain, Grid, and random
TCP Application Protocol	FTP
UDP/CBR flow rate	MP3 compress rate 180 Kbps
Packet Size	1000 bytes
Routing Protocol	AODV, DSR
MAC Protocol	IEEE 802.11b (direct-sequence spread spectrum)
RTS/CTS	ON
BasicRate	1 Mbps
DataRate	2 Mbps
RTS Length	160bits @ BasicRate + PHY header
CTS, ACK Length	112bits @ BasicRate + PHY header
MAC header	224 bits
PHY header	192 bits
TCP header	320 bits
SIFS	10 μs
DIFS	50 μs
Slot time σ	20 μs
CWmin	31
Short Retry limit (SRL)	7
Transmission Range	250 m
Carrier Sense Range	500 m
Long Retry Limit (LRL)	4
CTS_Timeout / ACK-timeout	300 μs
Queue Size	50 packets

3.5.2.4 Performance Metrics

Performance metrics can mean different things to different researchers depending on the context in which it is used. Furthermore, it is a clear consensus that congestion control mechanisms should be evaluated in tradeoff between a range of metrics such as throughput versus delay and loss rates; rather than optimizing of a single metric such as maximizing throughput or minimizing delay [160]. Furthermore, throughput and delay are frequently used for the performance evaluation purpose of TCP [51, 161, 162]. This section provides common and mathematical definitions of the performance metrics considered in the evaluation of the proposed TCP Sintok.

Throughput: It is simply defined by the total number of application bytes received by the destination per unit of time (i.e., experiment time per seconds). In other words, Throughput will be measured as a flow-based metric of per-connection transfer times. It is a clear goal of any efficient congestion control mechanism to increase significantly the Throughput, subject to application demand and to the constraints of the surrounding communication environment. In this research, throughput is measured as the number of data packets received successfully at the destination node per a unit of time (bit per second). The following formula is often used to calculate Throughput value:

$$Throughput = \frac{N}{T} \tag{3.1}$$

where N is the total number of bits received by the receiver node during the time interval T.

Delay: It is the total time required to send a packet from the source node to the destination node and receive its ACK from destination back to the source. This is often referred to it as Round Trip Time (RTT).

Jitter: Packet delay (Latency) variation is sometimes called "Jitter". Jitter is often used to measure the variability of end-to-end delay (e2e Delay) of packet delivery. It is calculated using the following formula:

$$jitter = abs(e2eDelay - preve2eDelay) \tag{3.2}$$

where $(e2eDelay)$ is the latest end-to-end delay, while $(preve2eDelay)$ is the previous end-to-end delay

3.5.2.5 Confidence Level of Simulation Results

The goal of performance evaluation is to obtain the mean (u) of the performance metrics discussed previously (i.e., throughput, delay). Random numbers are used in simulation experiments to generate random events such as channel error, packet arrival times, node movement, and so on. Due to this randomness, the obtained results will experience random phenomena as well. Therefore, the exact value of (u) is not possible to be obtained but it can be estimated from a number of different runs of simulation outputs. To emphasize, simulation output is only an estimation and some level of confidence on this outcome should be established. For the purpose of this study, 10 runs with different values of random number generator seeds were used for each scenario. For all collected metrics, interval of 95% confidence was estimated as well as point estimate for their averages. Details on the steps could be found in Hassan and Jain [9] and Kim et al. chapter [163].

3.6 Summary

This chapter has described in great detail the research approach to ensuring that the research objectives can be achieved. This research concentrates on developing TCP Sintok and its loss detection and contention avoidance mechanisms for ad hoc networks. Four main activities of the research were outlined in this chapter, in line with DRM. The first activity is the Research Clarification (RC) stage, which discusses methods to support the initial stage of this research. The aims of RC are to identify and refine a research problem, objectives, and research questions that are both academically and practically worthwhile and realistic. The second activity is called Descriptive Study-I (DS-I), which discusses steps to obtain sufficient understanding of the current situation, designs a reference model, and proposes a conceptual model. The third activity highlights the methods adopted in designing the proposed loss detection and contention avoidance mechanisms, named Prescriptive Study (PS). The last activity named DS-II focuses on the evaluation of the designed mechanisms and TCP Sintok.

After describing the methodology and experimental tool that was used to design and implement TCP Sintok in this chapter, the end-to-end delay model of TCP in ad hoc network and the design issues of the loss detection mechanism will be presented in the next chapter.

CHAPTER FOUR
DELAY-BASED LOSS DETECTION MECHANISM

After establishing the research methodology in Chapter Three as a guideline to achieve the objectives of this research, this chapter proposes a new Loss Detection Mechanism (LDM) for TCP in ad hoc networks. The proposed mechanism focuses on distinguishing mobility induced packet loss, mostly due to route failure and route change events, from congestion loss, by monitoring the trend of end-to-end delay as an indicator of the current network status.

Section 4.1 introduces an end-to-end delay model for TCP in multihop wireless ad hoc network. The model will be used to study the impact of congestion and contention on the delay variation, while the design objectives of the proposed Loss Detection Mechanism is presented in Section 4.3. Then, the Delay-Based Loss Detection Mechanism is designed to detect the cause of packet loss in Section 4.4. Finally, verification and validation are presented in Sections 4.5 and 4.6, respectively.

4.1 Theoretical Analysis

IEEE 802.11 is the dominant technology standard used in WLANs that provides detailed PHYsical layer (PHY) and MAC specifications for WLANs [10]. This standard supports the Distributed Coordination Function (DCF) by default, while the point coordination function (PCF) is used optionally [10]. The IEEE 802.11 DCF is used for wireless ad hoc networks and it is based on the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) technique. CSMA/CA is a popular MAC scheme that uses a combination of the CSMA and medium access with collision avoidance schemes being implemented. Additionally, the IEEE 802.11 specification defines two access techniques named basic access scheme and Request To Send / Clear To Send

(RTS/CTS) scheme.

In the basic access scheme, as shown in Figure 4.1, the transmitting node first senses the medium to check whether it is idle or busy. If the channel is idle and keeps idle for a period of Distributed Inter Frame Space (DIFS), the node transmits its frame. Otherwise, if the channel is sensed busy or changed its status from idle to busy within the period of DIFS, the nodes involved would generate a random backoff window to defer the transmission of the frame using the Binary Exponential Backoff (BEB) algorithm. The node selects a random backoff window from [0, CW]; the CW value is called contention window and it depends on the number of transmissions failures of the frame. The size of backoff window will be decreased by one if the channel has been in idle status and keeps idle for a (σ) period. Meanwhile, if the channel is sensed busy, the backoff window remains frozen. Once the backoff window size reaches zero, the node transmits the entire frame and then waits for the acknowledgment (ACK). Once the data frame arrives at the destination, the ACK will be transmitted after a period of time called the Short InterFrame Space (SIFS) of the data frame arrival. If the transmitting node receives the ACK that means its data frame has been transmitted successfully to the destination. If the acknowledgment is not received within a specified period (ACK_timeout), this means that the transmission has failed and the transmitting node re-enters the backoff phase. The transmitting node reschedules frame transmission based on the given backoff rules, while the contention window size will be doubled and another random backoff window is chosen for retransmission.

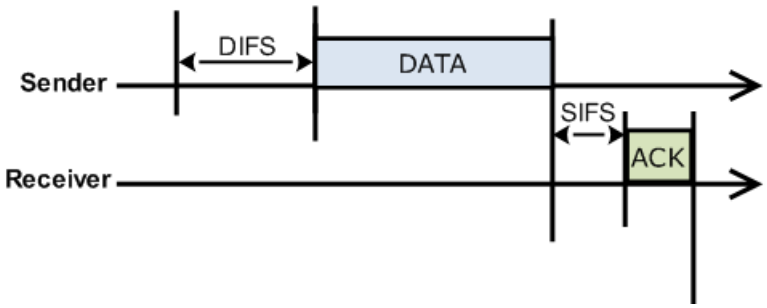


Figure 4.1: The IEEE 802.11 Basic Access Method

Additionally, DCF also provides the RTS/CTS reservation scheme for transmitting data packets, which uses small RTS/CTS packets to reserve the medium before large packets are transmitted in order to reduce the duration of a collision as shown in Figure 4.2. The IEEE 802.11 protocol uses the RTS/CTS-DATA-ACK sequence for data transmission. According to IEEE 802.11 DCF, each node that has a data packet for transmission sends the RTS packet and waits in order to receive a CTS packet. The receiver must wait a single SIFS time interval after receiving the RTS packet and then begin to send its CTS. Receiving the CTS packet at the transmitter means that the receiver is ready to receive a data packet, so the transmitter must wait a single SIFS time interval after receiving the CTS packet and then begins to send its own data frame. After receiving correct data packet by the receiver, the receiver will again wait for another SIFS time period, before transmitting an ACK packet to the transmitter. Before each node is allowed to transmit any RTS packets to start communication, the nodes should listen to the channel. If the channel is observed as being idle for a time interval longer than DIFS, then the BEB algorithm is launched.

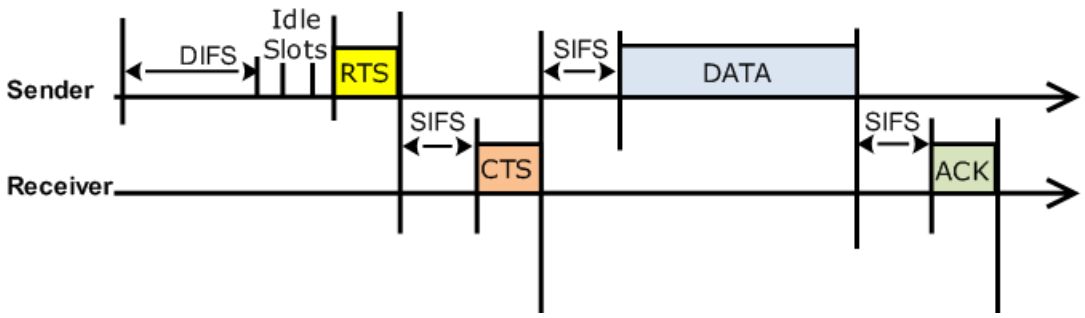


Figure 4.2: The IEEE 802.11 RTS/CTS Access Method

The BEB algorithm uniformly selects the backoff window in the interval [0, CW]. First of all, the DCF sets CW with the predefined value CWmin. The backoff window is decreased by one when the channel is sensed to be idle for more than a (σ) period, while the backoff window remains frozen in the case of busy channel. After each collision, the CW will be doubled until reaching another predefined CWmax value. When it reaches this CWmax, it keeps its value with subsequent failures. If the number of retransmissions of the same frame is greater than the maximum retry limit (by default, value of 7), the frame will be dropped. On the other hand, after each successful transmission, the backoff window and contention window (CW) will reset to the initial level regardless of the history of network condition or number of active nodes.

Previous studies, as discussed in Chapter Two, presented various different approaches to distinguish congestion loss from mobility loss based on one common ground, that is, delay (such as RTT, IDD, or Jitter) is a good congestion loss indicator. They all observed the delay variation of TCP in ad hoc network using network simulations and then proposed their improvement or enhancements to the protocols. Very few works however, had devoted more effort on theoretical analysis of TCP in IEEE 802.11 wireless ad hoc network [164, 165, 166]. This is because the analysis and modeling of end-to-end delay of TCP over IEEE 802.11 DCF based ad hoc network is a challenging task due to several reasons [167]. Firstly, TCP is characterized by end-to-end

closed loop flow control, which is in contrary to IEEE 802.11 that is characterized by a per link closed loop flow control. Secondly, IEEE 802.11 utilizes a four-way hand-shaking procedure to avoid collision caused by hidden terminals. Furthermore, ad hoc networks use a shared wireless channel which suffers from high bit error rates and contention. As a result, the analytical model of end-to-end delay is required for studying the delay variation of TCP segments in ad hoc networks.

In the literature, there are three models similar to the one presented in this thesis. The first model was proposed by Xiao at al. to analyze TCP performance over single and multihop wireless using the Markov chain model [165]. However, the model is too simplified by ignoring the impact of buffer overflow and contention in addition to some parameter values that were obtained from the simulation. The second model by Ding et al. introduced an analytical model to analyze TCP throughput in wireless multihop ad hoc networks over chain topology [164]. However, the model ignores wireless channel error and queueing delay, so more analyses are sorely needed to prove and validate the model in more complicated topologies. Recently, Ghadimi et al. [168] proposed a mathematical model of delay in multihop wireless ad hoc networks from the IEEE 802.11 MAC layer point of view, however, the model does not take wireless error into consideration.

4.2 System Model

This section shall propose a mathematical model of end-to-end delay of TCP over multihops IEEE 802.11 ad hoc networks. This model is built based on the Bianchi model [169] and aims to understand the impact of contention, congestion, and channel error on TCP end-to-end delay variation.

Let (s,d) denote a link from the source node (s) to the destination node (d) as depicted

in Figure 4.3. One TCP connection with infinite data is run from the node (s) to the node (d). There are TCP packets (data) at the source and ACK packets at the destination. Once a data packet is received by the destination node (d) an ACK will be generated immediately and sent to inform the sender of the next expected packet.

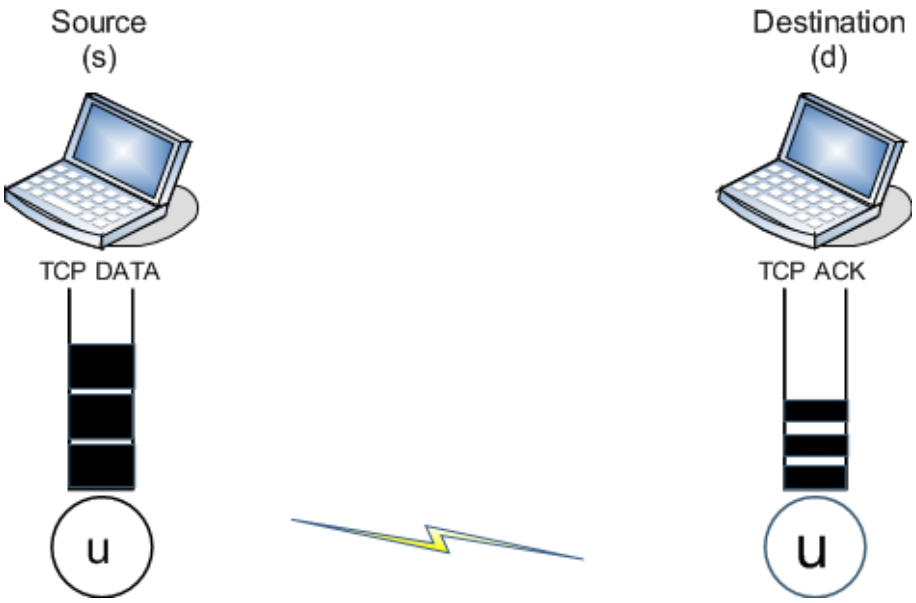


Figure 4.3: System Model of TCP with Single Hop

The network consists of (n) contending nodes within the same interference ranges of both (s) and (d) sharing a common wireless medium with capacity (μ) packet per second and First In First Out (FIFO) buffer of size (B) packets. Furthermore, each other node has packets available for transmission. All nodes use the IEEE 802.11 DCF with (RTS/CTS mode) as the MAC protocol.

The system model should capture TCP's unique communication features in multihop ad hoc networks including congestion (queueing delay), link layer contention, and error-prone wireless channel. The assumptions made to develop this model are as follows:

- Each data packet is acknowledged with an ACK packet.
- Data packet size is constant.
- The main cause of packet loss are collision (link layer contention) and channel errors.
- Collision occurs only in RTS control packets. The packet loss probability of CTS, DATA, and ACK due to collision is omitted. Furthermore, the collision-error probability of a transmitted packet is constant and independent of the re-transmissions that this packet has suffered in the past.
- Channel error will cause only TCP DATA packets to be corrupted. Furthermore, the corruptions of any packets are identically, and independently distributed on any links.
- Static route is used, where all nodes are stationary and routing protocol will run once. The route will be chosen with minimum path from the source to the destination. Packets are transmitted from the source node to the destination node using already known route omitting one feature of ad hoc network, which is dynamic frequent route changes due to mobility.

Delay is the summation of all transmission time and waiting time of a packet and its ACK over all the links in the path between the source node and the destination node. The delay in this context is named Round Trip Time (RTT) and its value is directly related to the probability of packet dropping. Thus, the following subsections present the packet dropping probability. Then, the delay model is formed for the case when the source node and the destination node are a single hop away. After that, the Model is generalized to represent the case when the source node and the destination node are multiple hops away.

4.2.1 Packet Dropping Probability

Assuming a simple ad hoc network consists of (n) contending nodes within the same interference ranges of both sender node (s) and destination node (d), each node has a packet available for transmission. The Round Trip Time (RTT) is directly related to the probability of packet dropping. Therefore, all packet dropping probabilities should be identified first before starting to analyze RTT. Packet loss due to channel error as well as simultaneous transmission (collision) is the point of focus based on the previous assumptions.

It is assumed that bit errors appear in the channel randomly with Bit Error Rate (BER). For convenience, Packet Error Rate (PER) is defined to represent the probability that backoff occurs in a node due to bit errors in packets and is given by:

$$PER = 1 - (1 - BER)^{l+H} \tag{4.1}$$

where (l) is the packet payload size and (H) is the header size.

Let (p) denote the probability that a transmitted packet encounters a collision or is received in error, which means at least one of the $(n-1)$ remaining nodes transmit in the same time slot, while considering the impact of packet error rate (PER), the following is obtained:

$$p = 1 - (1 - \tau)^{n-1}(1 - PER) \tag{4.2}$$

where (τ) denotes the probability that a node transmits a packet at a randomly chosen time.

The transmission probability of a given node is adopted from Bianchi [169], as follows:

$$\tau = \frac{2.(1-2p)(1-p)}{(1-2p)(W+1)+pW(1-(2p)^m)} \tag{4.3}$$

where (W) is the minimum contention window at the MAC layer, while (m) is number of backoff stages (equal to 6).

Equations (4.2 and 4.3) represent a nonlinear system in the two unknown parameters (τ) and (P) , which can be solved using numerical techniques and has a unique solution.

Let (P_{tr}) denote the probability that at least one transmission occurs in a randomly chosen time slot:

$$P_{tr} = 1-(1-\tau)^n \tag{4.4}$$

Let (P_s) denote the conditional probability that this transmission is successful:

$$P_s = \frac{n\tau(1-\tau)^{n-1}}{1-(1-\tau)^n}(1-PER) \tag{4.5}$$

Let (P_c) denote the probability that an occurring transmission collides due to two or more nodes transmitting at the same time:

$$P_c = 1 - \frac{n\tau(1-\tau)^{n-1}}{1-(1-\tau)^n} \tag{4.6}$$

Let (P_{er}) denote the probability that a packet is received in error:

$$P_{er} = \frac{n.\tau.(1-\tau)^{n-1}}{1-(1-\tau)^n}.PER \tag{4.7}$$

4.2.2 Single Hop Case

Let us have a link (s,d) that connects node (s) with node (d). Let (D_s) denote the DATA transmission time over this link (s,d), and (A_s) denote the ACK transmission time from node (d) to node (s) over the same link (s,d), then the Round Trip Time (RTT) will be equal to:

$$RTT = D_s + A_s \tag{4.8}$$

Although the transmission process of DATA packet and ACK packet are the same, their sizes are different. Therefore, the transmission time of a packet from one node to the next node is derived without discrimination whether it is DATA or ACK packets. Hence, the minimum time incurred by a packet to be transmitted from node (s) to the node (d) is given as follows:

$$D_s = E[T_{sd}] + \frac{1}{\mu_s} \tag{4.9}$$

where $E[T_{sd}]$ is the expected time of transmitting a packet over the link (s,d) and (μ_s) is the link speed.

The above analysis gives an optimistic one-way delay estimation, since it assumes that packet arrives to an empty buffer (i.e., queuing delay is ignored). Therefore, to get a more realistic formula the queuing delay should be added to the above estimation, which is as follows:

$$D_s = \sum_{1}^{B_s+1} (E[T_{sd}] + \frac{1}{\mu_s}) \tag{4.10}$$

where (B_S) is the average number of packets waiting in the buffer when a new packet arrives. Meanwhile, (B_S) is estimated as follows:

$$B_s = \lambda (E[T_{sd}] + \frac{1}{\mu_s}) \leq BufferSize \tag{4.11}$$

where (λ) is the average sending rate.

In addition to the processing time by the queue (i.e., queueing delay), the expected time for transmitting a packet $E[T_{sd}]$ over a link (s,d) consists of the following parts:

i. successful transmission time (T_{s_s}) of the packet from first node to the next one,

ii. backoff time (T_{bo_s}) based on the channel status and backoff window value,

iii. collision time (T_{c_s}), and

iv. transmission error time (T_{er_s}).

The expected transmission time (T_{sd}) from node (s) to node (d) is equal to the summation of the following four parts:

$$E[T_{sd}] = T_{s_s} + T_{bo_s} + T_{c_s} + T_{er_s} \tag{4.12}$$

The successful transmission time value (T_{s_s}) depends on the medium access mechanism and the RTS/CTS scheme, and it is defined as follows:

$$T_{s_s} = DIFS + T_{RTS} + SIFS + T_{CTS} + SIFS + T_{DATA} + SIFS + T_{ACK} \tag{4.13}$$

where $T_{DATA}, T_{ACK}, T_{RTS}, T_{CTS}$ are the transmission time of data (including payload and header), acknowledgment, RTS and CTS packets respectively as shown in Figure 4.4.

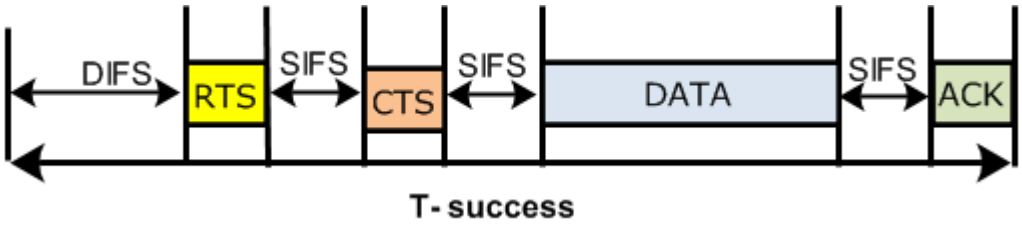


Figure 4.4: Successful Transmission Time Based on RTS/CTS Access Method According to [10]

The backoff time of a packet is determined by the probability of packet loss (P) at

the sender node (s) in the link (s,d) in addition to CW and backoff window values. In the initial backoff level (Stage 0), the CW value has the value CWmin. After each collision, the CW will be doubled until reaching the maximum CWmax as shown in Figure 4.5. Given that CW is measured by time slot (σ), the expected backoff time is calculated as follows:

$$T_{bo_s} = \sum_{k=1}^{SRL} P^{k-1}(1-P).2^{k-1}CW.\sigma \tag{4.14}$$

where (σ) is the time slot length, (SRL) is the Short Retry Limit, and (CW) is the minimum size of contention window.

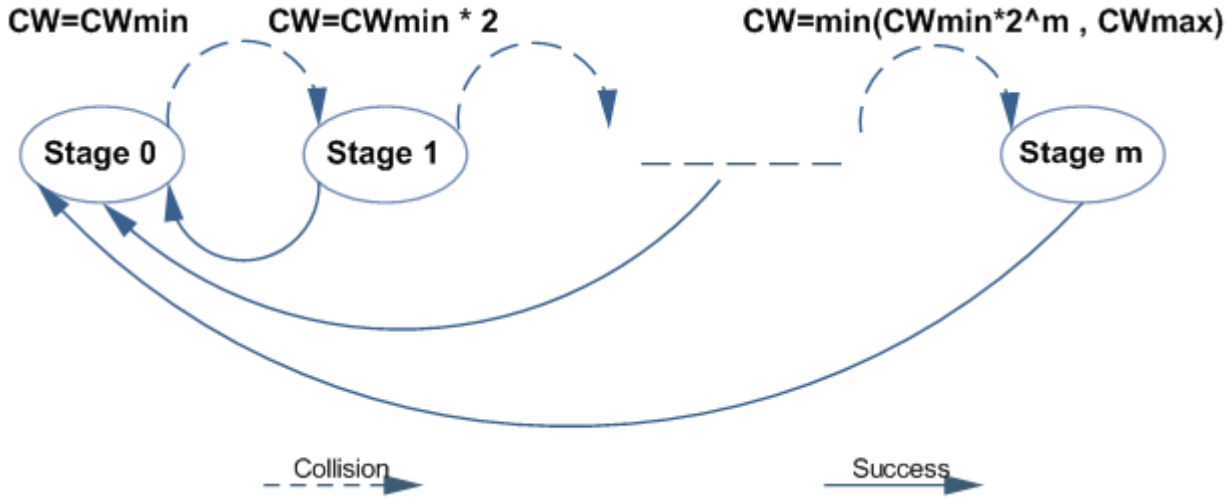


Figure 4.5: Contention Window Value According to [11]

The collision time (T_c) is determined by the Initial Collision Time as shown in Figure 4.6 and the average number of collisions (NC). T_c is calculated as follows:

$$T_c = (DIFS + T_{RTS} + CTS\_timeout).NC \tag{4.15}$$

where $CTS\_timeout$ is the sender's waiting time to receive the CTS packet from the receiver after transmission of RTS packet.

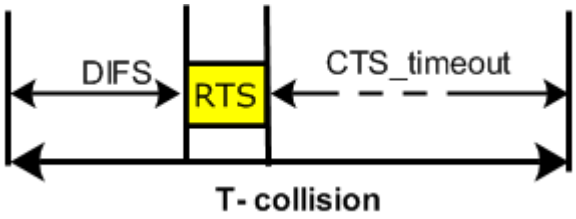


Figure 4.6: Initial Collision Time

The average number of packet collisions at the sender node (s) is needed. Suppose $(P_{tr}.P_C)$ is the probability of collision of any transmission attempt, if the sender makes independent attempts over and over, then the geometric random variable, denoted by $Nc \sim geo(1-P_{tr}.P_C)$, counts the number of trials until obtaining the first successful transmission. Accordingly, the average number of packet collisions is the expected value of the random variable Nc. $E[Nc]$ can be calculated as follows:

$$Nc = \sum_{k=1}^{SRL} (P_{tr}.P_C)^{k-1}(1-P_{tr}.P_C)(k) \tag{4.16}$$

where (SRL) is the Short Retry Limit.

The time cost of transmission error (T_{ers}) depends on the average number of errors (N_{ers}) and the transmission time of T_{RTS}, T_{CTS} and T_{DATA}:

$$T_{ers} = (DIFS + T_{RTS} + SIFS + T_{CTS} + SIFS + T_{DATA} + ACK\_timeout).Ner \tag{4.17}$$

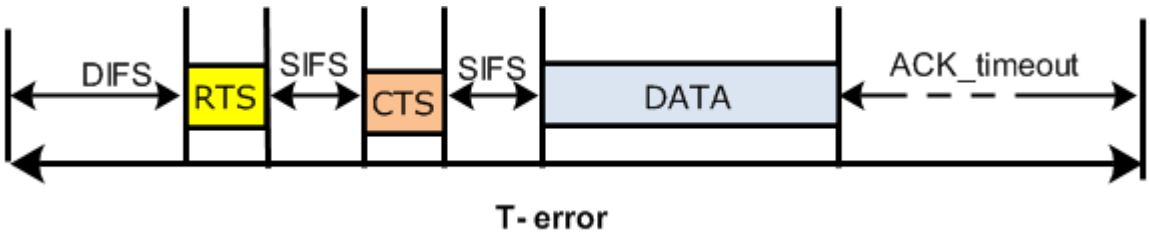


Figure 4.7: Initial Transmission Error Time

The average number of transmission error (Ner) is the expected value of the geometric random variable NER with the probability of transmission error $(P_{tr}.P_{er})$. $E[Ner]$ is calculated by Equation 4.18:

$$Ner_s = \sum_{k=1}^{LRL} (P_{tr}.P_{er})^{k-1}(1 - P_{tr}.P_{er}).(k) \tag{4.18}$$

where (LRL) is the Long Retry Limit for data packet and its default value is equal to 4.

4.2.3 Multi-Hop Case (Generalization of RTT)

RTT is determined by the transmission time and waiting time of a DATA packet and its corresponding ACK over all links through the path from the sender node to the destination node. Let (D_s) denote the transmission and waiting time of a data packet; (A_s) denote the transmission and waiting time of ACK packet from the destination node (d) to the sender node (s); then the round trip time (RTT) for sending data packet from the source node (s) and receiving its ACK is calculated as follows:

$$RTT = \sum_{i=1}^{h} D_s + \sum_{i=1}^{h} A_s \tag{4.19}$$

where (h) is the number of links (hops) between the source node (s) and destination node (d). The round trip time is the summation of the transmission time as well as the waiting time over each individual link between the source node and the destination node for the DATA packet and its related ACK.

4.2.4 Model Validation

To validate the accuracy of the proposed end-to-end delay model, a linear chain as well as dumbbell topologies were considered with a variety of scenarios using the NS-2 simulator. The former topology (chain) is used to study the impact of multihop and different *BER*; where the data packets originate at the first node in the chain and are forwarded by the intermediate nodes to the last node of the chain. While the latter topology (dumbbell) is used to investigate the impact of competing flows and contention on TCP delay. Figure 4.8 shows a dumbbell simulated scenario. The distance between any edge node and its neighboring intermediate node is set to 200 m. The same distance is applied between the two intermediate nodes. AODV is used as the routing protocol, while IEEE 802.11b DCF is used as the MAC protocol. More details about all related parameter values have been presented in Chapter Three.

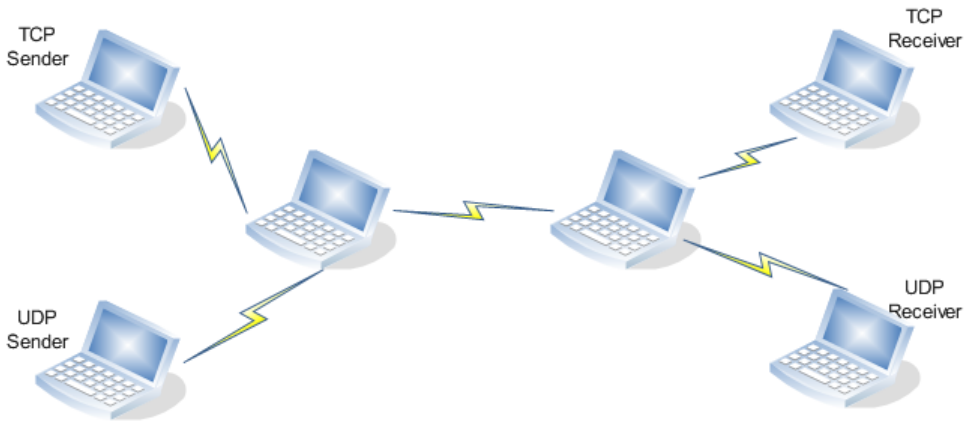


Figure 4.8: Dumbbell Topology

For each simulation scenario, the experiment is repeated ten times and the average

value of end-to-end delay is obtained. Then, the average value obtained from the simulation experiments is compared to the delay value calculated based on the proposed model using MATLAB [170, 171]. Additionally, for coordination between the analytic and simulation results, the transmission rate is obtained from the simulation and applied in the mathematical model for each case.

Figure 4.9 shows the average end-to-end delay of TCP over chain topology of one, two, and three hops with 10^{-6} BER. With a single hop distance, both analytical and simulation receive almost the same end-to-end delay values. When the number of hops increases, the analytical model achieves a slightly higher average delay as compared to simulation. However, both curves have the same delay trend with 90% accuracy level accomplished by the proposed mathematical model.

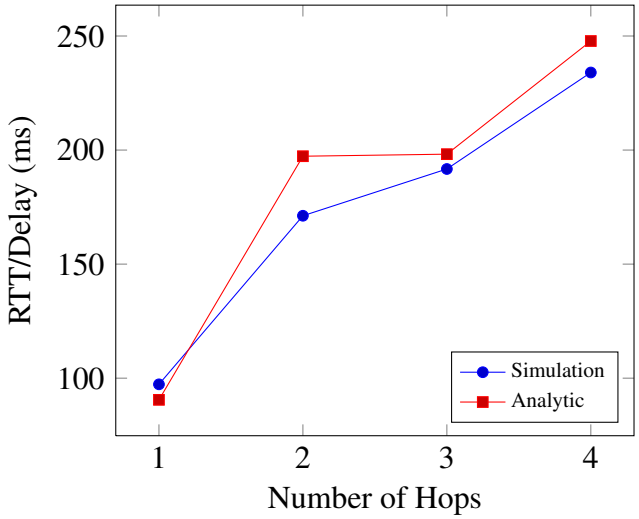


Figure 4.9: End-to-End Delay in Chain Topology with BER $= 10^{-}6$

To gain a deep insight on the impact of bit error rate on delay value, the previous scenario is repeated again with BER equal to 10^{-5}. Figure 4.10 shows a comparison between average end-to-end delay obtained from the simulation and the proposed analytical model over one, two, and three hops chain. Again, both models produce the

same delay trend and the proposed delay model achieves 90% accuracy level.

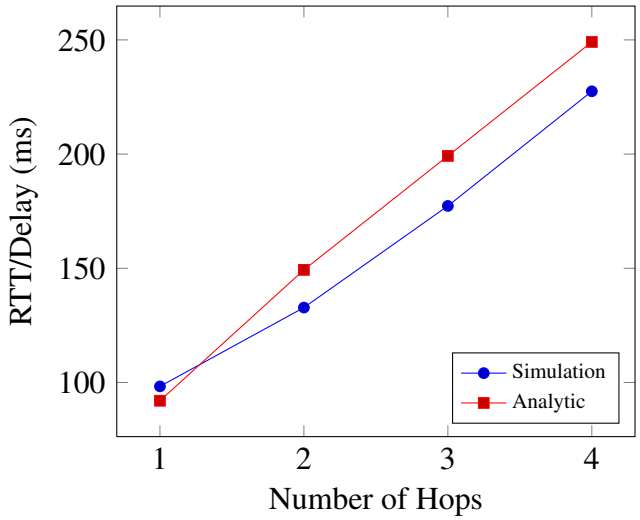


Figure 4.10: End-to-End Delay in Chain Topology with BER $= 10^{-}5$

To study the impact of contention on the average end-to-end delay, dumbbell topology is simulated with one TCP sender, and different number of competing UDP senders with sending rate (180 Kbps), as illustrated in Figure 4.8. Figure 4.11shows the relation between end-to-end delay of TCP in three cases; single TCP sender, TCP with one UDP competing flow, and TCP with two competing UDP flows in dumbbell topology.

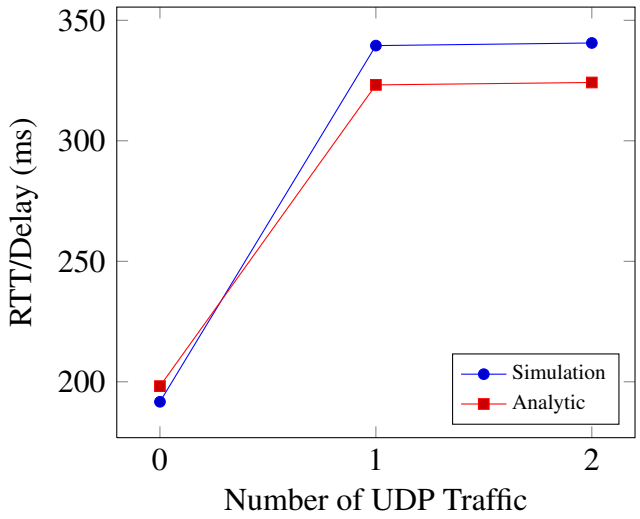


Figure 4.11: End-to-End Delay in Dumbbell Topology with Different Number of Back Ground Traffic

Figures 4.9, 4.10, and 4.11 summarize the result of the comparison between the proposed delay model and the simulation. The results prove that the proposed model is fairly accurate with 90% accuracy level in almost all cases.

After the proposed model is validated using simulation and for the purpose of studying the impact of contention and congestion in ad hoc network on end-to-end delay, the dumbbell topology scenario is used again to calculate the delay based on the proposed mathematical model. Dumbbell topology is used with three cases: i.e., single TCP sender, TCP with one UDP flow, and TCP with two competing UDP flows. TCP sending rate is fixed in all cases and the model is used to calculate the delay value as presented in the Figure 4.12. Using the model, it is very clear that contention accompanies increasing delay. Moreover, delay increases when the number of competing flows increases.

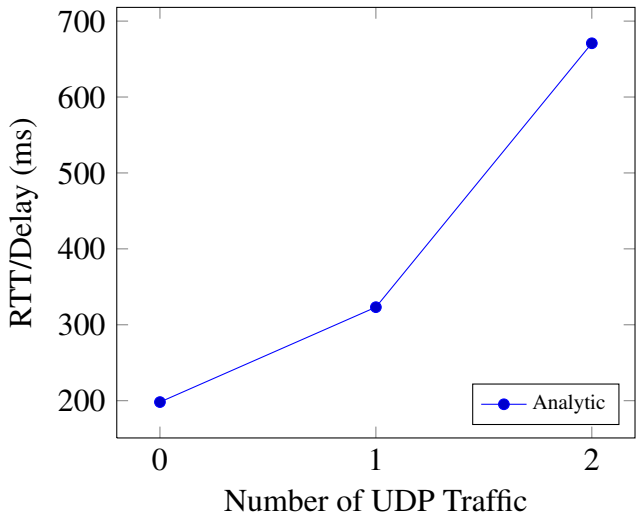


Figure 4.12: End-to-End Delay in Dumbbell Topology with Different Number of UDP Flows of 180Kbps Sending Rate

4.3 Design Objective of Loss Detection Mechanism (LDM)

As stated in Chapter one, TCP should be able to identify congestion loss from packet loss due to route failure or route change. However, using triple duplicate ACKs (3DU-

PACKs) and RTO alone are not enough to accurately determine network states. According to our proposed delay model, it is confirmed that network overload (contention) in ad hoc networks is accompanied by increasing delay as shown in Figure 4.12. Even so, a high bit error rate contributes to increase the delay value as well. However, most packet loss due to channel error could be recovered by the lower layer and hide this loss from TCP. Therefore, end-to-end delay has a potential to be used as a supporting metric to detect mobility loss from congestion loss. In line with that, Delay-Based Loss Detection Mechanism (LDM) is proposed for TCP congestion control over ad hoc network to distinguish packet loss due to route failure and route change from congestion loss. LDM is aimed to meet the following key features:

- Provide TCP sender with updated and accurate information about the network condition without interfering with the traditional TCP operations.
- Maintain TCP end-to-end semantics. Furthermore, the proposed mechanism in this research is a sender based only (i.e., no processing or data collection will be conducted at the receiver side). Therefore, LDM is easier to be implemented in such a heterogeneous environment.
- Use Delay as a loss indicator similar to previous works, but the proposed mechanism in this research applies a simpler approach to identify network condition. Thus, LDM is expected to reduce the complexity and computational costs as compared to others.

Traditional definition of congestion is a network overload at the bottleneck node (i.e., buffer overflow). Meanwhile, in ad hoc networks, contention for access to the shared wireless channel offers the first sign of network overload or congestion. Furthermore, simulations showed that link-layer contention induced packet drop dominates, while buffer overflow is almost never experienced by TCP flows in typical multihop wireless

networks [28]. Thus, network overload becomes a phenomenon that occurs in an area rather than in a single node and it is still accompanied by increasing delay as proven by the proposed delay model. Consequently, it is more accurate to calculate the delay along the forward and reverse paths to estimate the network status. However, the following questions remain to be answered:

- How to calculate RTT samples and maintain TCP semantics?
- How to identify the RTT trend (evolution) based on the observed history of RTT samples?
- What is the suitable sample space size? How to map RTT measurement to network states?
- How to answer these questions without posing extra overhead to TCP?

All these questions will be answered by LDM mechanism.

4.4 The Design of Loss Detection Mechanism (LDM)

RTT is defined in [172] as the time interval between sending a packet and receiving its acknowledgment (i.e., RTT measurements conflate delays along the forward and reverse paths). For example, let (TS_i) denote the sending time of packet (i), and (TR_i) denote the receiving time of packet (i) ACK; then RTT is expressed as follows:

$$RTT = TR_i - TS_i \tag{4.20}$$

Accurate RTT estimation is necessary to identify changing network conditions. However, it may be difficult both in theory and in implementation. A solution to this issue

is using TCP options proposed in RFC1323 [172], where the sender places a timestamp in each data packet, and the receiver reflects these time stamps back in ACK packets. Then a single subtraction gives the sender an accurate RTT measurement for every packet, as illustrated in Equation 4.20.

The RTT values are collected by TCP sender and stored in a sample space (SS), where the size of (SS) should be set appropriately. If it is set too small, the observed samples are not enough to analyze network status; if it is too large, the observed samples may be outdated or not valid. Conventional NewReno assumes a packet loss to have occurred upon receiving of 3DUPACKS, in other words, four ACKs are enough to deduce a packet loss event. Therefore, (SS) with four (RTT) samples will help to diagnose network conditions. In addition to the four most recent (RTT) values, the Smooth RTT (SRTT) will be added as well to the sample space as a representative of the previous RTT values. As a result, the total (SS) size is equal to five and the content includes four most recent (RTT) and (SRTT), as shown in Figure 4.13.

Value	RTT1 (most recent)	RTT2	RTT3	RTT4 (oldest one)	SRTT
index	0	1	2	3	4

Figure 4.13: Sample Space Content

Next, the content of this (SS) should be determined to express the current network status. The standard TCP uses RTT samples to update the averaged RTT measurement only if the packet acknowledges some new data, i.e., only if it advances the left edge of the send window. RTT value of any packet that contains duplicate acknowledgment will be excluded in the calculation to avoid any fluctuation that may affect the accuracy of smooth RTT estimation. Therefore, the sample space will contain the latest four RTT of new ACK packets in addition to the SRTT. Figure 4.14 shows an example for

sample space values.

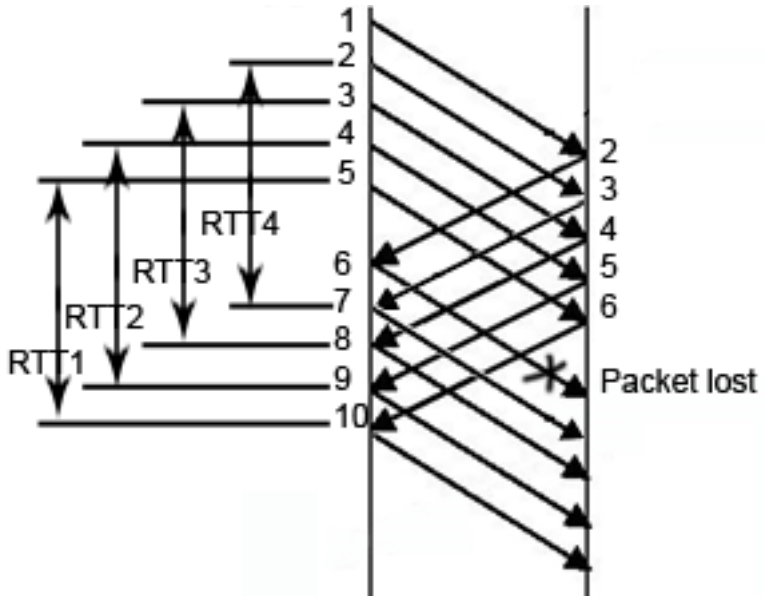


Figure 4.14: LDM Sample Space

Given the RTT samples, the proposed mechanism should indicate the trend of RTT samples once the third DUPACKS arrived or when timeout has occurred. If the trend is not increasing, it indicates a sign of route change or route failure. Otherwise, the network is congested. To capture this trend, Pair Wise Comparison Test (PCT) [173] is applied on the sample values, where each value in the sample space will be compared to the next value, as follows:

$$PCT = \frac{\sum_{k=1}^{4} I(RTT_{k-1} > RTT_k)}{4} \tag{4.21}$$

where $I(X)$ is one if X holds (i.e., RTT_1 Sample is greater than RTT_2), and zero otherwise.

PCT takes a value in the range between $[0,1]$. If there is an increasing trend, then PCT

approaches one. For the purpose of LDM, if PCT value is greater than 0.5, then the trend is increasing. Pseudo code to illustrate LDM steps is presented as follows:

Algorithm 4.1 UPON 3rd DUPLICATE ACK or RTO

Step 1. Estimate the evolution of delay using PCT

$$PCT = \frac{\sum_{k=1}^{4} I(RTT_{k-1} > RTT_k)}{4} \tag{4.22}$$

Step 2. **if** PCT <= 0.5 **then {**

Step 3. return network is not congested (ROUTE FAILURE/CHANGE status)

Step 4. **} else {**

Step 5. return network is congested (CONGESTION status)

Step 6. **}**

4.5 The Verification of LDM

Verification of LDM mechanism was done using the method illustrated in Chapter Three. Portion of the LDM mechanism after the verification process is presented in Figure 4.15 and confirmed the following:

- LDM has been programmed correctly, and
- LDM implementation does not contain any errors or bugs.

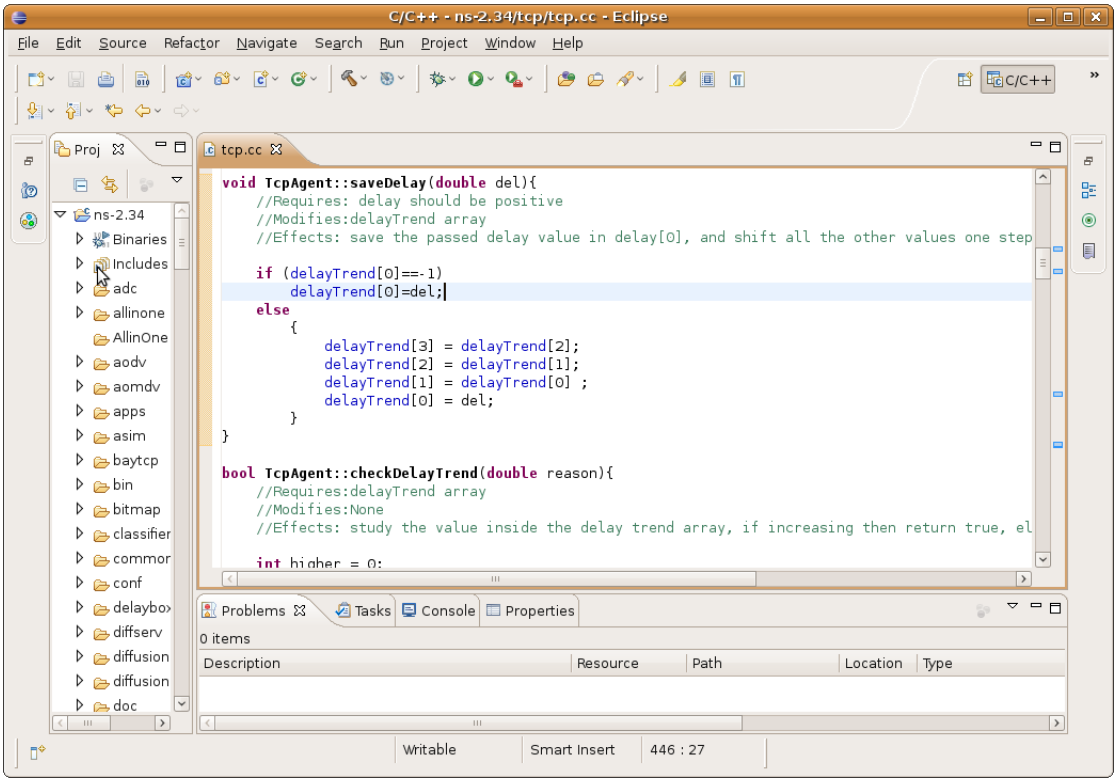


Figure 4.15: LDM Code in Eclipse

4.6 The Validation of LDM

It was proven that network congestion is accompanied with increasing delay. Therefore, this section will focus on detecting mobility induced packet loss using the proposed LDM mechanism. The validation of LDM mechanism was done to ensure that the LDM mechanism meets its intended requirements in terms of distinguishing mobility loss from congestion loss. In other words, the proposed LDM is intended to increase the accuracy of loss detection by reducing the false alarm due to misinterpreted mobility loss as congestion loss.

Assume two Hypotheses $(H0)$ and $(H1)$, where $H0$: Congestion is absent (Mobility is present), $H1$: Congestion is present. One and only one of these hypotheses is true, i.e., $H0$ and $H1$ are mutually exclusive.

Additionally, let event (A) denote that $(H0)$ is selected and (A_c) denote that $(H1)$ is selected. Based on these assumptions, false alarm occurs due to choosing $(H1)$ as cause of packet loss when $(H0)$ is true (congestion loss is present when it is actually absent). Figure 4.16 shows the probability in tree structure and the average probability of false alarm is estimated as follows:

$$P[FalseAlarm] = P[A_c|H0]P[H0] \tag{4.23}$$

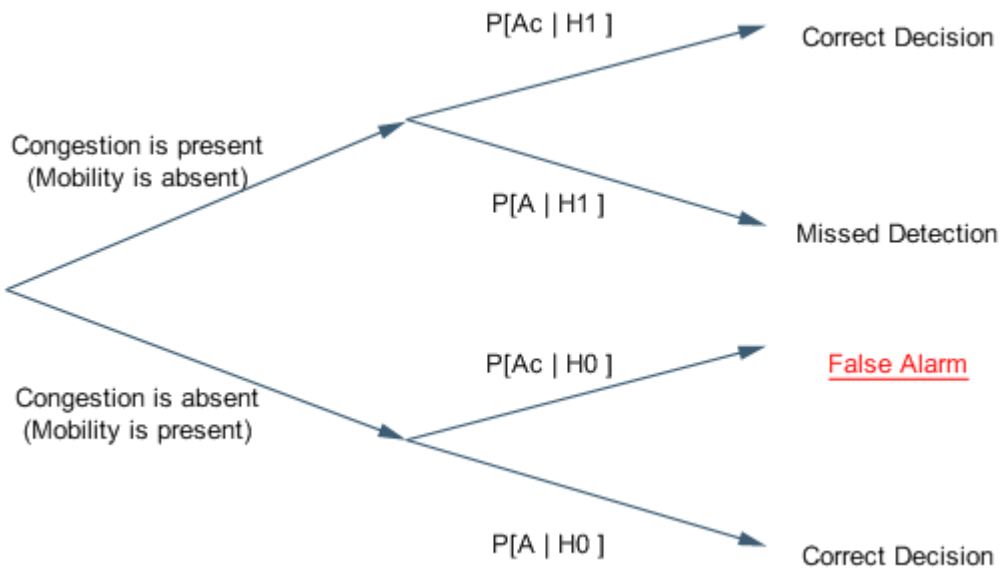


Figure 4.16: Probability Tree Structure in Normal Case

The proposed LDM focuses on distinguishing mobility loss from congestion loss leading to reduce false alarms as follows:

Let event (B) denote packet loss with increasing delay (RTT) trend and its complement (Bc) denote packet loss with non-increasing (RTT) trend. Figure 4.17 shows the probability in tree structure in case of applying LDM. The average probability of false alarms $P[FalseAlarm]$ is calculated as follows:

$$P[FalseAlarm] = P[A_cB|H0]P[H0] = P[Ac|H0]P[B|H0]P[H0] \quad (4.24)$$

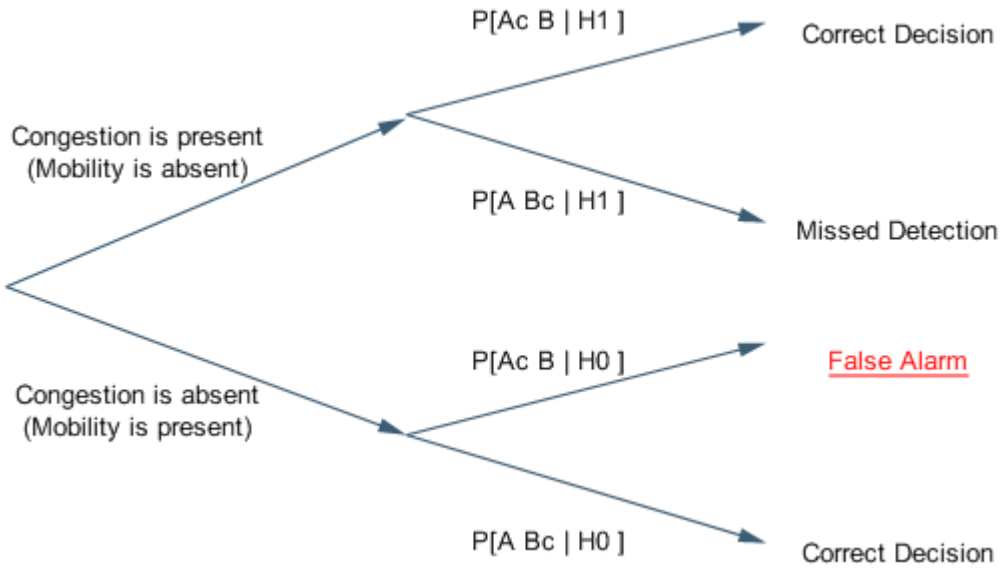


Figure 4.17: Probability Tree Structure in the Case of Using LDM

Since probability, $0 < P[(B|H0)] < 1$, is non-negligible, the proposed mechanism achieves higher accuracy in terms of decreasing false alarms.

To give a quantitative analysis of the percentage of false alarm reduction, validation experiments were conducted by using NS-2 simulator. The default settings of the simulation are exactly as stated in Chapter Three. Simulations were run under non-congested and congested scenarios. In the non-congested case, a single TCP flow was created. In the congested case, four competing TCP flows were created. In both cases, 30 wireless nodes were initially positioned at random locations over a 400mX800m area. Nodes move randomly following a random way point mobility model. During the simulation, attention was given to detecting mobility losses using LDM to prove the assumption that packet loss without increasing delay is a sign of mobility loss. For each packet loss due to mobility, the actual network state obtained from NS-2 trace

file was compared to the identified network state by LDM to determine the percentage of false alarm reduction.

Table 4.1 illustrates the percentage of false alarms reduction through non-congested and congested scenarios, over a pause time zero and $(10sec)$ cases. It is clear that there is a consistent trend in the percentage of false alarm reduction in the case of pause time zero, while the case of pause time $(10sec)$ saw a downward trend.

Looking at the details, the percentage of false alarms reduction in the non-congested case and pause time zero is 89.5%, and it is similarly maintained for the congested case with 89.8%. However, in the case of pause time $(10sec)$, the percentage of false alarms reduction starts with 93.6%, and then dropped slightly to 80.4% .

Table 4.1: Percentage of False Alarms

Random Topology	**% False Alarms**	
	Non-congested Case	**Congested Case**
Pause Time Zero	89.5%	89.8%
Pause Time 10	93.6%	80.4%

To eliminate the impact of randomness due to the mobility model, further experiments were conducted with various speeds from 5, 10, and 15 m/s and different pause time 0, 10 and 20. The simulations were repeated 10 times for each speed and pause time so as to avoid the impact of random factors. The average percentage of false alarms reduction for non-congested case and congested case was calculated based on 90 ex-

periments, as shown in Table 4.2. In the non-congested case, mobility is the dominant cause of packet loss. The observed false alarm reduction was 86.1% in the average of 90 experiments. On the other hand, in the multiple TCP flow case, congestion occurs more frequently, and the observed accuracy reduced to 79.1%. This is due to the fact the congestion and mobility happened at the same time in many cases. As a result, the achieved accuracy is acceptable in all cases.

Table 4.2: Overall Percentage of False Alarms

	% False Alarms	
	Non-congested Case	**Congested Case**
Random Topology	86.1%	79.1%

4.7 Summary

This chapter introduced end-to-end delay model and Delay-Based Loss Detection Mechanism (LDM) for TCP over mobile ad hoc networks. The proposed Loss Detection Mechanism (LDM) for distinguishing mobility loss from congestion loss using delay trend was described in this chapter. TCP uses LDM only once packet loss is detected by RTO or 3DUPACK. Results demonstrated the potential of using LDM for reducing false alarms (i.e., distinguishing mobility induced packet loss from congestion loss) and improving the accuracy of loss detection. LDM achieves around 86.5% false alarms reduction in the non-congested case and 79.1% in the congested case.

LDM detects route failure/change loss without addressing the issue of contention loss. Since detecting contention loss will not solve the problem, a Contention Avoidance

Mechanism is required to adapt the sending rate based on network conditions. The following chapter will address this issue in greater detail.

CHAPTER FIVE
CONTENTION AVOIDANCE MECHANISM

In the previous chapter, Loss Detection Mechanism (LDM) was introduced to distinguish mobility loss from congestion loss. Furthermore, Chapter Four shows the need for adapting the growth of congestion window to avoid contention and increase spatial channel reuse. This chapter considers the design and implementation of Contention Avoidance Mechanism (CAM) for TCP in ad hoc networks. Theoretical analysis of TCP AIMD is presented in Section 5.1. Section 5.2 introduces Communication Accommodation Theory (CAT) and its terminology as the main theory to be applied in the proposed contention avoidance mechanism. In Section 5.3, the design of CAM is discussed, while the implementation of CAM is presented in Section 5.4. Finally, verification and validation of CAM are discussed in Section 5.5.

5.1 Theoretical Analysis

TCP congestion control is often referred to as an Additive Increase, Multiplicative Decrease (AIMD) algorithm. TCP additively (linearly) increases its congestion window until a triple duplicate ACKs (3DUPACKs) is received as an indication of packet loss [161]. It then retransmits the lost packet and decreases its congestion window size by a factor of two (multiplicatively decrease) but again begins increasing it linearly, probing to see if there is additional available bandwidth.

AIMD is able to recover a single packet loss per window. However, it is not capable of recovering multiple packet losses in the same congestion window and has a negative impact on TCP performance. Therefore, TCP NewReno is proposed to cater for the recovery of multiple packet losses within the same window. Hence, it allows fast retransmit to recover multiple losses while the sender only receives a partial new ac-

knowledgment and the fast recovery exits when all packets have been acknowledged. With that, TCP was well designed to work over wired networks where most packet losses occur due to a buffer overflow event at the bottleneck router (i.e., network overload). However in ad hoc networks, packet dropping may occur due to either buffer overflow or link-layer contention. Further analysis of the packet loss reasons revealed that link layer contention typically happens before buffer overflow [100]. In particular, packet loss due to link layer contention dominates while buffer overflow imposed packet loss is rare [77, 174]. As a result, packet loss due to link layer contention offers the first sign of network overload in ad hoc networks [175].

In ad hoc networks, senders within a local neighborhood have to compete for wireless channel access before transmitting. The shared wireless channel allows a single sender to transmit per time [176]. The likehood of packet loss due to link contention increases when the offered load increases. Fu et al. in [28] showed that the optimal window size (W*) at which TCP achieves the highest throughput depends on the number of hops the TCP flow travels. However, TCP congestion avoidance mechanism increases the window size largely beyond (W*). The large TCP window size of the sender causes an excessive number of medium accesses leading to contention at the shared wireless channel, thus resulting in excessive collisions and packet losses [69]. This implies that TCP congestion avoidance, designed to adapt the sending rate (congestion window) in wired networks, does not work well in wireless ad hoc networks where packet loss due to link layer contention dominates. Thereby, this is considered one of the main reasons for poor performance of TCP over IEEE 802.11 MAC protocol.

Simulation experiments were conducted to gain a deep understanding on the impact of contention on TCP performance. Three TCP congestion control mechanisms, namely TCP Tahoe, TCP Reno, and TCP NewReno were studied over chain and grid topolo-

gies with different number of hops. In each simulation experiment, a single TCP flow runs between a pair of nodes. Single flow will cause TCP data and ACK packets of the same flow to compete with each other (self contention) to access the shared medium. Other ad hoc network features such as mobility, channel error, and congestion are ignored to simplify the analysis and focus on contention.

Figures 5.1 and 5.2 show the throughput of TCP Tahoe, Reno, and NewReno in chain and grid topologies, respectively. It is obvious that the throughput of all TCP flavors over both topologies degrades as the number of hops increases between the source and destination. This is because hidden and exposed terminals lead to collision among data and ACK packets in the same flow. Additionally, results support that TCP NewReno is recommended to be the benchmark for future improvements of TCP.

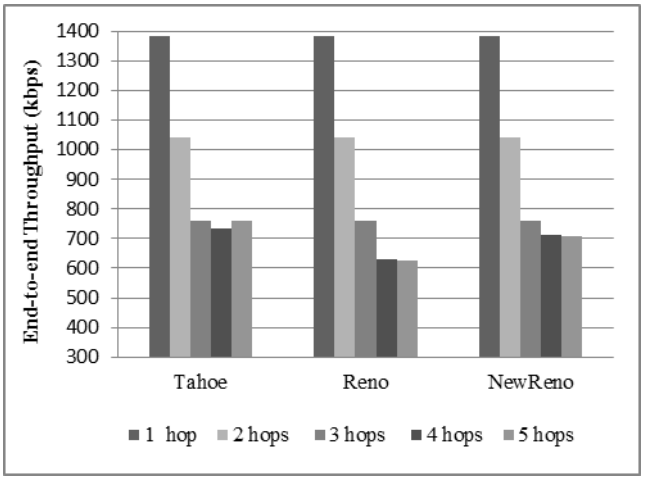


Figure 5.1: TCP Throughput in Chain Topology

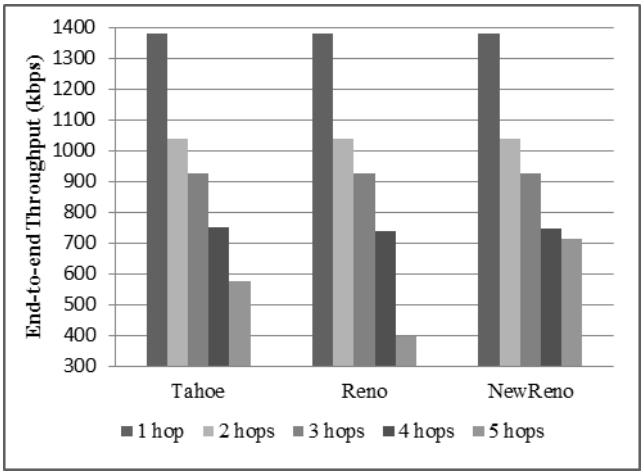


Figure 5.2: TCP Throughput in Grid Topology

To this end, detecting contention loss will not solve the problem but avoiding the contention has higher impact on improving TCP performance. Therefore, the main objective of this chapter is to propose a new Contention Avoidance Mechanism (CAM) to address this issue.

5.2 Applying CAT to TCP Congestion Control

In order to apply CAT, a foundational understanding of CAT strategies should be obtained. Then, the mapping between CAT characteristics and TCP should be figured out. The concepts and strategies invoked by CAT are available for addressing pragmatic concerns such as understanding relational alternatives, the alignment of radio broadcasters with their audiences, development, difficulties and outcomes in health care, etc. The main strategies identified through CAT include discourse management, interpretability, interpersonal control, and accommodation. Accommodation strategy is the point of focus for this research. It can characterize wholesale realignments of language selection or code patterns, attitudes, and socio structural conditions.

Accommodation is defined as the ability to adjust, revise, or regulate one's behavior

in response to another. There are a variety of reasons for accommodating the communication including but not limited to elicit approval, maintain a positive social identity, and achieve communicative efficiency. However, CAT maintains the following assumptions [177]:

i. speech and behavioral similarities and dissimilarities exist in all conversations,
ii. the manner in which we perceive the speech and behaviors of another person will determine how we evaluate a conversation,
iii. language and behaviors impart information about social status and group belonging, and
iv. accommodation varies in its degree of appropriateness, and norms guide the accommodation process.

Accommodation has two different strategies called Convergence and Divergence. *Convergence* is defined as other-directed strategy since it is deployed during the communication as tendencies to emphasize similarities between the speaker (himself) and the interlocutor. Whereas *Divergence* is defined as self-directed strategy targeted at maintain one's own speaking style without adjustments.

Convergence: Convergence has been defined as the process through which individuals adapt to each other's communicative behaviors to reduce interpersonal differences and improve effectiveness of communication. People can converge in a wide range of communication features for instance, speech rate, self-disclose, information density, and response latency, but they do not necessarily have to converge simultaneously at all of these levels.

Divergence: Divergence represents the opposite direction to convergence, in which

the individual accentuates speech and non-verbal differences between themselves and others. Divergence can take many forms both verbal and non-verbal forms in order to reinforce individual or group identity, maintain integrity, and distance from engaging in lengthy conversation.

The benefits of CAT in increasing communication efficiency and accommodating the differences in ability can meet the challenge of adapting TCP's sending rate in ad hoc networks [110]. More specifically, CAT is the primary theory to be applied in proposing the Contention Avoidance Mechanism (CAM) for TCP Sintok in ad hoc networks. CAM's main objective is to provide TCP with contention avoidance and control capabilities so that TCP can control the growth of congestion window and avoid contention.

On the one hand, CAT assumes that individuals bring their background and previous experience into conversations through their speech rate. Transferring this concept to CAM, it is clear that current congestion window represents the previous experience of network condition. Therefore, the congestion window parameter is preserved. CAT also assumes that accommodation is influenced by the way in which individuals evaluate what takes place during a conversation, that is, how people interpret and judge the messages exchanged in conversation. For example, a speaker initially exhibiting a rate of 50 words per minute can move to match another speaker's rate of 100 words per minute or can move to a rate of 75 words per minute [178]. In ad hoc networks, nodes are sharing the same transmission medium. Furthermore, nodes are helping each other to transmit their data to the destination as each node is acting as a host and an intermediate node. Therefore, there is a need to assess and estimate what happens in the shared medium in addition to the path status from the source to the destination. This is the missing part in the current congestion avoidance mechanism. According to

CAT, the proposed contention avoidance mechanism should have another parameter in addition to congestion window to adapt the sending rate; in specific, to control the growth of congestion window during congestion avoidance phase. CAM will evaluate what is happening through the transmission by measuring the contention ratio during the most recent congestion (contention) avoidance phase. Using both parameters, the TCP sender will adapt the sending rate aiming at promoting communicative efficiency between nodes. This goal is considered *convergence* since it seeks an effective communication.

On the other hand, CAT assumes individuals accentuate speech and nonverbal differences between themselves and others in order to distance themselves from their conversation partners. The adjustment done here aims at maintaining positive image of one's in-group and hence to strengthen one's social identity. This is seen as *divergent* because the speaker wants to keep an identity with a reference group. In this case, the interlocutors behave competitively diverging from each other by emphasing the differences in their speech. This feature already exists in TCP where the sending rate (i.e., congestion window size) will be cut once packet loss is detected.

5.3 The Design of Contention Avoidance Mechanism (CAM)

CAT shows the key parameters that should be included in the proposed contention avoidance mechanism. TCP with CAM will converge to accomplish efficient communication. Nodes involved in the transmission process are expected to adjust and adapt the sending rate according to the communicative situation of the wireless channel. The main objectives of CAM are:

- to observe the link utilization ratio in order to maintain a high sending rate at the TCP's sender, with smaller rate variations, which assists to avoid contention

and improves link utilization; and

- to control the size of congestion window to avoid contention while maintaining high link utilization.

In order to determine the contention level (i.e., communicative situation) of the wireless channel between the source node and destination node, Efficient Link Utilization ratio (ELU) is proposed. ELU is estimated based on the Sender Utilization (U_s) of the link in addition to its Neighbors' Utilization (U_n) of the shared medium.

Let us define sender utilization as the number of data packets sent by the source plus the number of ACKs received by the source node during period $(T0)$.

$$U_s = \sum_{T0} PacketSent + \sum_{T0} AckReceived \tag{5.1}$$

Let us refer to neighbors' utilization as the total number of packets lost by the sender during period $(T0)$. GoBackN mechanism is applied here to estimate neighbors' utilization. Where, once packet loss is detected by RTO or 3DUPACKs, all packets in flight are considered lost.

$$U_n = \sum_{T0} PacketSent - \sum_{T0} AckReceived \tag{5.2}$$

From equations 5.1and 5.2, ELU is derived as follows:

$$ELU = \frac{(SenderUtilization - Neighbors'Utilization)}{SenderUtilization} \tag{5.3}$$

$$ELU = \frac{(\sum_{T0} PacketSent + \sum_{T0} AckReceived - [\sum_{T0} PacketSent - \sum_{T0} AckReceived])}{\sum_{T0} PacketSent + \sum_{T0} AckReceived} \tag{5.4}$$

$$ELU = \frac{2(\sum_{T0} AckReceived)}{\sum_{T0} PacketSent + \sum_{T0} AckReceived} \tag{5.5}$$

where $0 < ELU \leq 1$

The ELU ratio is monitored by the sender node only during period $(T0)$. The length of $(T0)$ should be appropriate and dynamic. Therefore, $(T0)$ is set to the congestion (contention) avoidance interval. Upon receiving triple duplicate ACKs, the sender estimates the link contention level through the ELU ratio value as illustrated by Equation 5.5.

During congestion (contention) avoidance, conventional TCP continuously increases its sending rate based on Equation (5.6) until a triple duplicate ACKs is received as an indication of packet loss.

$$W_{new} = W_{current} + \frac{1}{W_{current}} \tag{5.6}$$

Whenever an ACK is received by the destination, the congestion window size is increased by a fix rate equal to $\frac{1}{W_{current}}$. However, it was proven that ignoring the contention and increasing congestion window based on Equation (5.6) will let the window size grow beyond the optimal size leading to high level of contention as a result degrading TCP performance. Therefore, the increase factor should be dynamic and related to the contention status of the path as well as the shared medium status. To address this issue, an increase factor (I) is proposed based on link utilization status. The increase factor should be small if the utilization is high and vice versa. The increase factor (I) is proposed to be calculated as follows:

$$I = 1 - ELU \tag{5.7}$$

where $0 < I \leq 1$, if $I = 1$, this leads to the same window update rule of congestion avoidance. In other words, the proposed scheme is compatible with the conventional congestion avoidance mechanism.

Let us have a look at the following two cases:

i. if ELU is equal to one, this means the link is fully utilized and increasing the window size will not provide a better performance, therefore, increase factor will be zero, and
ii. if ELU is equal to zero, this means the link is underutilized and increasing the window size is required, therefore, the increase factor will be one.

Based on CAT concepts and the above discussion, the new contention avoidance scheme is proposed. Suppose that the current congestion window size is W, then during the contention avoidance stage, the congestion window will be increased based on Equation (5.8):

$$W_{new} = W_{current} + \frac{I}{W_{current}} \tag{5.8}$$

5.4 The Implementation of Contention Avoidance Mechanism (CAM)

The slow start algorithm is used when congestion window is less than slow start threshold (cwnd < ssthresh), while the contention avoidance algorithm will be used when congestion window is greater than or equal to slow start *threshold* (cwnd $\geq$ ssthresh).

During the slow start phase, a TCP sender increases *cwnd* size by at most SMSS bytes for each ACK received that acknowledges new data. Slow start ends when *cwnd* size exceeds or equals *ssthresh* (or, optionally, when it reaches it) or when loss is detected.

During contention avoidance period, *cwnd* is incremented based on the increase factor per Round-Trip Time (RTT), implying linear growth instead of an exponential growth. Contention avoidance continues until loss is observed. One formula commonly used to update *cwnd* during contention avoidance is given in Equation (5.8). This adjustment is executed on every incoming non-duplicate ACK. Equation (5.8) provides an acceptable approximation to the underlying principles of CAT.

When the TCP sender detects packet loss using the 3DUPACKs, the value of ELU will be calculated. Then, the increase factor of the next contention avoidance phase will be derived. After that, fast retransmit and fast recovery will be invoked to recover from

the loss. The pseudo-code to illustrate the CAM steps is presented in Algorithm 5.1:

Algorithm 5.1 UPON receiving 3rd DUPLICATE ACK

Step 1. Calculate the value of ELU:

$$ELU = \frac{2(\sum_{T0} AckReceived)}{\sum_{T0} PacketSent + \sum_{T0} AckReceived}$$

Step 2. Calculate Increase Factor (I) value

$$IncreaseFactor = 1 - ELU$$

Step 3. Invoke Fast Retransmit and Fast Recovery Mechanism

During the contention avoidance phase, on the one hand, whenever the TCP sender sends a packet, it increases the total packets sent by one. On the other hand, whenever ACK is received by the sender, the total ACK received will be increased by one. Furthermore, the *cwnd* value will be adjusted using the recent increase factor (I) value and current *cwnd* value based on the Equation (**??**). The pseudo-code to illustrate these steps is presented in Algorithm 5.2:

Algorithm 5.2 DURING CONTENTION AVOIDANCE

Step 1. Increase the total number of PacketSent by one once packet sent

$$PacketSent = PacketSent + 1$$

Step 2. Increase the total number of AckReceived by one once New Ack Received

$$AckReceived = AckReceived + 1$$

Step 3. Increase congestion window based on the calculated increase factor (I)

$$W_{new} = W_{current} + \frac{I}{W_{current}}$$

As shown in the Algorithms 5.1 and 5.2, the implementation of CAM requires slight modifications at the TCP layer without any feedback from the lower layer or intermediate node. This amendment includes adding two counters for the total number of packets sent and ACKs received; and one variable for the increase factor. These variables will be used by the sender to monitor the contention status during the contention avoidance phase. Therefore, CAM can be deployed and implemented easily with very minimum overload/overhead to TCP operations.

5.5 Verification and Validation of CAM

The verification's primary concern is to ensure that the proposed CAM has been programmed correctly in the NS-2 simulation environment, and it has been implemented properly on the computer. Verification was conducted following the techniques mentioned in Chapter Three. After the verification process, a snapshot of CAM implemen-

tation in Eclipse is illustrated in Figure 5.3 and confirmed the following:

- CAM is programmed correctly, and
- CAM implementation in NS-2 does not contain any errors or bugs.

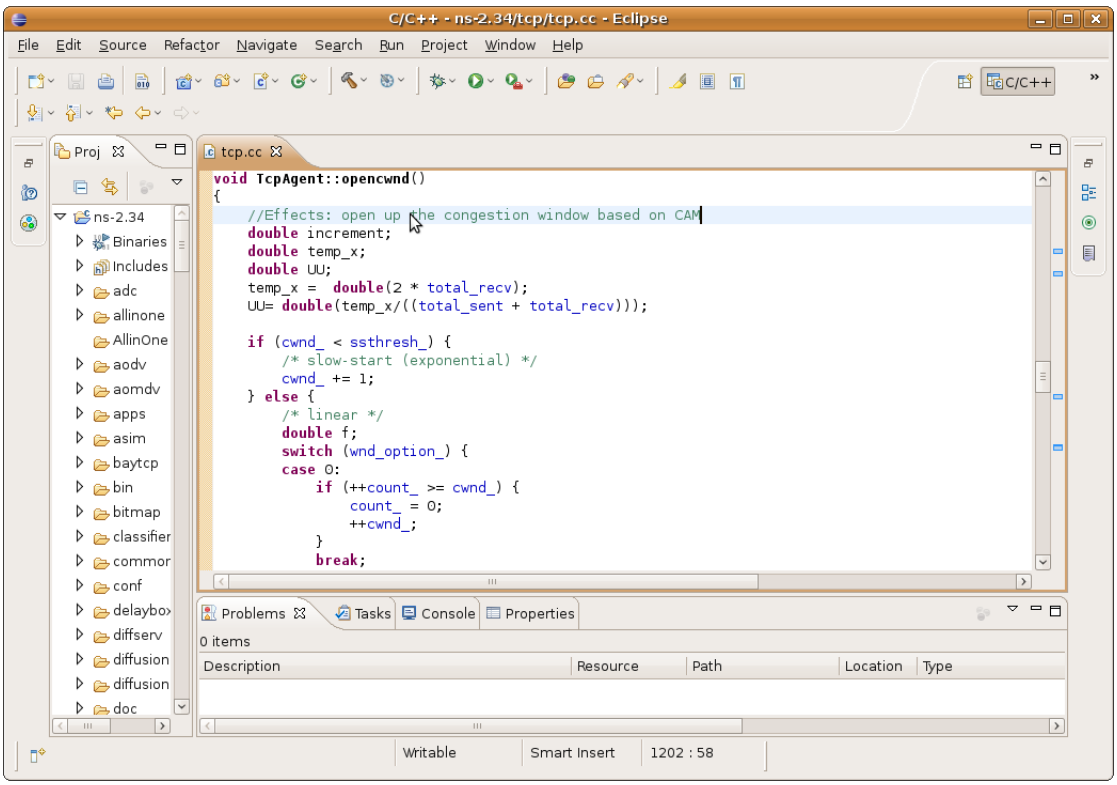


Figure 5.3: Implementation of CAM in Eclipse

Next, CAM was validated to ensure that it meets its intended requirements in terms of adjusting the sending rate based on the status of the shared medium. The validation of CAM focused on examining the relationship between congestion window size and the number of hops in a variety scenarios of multihop wireless ad hoc networks. This is because the congestion window size is closely related to route length rather than to the Bandwidth Delay Product (BDP). The obtained results using CAM are compared to the standard TCP congestion avoidance and the optimal window size proposed in theory based on the number of hops.

The validation was conducted using NS-2 simulator. The simulation default settings have been provided in Chapter Three. The AODV protocol was used as it is the main routing protocol used in the evaluation of various transport protocols. In addition, since AODV is reactive, no routing overhead will be imposed after the route from the source to the destination is established. A single TCP flow (of 500 seconds) was run over two different network topologies with various number of hops in chain and grid topologies. The result was obtained from the average of ten runs with different random number seeds.

5.5.1 Chain Topology

It is quite common to have chain-like topology in ad hoc networks. Moreover, chain topology offers only one path from the source node to the destination. For instance, the successive transmission of a single TCP data flow interferes with each other as they transfer toward the destination. Furthermore, data flow interferes with the ACK flow (self-contention). So, chain topology impact on TCP performance can be clearly observed as it is the most resource-limited topology. Therefore, CAM is validated over a different number of hops in chain topology with a zero bit error rate. A single TCP flow runs from the first node (S) of the chain and forwarded through the intermediate nodes to the last node (D), as illustrated in Figure 5.4. All nodes are stationary and separated by 200 m. The nodes communicate with identical, half-duplex wireless radios. The assumption is that the source node has data packets to send, while the destination node has ACK packets to send, while intermediate nodes have both data and ACK packets to send. The detailed information of the simulation model is stated in Table 3.2.

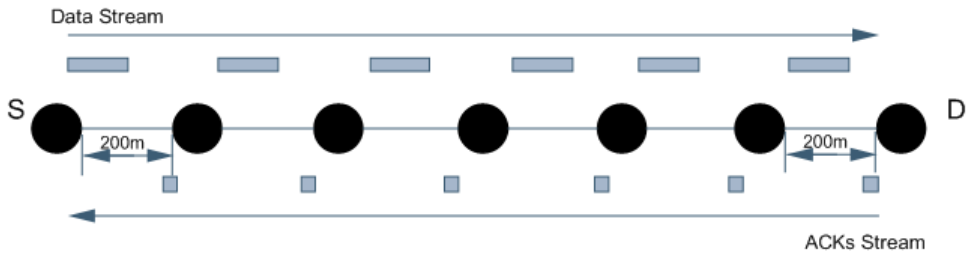


Figure 5.4: Chain Topology with 6-hop

The proposed Contention Avoidance Mechanism (CAM) was validated over chain topologies with six to ten node lengths (i.e., five to nine hops). The conventional TCP NewReno congestion avoidance was used as a reference of the performance bound. Figure 5.5 shows the average congestion window size of CAM and TCP NewReno over networks of chain topology within five to nine hops. Overall, TCP NewReno congestion window size is considerably higher than CAM in all scenarios. The gap is greatest in short chains and reduces/declines in long chains. TCP NewReno window size fluctuates between seven packets in 5-hop and five packets in 9-hop chains, while CAM window size increases gradually from two packets to three packets. To conclude, CAM gently adapts the growth of congestion window size as a result of using the proposed increased factor. In contrast, TCP NewReno congestion avoidance could not control the growth of congestion window size, instead the window size increased and decreased randomly without an appropriate trend in line with the network status.

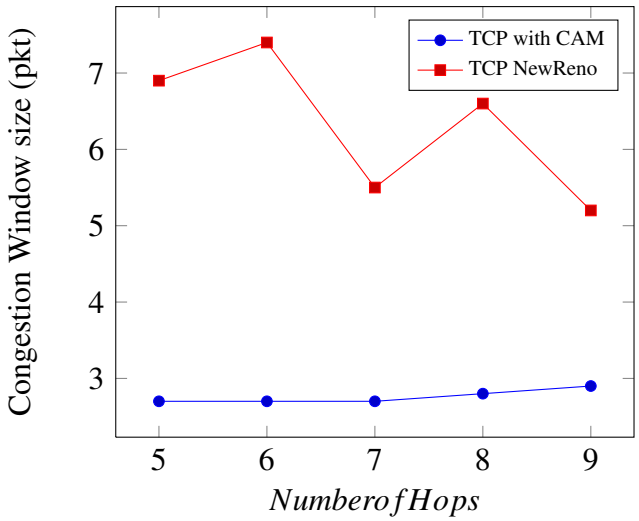


Figure 5.5: The Impacts of Chain Length in # of hops

Further validation was conducted by comparing CAM average window size with the optimal window size obtained from real experiment results, as reported in Fu et al. [28]. The simulation experiment was run in 6-hop and 7-hop chain topologies. The same settings and parameters adopted in [28] were applied to the CAM simulation experiment. Table 5.1 shows the optimal congestion window size in packets, average window size of TCP NewReno, and average window size of TCP NewReno using CAM in chain topologies with 6-hops and 7-hops. As shown in the table, the optimal window size is equal to two packets in 6-hop and 7-hop chain topologies. In contrast, TCP NewReno window size is higher than the optimal value with approximately seven packets in 6-hop chain and six packets in 7-hop chain. CAM, meanwhile, manages to keep average window size very close to the optimal size with 2.7 packet size in both cases. This table shows that CAM with the new increase factor is capable of controlling the growth of the congestion window value and keeps it very close to the optimal window size. As a result, this would lead to improve the performance of TCP, as will be illustrated later in Chapter Six.

Table 5.1: Optimal and Measured Congestion Window Size

Topology	Optimal Win Size	Avg. Win Size (*)	Avg. Win Size (+)
6-hop Chain	2	7.4	2.7
7-hop Chain	2	5.5	2.7

* refers to TCP with congestion avoidance (original) mechanism

+ refers to TCP with contention avoidance (proposed) mechanism

5.5.2 Grid Topology

The previous section focused on validating CAM in chain topology while data packet contends with each other as well as the ACK packet on the reverse path. This section will present the CAM validation results using grid topology. CAM has been tested over different scales of grid topology (5x5, 6x6, 7x7, and 8X8) with $PER = 0$. The optimal value of congestion window obtained from dividing the number of hops between the source node and the destination node to four is used as a benchmark. In this scenario, there was only one TCP connection flow from the first node in the grid (sender) to the last node in the grid (destination), as illustrated in Figure 5.6. Thus, packets are lost due to contention among data and ACK packets only. All nodes are stationary, separated by 200 m, and they communicate with identical, half-duplex wireless radios. The detailed information of the simulation model is stated in Table 3.2.

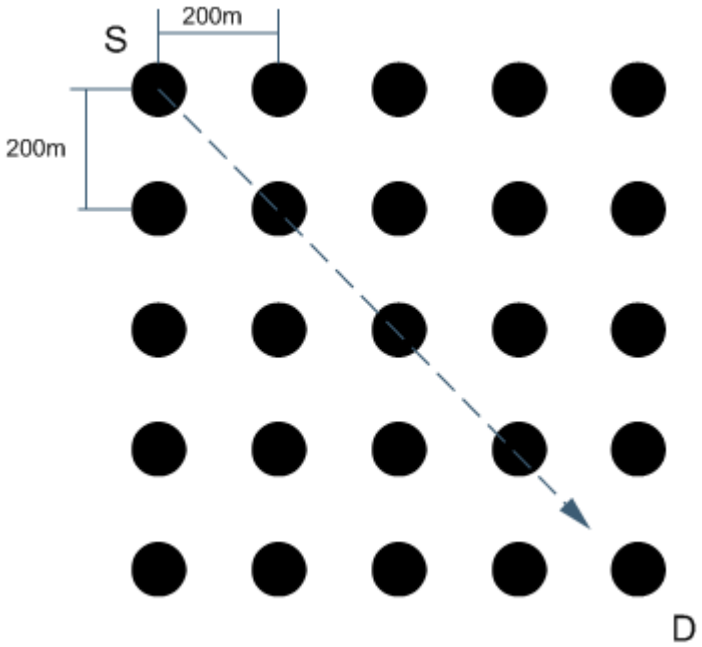


Figure 5.6: Grid Topology with 5X5 Nodes

Figure 5.7 illustrates the average window size of TCP with CAM as compared with optimal window size estimated based on the number of hops (theory) in grid topology with different sizes. Overall, the graph shows gradual increased in window size of both TCP with CAM and theory as the number of hops increases. Furthermore, there is a slight difference in congestion window size between TCP with CAM and theory. The difference becomes almost 1.9 in grid topology with 6X6 nodes.

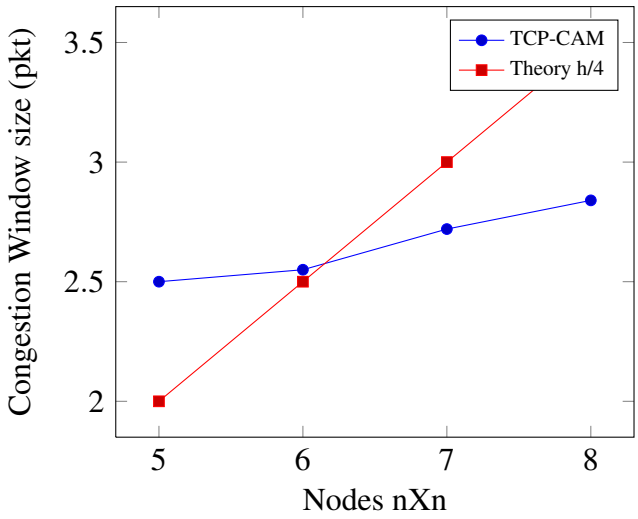


Figure 5.7: TCP with CAM versus Theory Congestion Window Size in Grid with nXn Nodes

The previous observation is further analyzed in Table 5.2. Table 5.2 presents the difference between average congestion window size of TCP with CAM and optimal window size according to theory. It is evident that the difference declines in percentage from 22.22% in 5X5 grid, to 1.9% in 6X6 grid topology. Then, the difference increases in percentage from 9.7% in 7X7 grid to 20.82% in 8X8 grid. This observation confirms that CAM responds to contention as intended and keeps sending rate close to the optimal window size, even when the grid size increases.

Table 5.2: TCP with CAM versus Theory Congestion Window Size in Grid

Grid Size	**CAM Win Size**	**Theory Win Size**	**Difference (%)**
5 X 5	2.5	2.0	22
6 x 6	2.55	2.5	2
7 X 7	2.72	3.0	10
8 X 8	2.84	3.5	21

To gain a deeper understanding on CAM performance, the previous scenario is repeated using the standard TCP NewReno. Figure 5.8 shows the average window size of TCP NewReno as compared to optimal window size calculated based on the number of hops (theory) in grid topology with different sizes. In brief, the graph shows gradual increase in optimal window size (theory) as the number of hops increases, whereas the average window size of TCP NewReno declines. In 5X5 grid, for instance, the average window size of TCP NewReno is three times more than reported by the theory value. The gap decreases when the network size increases (i.e., the number of hops between the sender and receiver increases). In 8X8 grid topology, TCP NewReno average window size is 5.2 packets while window size as estimated in theory is around 3.5 packets.

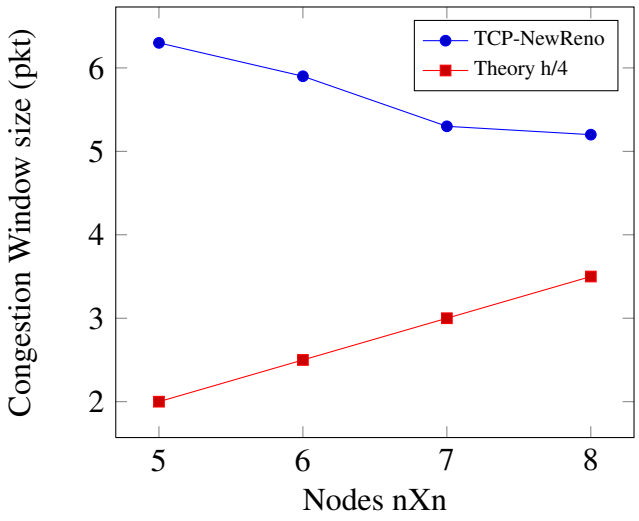


Figure 5.8: TCP NewReno versus Theory Congestion Window Size in Grid with nXn Nodes

Table 5.3 illustrates the difference between average congestion window size of TCP NewReno and optimal window size according to theory in different scales of grid

topology. It is clear that there is a significant difference in congestion window size between TCP NewReno and theory. In 5X5 grid, the difference in congestion window value is (103.01%). In 6X6 grid, the difference becomes (80.95%). Then, the gap decreases to reach about (39.08%) in 8X8 grid topology. However, it is still considerably high and this leads to TCP performance degrading in ad hoc networks.

Table 5.3: TCP NewReno versus Theory Congestion Window Size in Grid

Grid Size	**NewReno Win Size**	**Theory Win Size**	**Difference (%)**
5 X 5	6.3	2.0	103
6 x 6	5.9	2.5	81
7 X 7	5.3	3.0	55
8 X 8	5.2	3.5	39

As a result based on Table 5.2 and Table 5.3, it it can be seen the CAM adapts congestion window size closer to optimal window as compared to traditional TCP NewReno. In 5X5 grid, the difference between CAM and theory is (22.22%) as compared to (103.01%) in the case of TCP NewReno. The difference reduces as the grid size increase. However, the difference between CAM and theory gets smaller and reaches the smallest value with (1.9%) in 6X6 grid topology. On the other hand, TCP NewReno difference reaches to (80.95%) which is considered very high in contrast to CAM. Finally, the difference becomes (20.82%) between CAM and Theory in contrast to (39.08%) for TCP NewReno.

5.6 Summary

Chapter Four highlighted the need to adapt the sending rate based on the current medium conditions. This chapter explored the applicability of one matured human theory named Communication Accommodation Theory (CAT) in computer communication. CAT shows significant impact in improving and enhancing the quality of human communication and have potential application in computer network. This Chapter also revealed the state of the art of CAT theory by reviewing all articles published and indexed in Scopus database between 1990 and 2012. Then, CAT assumptions and fundamental strategies are explained to identify all matching elements between CAT and TCP congestion control. After that, the integration of CAT theory in TCP congestion control was elaborated in greater detail.

Next, Contention Avoidance Mechanism (CAM) for TCP Sintok was designed based on CAT recommendations. Two new parameters were proposed, referred to as Efficient Link Utilization (ELU) and Increase Factor (I). These parameters are used to accommodate the sending rate in order to increase network resource utilization and avoid contention.

Moreover, the proposed mechanism was implemented in the simulation tool. The final section is the verification and validation of the proposed mechanism. The proposed mechanism was verified inside the NS-2 simulator by observing the error and bugs in the code. The validation was accomplished by examining CAM average window size as compared to the result of real testbed experiments obtained by other researchers. The validation requirements also were fulfilled by comparing the average window size of CAM, conventional TCP NewReno, and the theory. Finally, the obtained results prove the applicability of CAT in the proposed computer communication in general, and in CAM application in specific.

The next Chapter introduces the new transport protocol called TCP Sintok by combining the proposed two mechanisms LDM and CAM. The evaluation of the proposed TCP Sintok will be carried out by comparing its performance with previously selected proposals. The comparison and results will be presented in Chapter Six using the NS-2 simulation environment.

CHAPTER SIX
TCP SINTOK PERFORMANCE ANALYSIS

Previously, the design of LDM and CAM mechanisms were covered in Chapter Four and Chapter Five, respectively. This chapter introduces a new TCP named TCP Sintok, which is specifically designed for ad hoc networks. Furthermore, this chapter discusses the new TCP design and implementation issues then presents detailed performance evaluation of TCP Sintok. The chapter starts by describing the structure of TCP Sintok in Section 6.1. In Section 6.2, the implementation of TCP Sintok is elaborated in greater details. Results of the performance evaluations of TCP Sintok over the standard TCP NewReno are discussed in Section 6.3. TCP Sintok is tested further in Section 6.4 by comparing its performance to recent related works. Lastly, discussion on the performance evaluation is deliberated upon in Section 6.5.

6.1 TCP Sintok: An Overview

The ultimate aim of this thesis is to propose TCP Sintok for ad hoc networks. TCP Sintok is an end-to-end transmission control protocol based on TCP NewReno. It uses the same conventional connection establishment and tear-down approach of TCP NewReno. Furthermore, it adopts TCP NewReno congestion control at the sender whenever congestion state is detected at the network. In addition, TCP Sintok incorporates the two proposed mechanisms LDM and CAM, as elaborated in Chapter Four and Chapter Five to improve TCP performance in ad hoc networks. Additionally, TCP Sintok makes several changes and extensions at TCP sender to cope with the new behaviors of ad hoc networks. Finally, it should be stated that the control actions adopted at TCP Sintok sender are not necessarily the finest, however they are great steps towards improving TCP's performance over ad hoc networks.

TCP Sintok congestion control comprises the following three states: slow start, contention avoidance, and fast retransmit/recovery, as shown in the Finite State Machine (FSM) of TCP Sintok congestion control in Figure 6.1. Each state in the FSM diagram addresses a particular condition in ad hoc networks and it is described as follows:

Slow Start:

This is the initial state after connection establishment in TCP Sintok and resembles the conventional function of TCP NewReno. TCP Sintok enters this state when 1) TCP connection begins and 2) timeout occurs. In the slow start, the value of congestion window *cwnd* is initialized to one Maximum Segment Size (MSS) and increases by one MSS for each incoming new ACK. This process doubles the congestion window every RTT. Thus, TCP congestion window starts slowly/linearly. However it grows exponentially until threshold, *ssthresh*, is reached. Slow start ends when 1) *cwnd* equals to *ssthresh* and TCP Sintok transitions into contention avoidance state, 2) there is a loss event (i.e., congestion or route failure) indicated by a timeout, 3) three duplicate ACKs are detected and TCP performs a fast retransmit and enters fast recovery state.

Contention Avoidance:

TCP Sintok increases *cwnd* value more cautiously when it is in contention avoidance state. TCP Sintok enters this state when *cwnd* value is greater or equal to *ssthresh*. TCP Sintok adopts proactive approach using CAM to control the growth of congestion window and avoid contention in the shared medium. In the contention avoidance state, TCP Sintok increases the value of congestion window by $(I/cwnd)$ rather than $(1/cwnd)$ every time a transmitted segment is acknowledged. Furthermore, upon the ACK arrival at the sender, RTT value will be saved in the sender sample space as stated

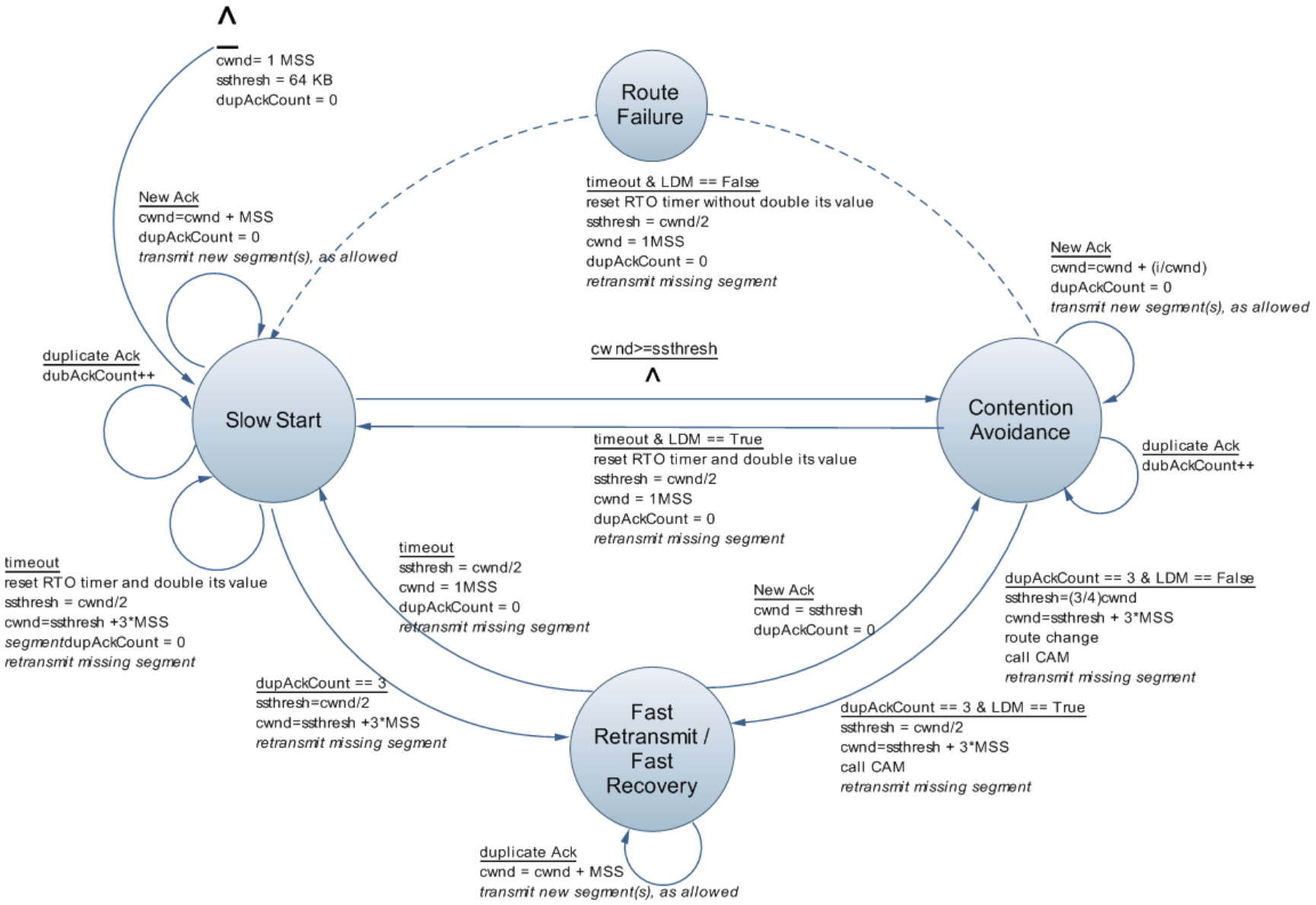


Figure 6.1: Finite State Machine of TCP Sintok Congestion Control

in LDM (in addition to CAM as well as the conventional operations). Moreover, the total number of received segments will increase by one. The sender maintains these values and proceeds with normal operations. Contention avoidance state ends when loss is triggered by either a retransmission timeout (RTO) or third duplicate ACK being received.

Once packet loss is detected, the sender identifies the network conditions using LDM. In the case of retransmission timeout, if the delay trend is increasing; conventional congestion control action will be invoked to react and recover. Otherwise, route failure status is detected and TCP should recover in the normal way, except the RTO back off mechanism will not be called to avoid increasing RTO timer value and reduce the responsiveness of TCP. Hence, TCP will be in a probing state, where it transmits a data packet and waits for its new ACK signaling that a new route is established. The pseudo-code to illustrate these mechanism steps are presented as follows:

Algorithm 6.1 Sender Side: Upon Retransmission Timeout

Step 1. Check the network state using LDM;

Step 2. **if** Network status = = CONGESTION **then {**

Step 3. Invoke identical TCP NewReno congestion control to reccover

Step 4. Invoke RTO back off algorithm and double RTO value

Step 5. **} else { //** It is ROUTE FAILURE

Step 6. Invoke identical TCP NewReno congestion control to reccover .

Step 7. Reset RTO timer without increasing its value.

Step 8. **}**

In the case of three duplicate ACKs, if the delay trend is increasing then contention (congestion) is detected, and the identical TCP NewReno congestion control will be invoked. Otherwise, if route change status is detected, fast retransmit and fast recovery

will be invoked with exception that the new *cwnd* will be set to $(3/4)$ of its current value. After that, CAM will be invoked to calculate the Increase Factor of the next contention avoidance phase. The pseudo-code to illustrate these mechanism steps are presented as follows:

Algorithm 6.2 Sender Side: Upon 3rd Duplicate ACKs

Step 1. Check the network status using LDM;

Step 2. **if** Network status = = CONGESTION **then {**

Step 3. Invoke identical TCP NewReno congestion control to reccover

Step 4. **} else { //** It is ROUTE CHANGE

Step 5. Invoke identical TCP NewReno congestion control to reccover

Step 6. cwnd = cwnd * 3/4

Step 7. **}**

Step 8. Invoke CAM to calculate the Increase Factor for the next contention avoidance phase.

Fast Retransmit/ Fast Recovery:

TCP Sintok enters the fast retransmit and fast recovery state when three duplicate ACKs are detected in any of the other two states. In this state, the congestion window value is increased by one MSS for every duplicate ACK received for each of the missing segment that caused the TCP to enter the fast recovery state. TCP Sintok exits this state when 1) new ACK is received that acknowledges all missing segments, or 2) there is a loss event indicated by a timeout.

6.2 The Implementation of TCP Sintok

TCP Sintok was implemented in the NS-2 (2.34) environment based on TCP NewReno code. LDM and CAM mechanisms were plugged into TCP sender-side code. Incor-

porating LDM and CAM aims to help TCP sender to take more accurate control and enhance TCP performance significantly. Some implementation details are provided as follows:

Sender Side: Sample space of five doubles is allocated for storing the latest four delay samples in addition to smooth RTT. Furthermore, two variables are declared to save the total number of sent segments and total received segments during the contention avoidance phase. Pairewise Comparison Test (PCT) related calculation is performed once packet loss is triggered by RTO or third duplicate ACKs. In addition, the code that handles third duplicate ACKs and retransmission timeouts is extended to follow TCP Sintok design.

Receiver Side: Receiver will echo the time value stored in the TCP segments header in its ACK header. This field records the time when this segments was sent and it will be used by the sender to calculate the end-to-end delay value.

TCP Sintok Source code, as shown in Figure 6.2, will be made available at the Inter-NetWorks Research Lab website: "http://internetworks.my" after the final presentation of this thesis. In addition, the author is ready to lend a hand and advice on the implementation issues of TCP Sintok in any environments besides the NS-2 simulator.

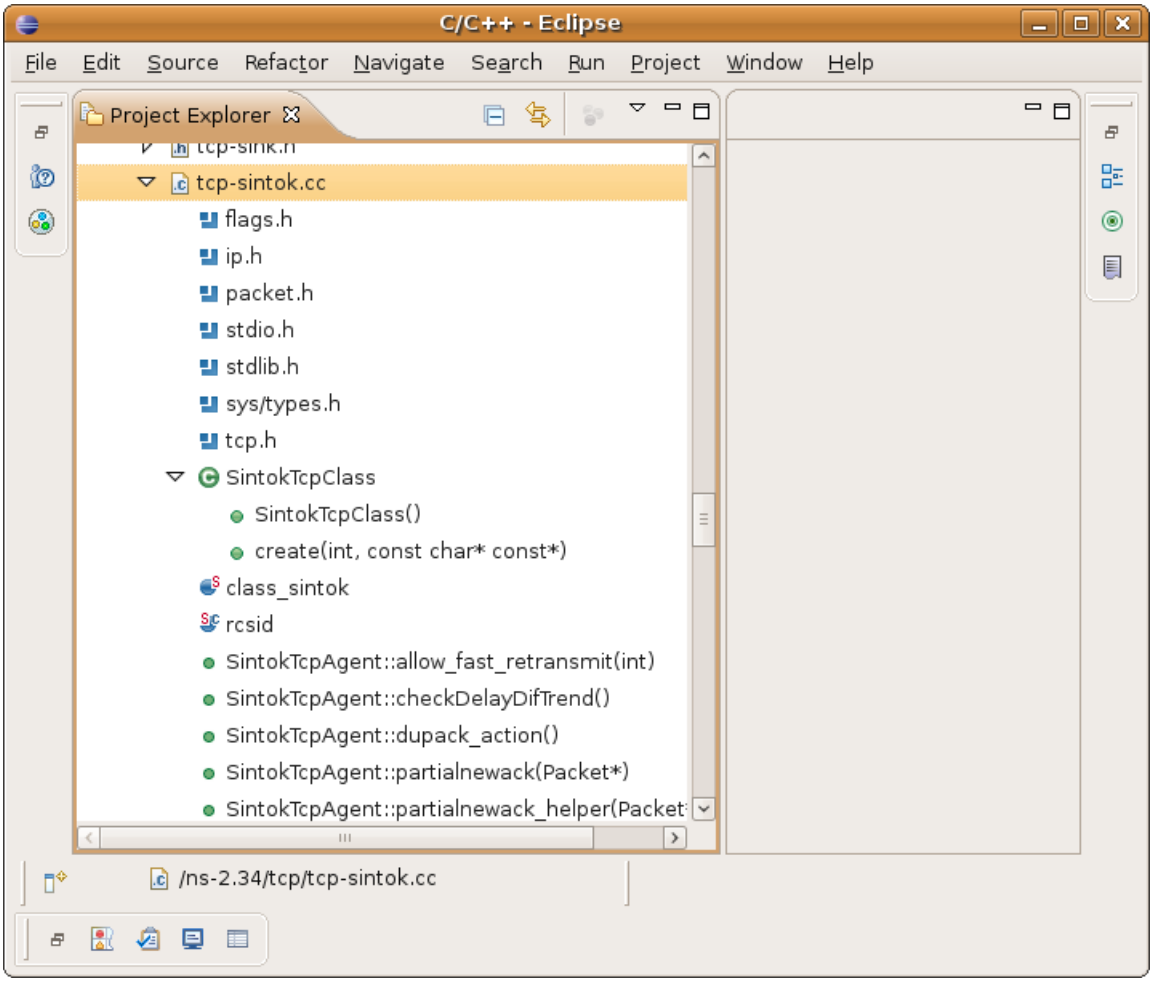


Figure 6.2: TCP Sintok in Eclipse

6.3 Performance Evaluation of TCP Sintok

The main goal of these experimental evaluations is to test the ability of TCP Sintok to react in ad hoc networks as compared to the standard TCP NewReno. The attempt was not to measure TCP Sintok performance on a particular workload captured from a real network, but rather to measure its performance under a range of network conditions and scenarios. To achieve this goal, the focus was set on three network topologies, namely chain, grid, and random with a variety of workloads and network conditions.

6.3.1 Chain Scenario

It is quite common to have chain-like topologies in ad hoc networks. Moreover, chain topology is the most resource-limited case that offers only one path so that its impact on TCP performance can be clearly observed. Therefore, TCP Sintok was tested over 5-hop and 6-hop chain topologies with varying channel error rates ranging from (0%) to (10%). Single TCP flow runs from the first node (S) of the chain to the last node (D) of the chain as illustrated in Figure 6.3. All nodes are stationary and separated by 200 m, where they communicate with identical, half-duplex wireless radios. The detailed information of the simulation model is stated in Table 3.2.

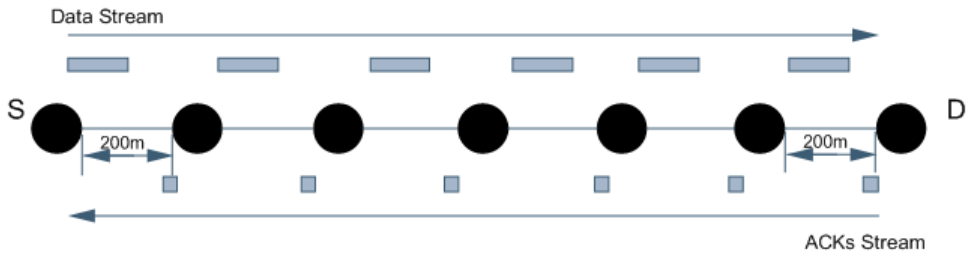


Figure 6.3: Chain Topology with 6-hop

Figure 6.4 shows average throughput of TCP Sintok and TCP NewReno in a 5-hop chain topology with channel error rates between (0%) to (10%). Both TCPs experienced a considerable decline as the channel error rate increases. When channel error rate is equal to zero, TCP Sintok average throughput is equal to (219.8 Kbps). After that, there was a dramatic decline between channel error rates equaling to (3%) and (7%); then it stabilized at about (12 Kbps) with channel error more than (8%). Over the same channel error rate, TCP NewReno started at (188.5 Kbps) with (0%) error rate and dropped dramatically to (6 Kbps) at (10%) channel error rate. In brief, TCP Sintok achieves higher average throughput than TCP NewReno over all presented channel error rates. Remarkably, the performance gap between TCP Sintok and TCP NewReno increases with the channel error rate to reach almost double at channel error

equal to (10%).

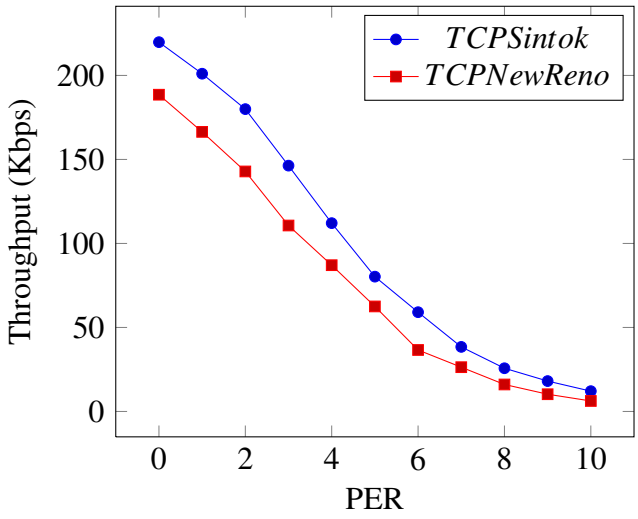


Figure 6.4: Throughput in Chain Topology with 5-hop

Figure 6.5 shows the average throughput in Kbps of TCP Sintok and TCP NewReno in 6-hop chain topology with channel error rates between (0%) to (10%). The graph demonstrated similarities between TCP Sintok and TCP NewReno throughput over 5-hop and 6-hop chain. Over all, average throughput declines sharply as the channel error rate rises. However, the average throughput of TCP Sintok achieves approximately two times higher than TCP NewReno at (10%) channel error.

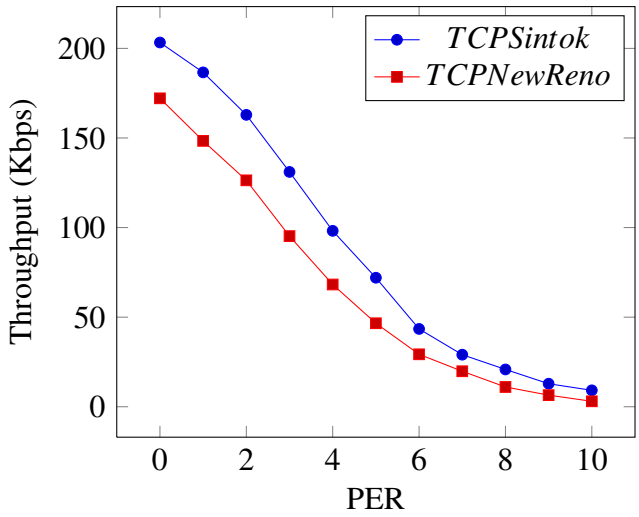


Figure 6.5: Throughput in Chain Topology with 6-hop

From the above Figures 6.4 and 6.5, it is clear that average throughput decreases as channel error rate increases. Yet, TCP Sintok achieves better throughput than TCP NewReno in all previous cases. The next scenario is to study TCP Sintok in a grid topology with different number of hops.

6.3.2 Grid Scenario

As mentioned previously, the grid topology has much more redundancy that provides alternative routes and back-up resources as compared to chain topology. In this subsection, TCP Sintok has been tested over different scales of grid topology (5X5, 6X6, 7X7, 8X8, and 9X9) with ($ChannelErrorRate = 5\%$). Two crossing TCP flows coexist to see the possible performance improvement. An example of grid topology with (5X5) nodes is shown in Figure 6.6. All nodes are stationary and separated by (200 m), where they communicate with identical, half-duplex wireless radios. The detailed information of the simulation model is stated in Table 3.2.

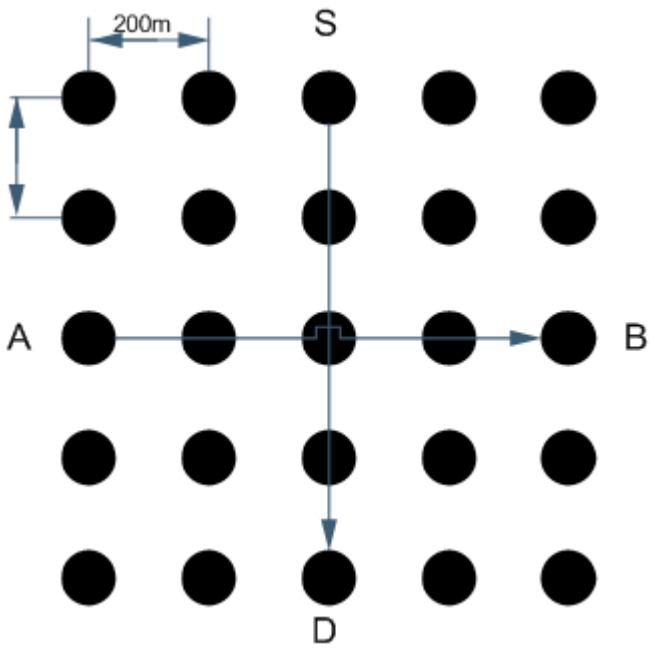


Figure 6.6: Grid Topology with 5x5 Nodes

Figure 6.7 illustrates the average throughput in Kbps of TCP Sintok and TCP NewReno over grid topology with different sizes scale from (5X5) to (9X9) nodes. The general trend of TCP Sintok and TCP NewReno was a decline in the average throughput as the number of hops increases between the sender and the receiver. However, TCP Sintok achieves higher throughput in all grid scales. For instance at (5X5) grid, the average throughput of TCP Sintok is (73.8 Kbps) while TCP NewReno is (63.2 Kbps). Approximately, TCP Sintok achieves (16.6%) higher throughput compared to TCP NewReno. As the number of hops increases, TCP Sintok manages to accomplish higher throughput. At (9X9) grid, TCP Sintok achieves (33%) higher than TCP NewReno. Overall, it can be seen that average throughput of TCP Sintok was far higher than the TCP NewReno in all grid sizes.

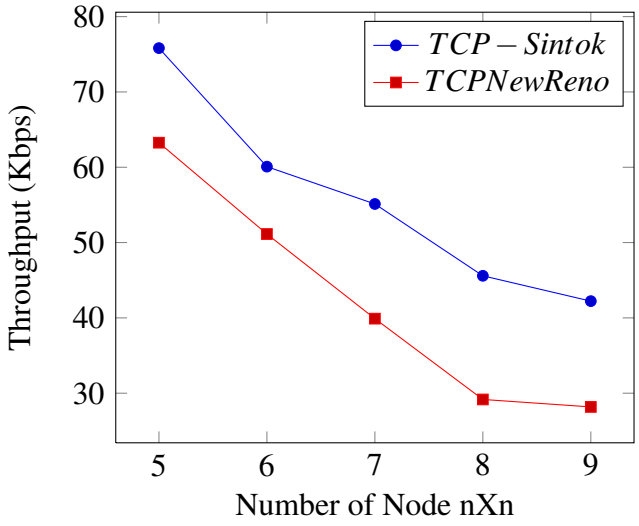


Figure 6.7: Throughput in Grid Topology with nXn Nodes

Figure 6.8 presents the average end-to-end delay of TCP Sintok and TCP NewReno over different grid size. As far as TCP NewReno is concerned, the average delay fluctuated between (198.7 ms) and (359.4 ms). In contrast, TCP Sintok average delay increases gradually from approximately (119.7 ms) to around (312.4 ms). Interestingly, TCP Sintok reduces end-to-end delay significantly as compared to TCP NewReno.

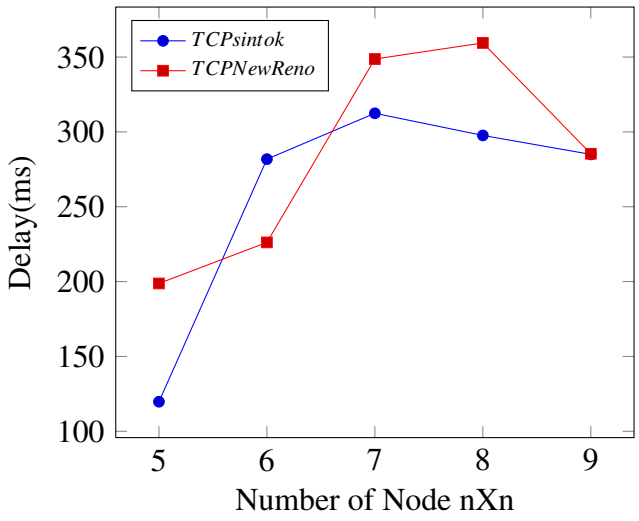


Figure 6.8: Delay in Grid Topology with nXn Nodes

6.3.3 Random Scenario

In order to model a more realistic network environment, TCP Sintok was evaluated over a random topology of (30) mobile nodes moving according to Random Way Point Mobility Model over a rectangular area (400m X 800m) for (500 sec) of simulation time. The node speeds were varied from (5, 10, 15, and 20 m/s) with pause time (0), and AODV was used as a routing protocol. A rectangular area was selected in order to force the use of longer routes between the source node and the destination node. All nodes were randomly distributed. The detailed information of the simulation configuration and parameter values are provided in Table 3.2.

The simulations were performed under moving nodes with 5% channel error rate to represent a rather noisy wireless channel and three competing TCP flows in addition to the main one to introduce contention in the network. Three different performance metrics, which are throughput, delay, and jitter were analyzed to study the performance of TCP Sintok.

Figure 6.9 shows average throughput of the main flow for TCP Sintok and TCP NewReno over a random topology. The vertical axis represents the average throughput in Kbps. The horizontal axis represents the speed from (5 m/s) to (20 m/s). In (5 m/s) speed, TCP Sintok throughput started at (368 Kbps), reached a peak in (10 m/s) of (391.6 Kbps). This is followed by dramatic decline to its lowest value of (320 Kbps) at (20 m/s). TCP NewReno throughput over the same speeds followed a similar trend, but with lower values. Also both TCPs suffered from performance degradation as the speed increases. However, there was considerable increase in TCP Sintok throughput over TCP NewReno in all speed cases.

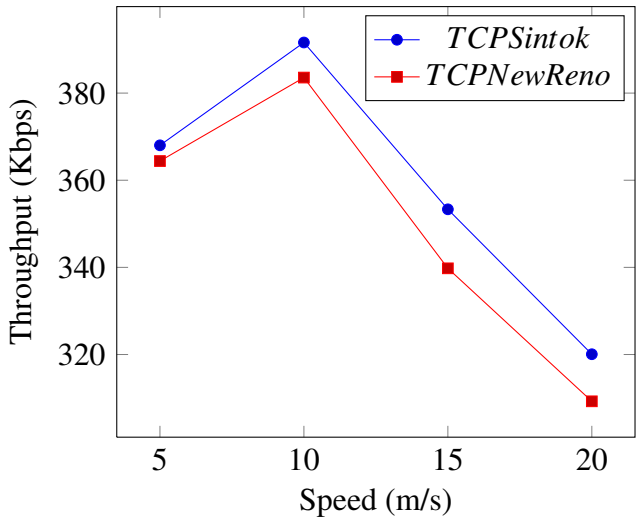


Figure 6.9: Throughput in Random Topology

Figure 6.10 presents average end-to-end delay of TCP Sintok and TCP NewReno with a variety of node movement speeds. As far as TCP NewReno is concerned, the average delay fluctuated between (184 ms) and (175 ms). In contrast, TCP Sintok delay increased gradually from approximately (31 ms) to around (40 ms) on speed (20 m/s). Interestingly, TCP Sintok reduces end-to-end delay significantly as compared to TCP NewReno.

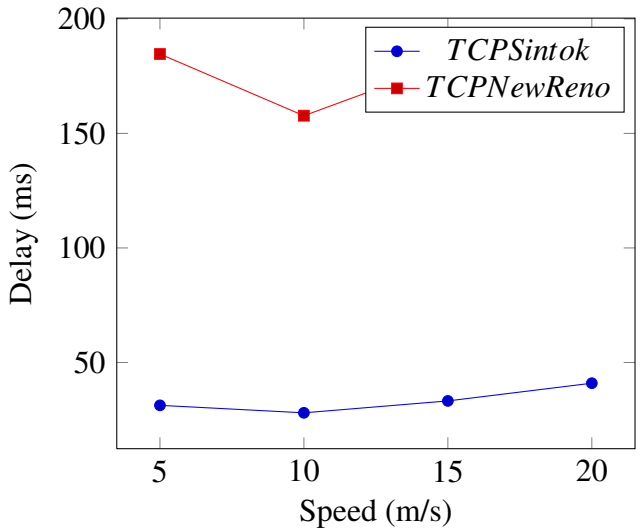


Figure 6.10: Delay in Random Topology

The next Figure 6.11 illustrates average delay jitter differences of TCP Sintok and TCP NewReno over four different speeds. Although both groups showed a gradual rise in jitter as the nodes speed up, except for a slight drop at speed (10 m/s), TCP NewReno seemed to have higher jitter than TCP Sintok. In brief, TCP Sintok achieves between (29.6%) and (36.47%) lower than TCP NewReno jitter.

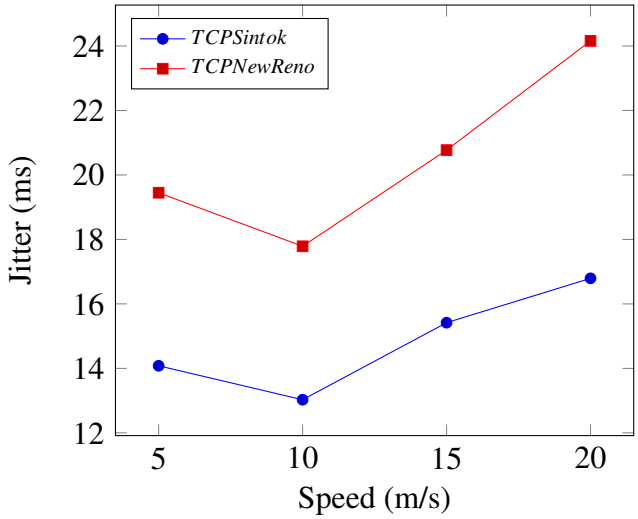


Figure 6.11: Jitter in Random Topology

As a conclusion, TCP Sintok and TCP NewReno were compared in random topology and results are presented in Figures 6.9 to 6.11. The analysis shows that both TCP Sintok and TCP NewReno behave similarly in the dealing with various speeds and network conditions. TCP Sintok achieves slightly higher throughput when compared to TCP NewReno, and also TCP Sintok reduces significantly the average delay and average Jitter experienced by FTP traffic. In the next section, the analysis is extended to compare the performance of TCP Sintok with two recent proposals for ad hoc networks called ADTCP and ELFN.

6.4 Performance Improvement of TCP Sintok

While Section 6.3 focused on studying the ability of TCP Sintok to react in different network conditions as compared to the conventional TCP NewReno, this section aims to evaluate the performance of TCP Sintok relative to current TCP proposals for ad hoc networks. Two proposals named ADTCP and ELFN were chosen to compare TCP Sintok performance through an extensive simulation, since any performance gained that is close to ADTCP or TCP ELFN would indicate the effectiveness of any proposal.

For all the simulation results discussed in this section, all settings and parameters are kept identical to those in [1]. Thirty nodes move randomly in a (400m X 800m) topology. The nodes follow Random Way Point Mobility Model in which the pause time is set to zero so that each node is constantly moving. Dynamic Source Routing (DSR) is used as routing protocol. The nodes communicate with identical, half-duplex wireless radios. The data transmission and data packet size is set to (2 Mbps) and (1460 bytes), respectively. Each value presented in any graph is the average of ten experiments and each simulation experiment is run for $(300sec)$. In addition to TCP flow, three competing UDP/CBR flows are run to introduce congestion within the time intervals [50,250], [100,200], and [130, 170], respectively. Each UDP flow transmits at (180 Kbps). Three different scenarios will be carried out, the first is mobility only, second mobility with 5% channel error rate, and final scenario is combined mobility with 5% channel error rate in addition to three UDP flows.

6.4.1 TCP Sintok versus ADTCP

ADTCP is the first multi-metric end-to-end approach and it has served as a basis for many later approaches. ADTCP was tested not only in simulation environment but real testbed as well; and it improves TCP performance significantly in both cases. Therefore, comparing the performance of any proposal to ADTCP is an essential step to

provide support for the new proposal. This section evaluates TCP Sintok performance as compared to ADTCP in the NS-2 simulation environment.

6.4.1.1 Mobility Scenario

This scenario analyzes the performance of TCP Sintok and ADTCP in random topology with single TCP flow and zero error rate. Figure 6.12 compares the average throughput of TCP Sintok and ADTCP over (5 m/s), (10 m/s), (15 m/s), and (20 m/s) speeds. As far as TCP Sintok is concerned, the average throughput decreased slightly as the speed increases. In specific, the average throughput is (646.5 Kbps) at (5 m/s) speed and declines to (588.6 Kbps) at (20 m/s). By contrast, ADTCP throughput suffers a significant drop at the same speed rates from approximately (122.7 Kbps) to (48.8 Kbps) at (20 m/s) speed. Interestingly, the graph showed that TCP Sintok achieves a considerable increase over ADTCP from (136.2%) at (5 m/s) speed to (169.3%) at (20 m/s) speed.

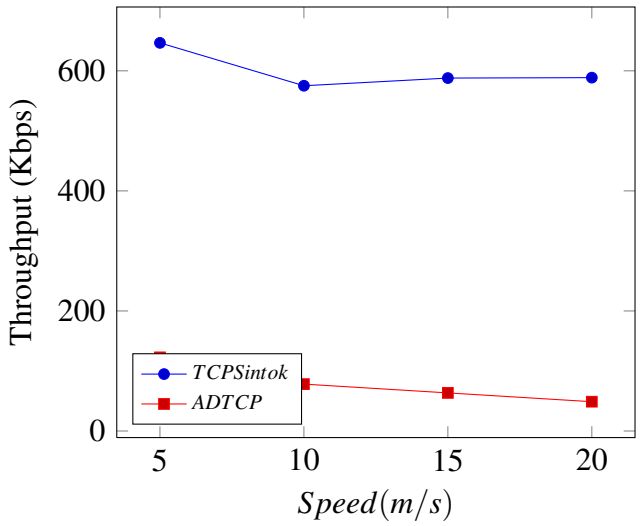


Figure 6.12: Throughput over Different Speeds

6.4.1.2 Mobility with 5% Channel Error Scenario

In this section, the previous scenario is repeated with 5% channel error instead of 0%. Figure 6.13 shows the differences in throughput over four different speeds. The average throughput of TCP Sintok has been relatively stable over all speeds. In contrast, the throughput of ADTCP has suffered a severe decline, particularly in (20 m/s) speed when throughput fell dramatically to (34.1 Kbps) as compared to TCP Sintok (363.4 Kbps). In (5 m/s) speed, TCP Sintok achieves (121.2%) better/higher average throughput than ADTCP, then the gap increases as the node movement speeds up and reaches to the highest at 20m/s with (165.7%).

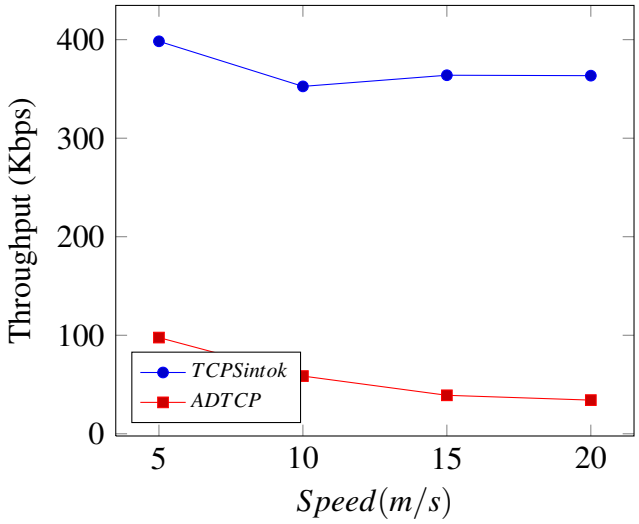


Figure 6.13: Throughput over Different Speeds and 5% Channel Error Rate

6.4.1.3 Mobility with 5% Channel Error and Three UDP Flows

In this scenario, three UDP flows are added to the last scenario to make it more realistic. Figure 6.14 shows the average throughput of TCP Sintok and ADTCP over four different speeds. Over all speeds, the average throughput of TCP Sintok has steadily decreased while ADTCP throughput dropped dramatically. In speed (5 m/s), TCP Sintok average throughput is (269.6 Kbps), around (128%) better than ADTCP throughput. As the node speed rises up, TCP Sintok continues to gain more throughput

than ADTCP at approximately (156.3%) at (10 m/s) and (159.2%) at (15 m/s), respectively. Interestingly, the performance improvement of TCP Sintok reaches (167.9%) more than ADTCP at (20 m/s) speed.

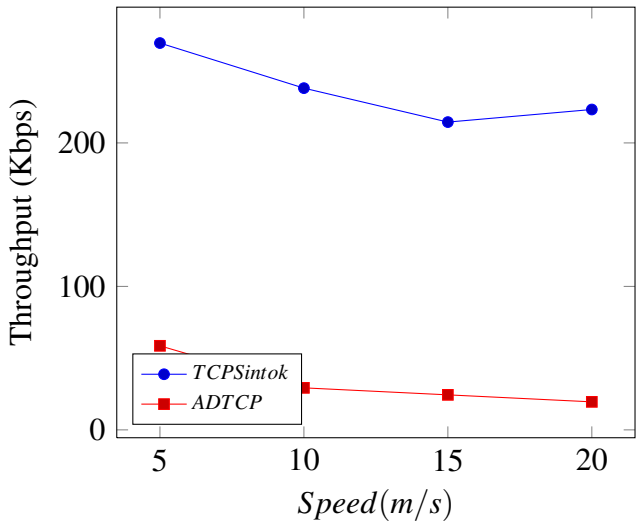


Figure 6.14: Throughput over Different Speeds, 5% Channel Error Rate, and Three UDP Flows

Over all, TCP Sintok and ADTCP are compared and results are presented in Figures 6.12 to 6.14 . The analysis showed that both TCP Sintok and TCP NewReno behave similarly in dealing with various speed and network conditions. However, TCP Sintok increases the average throughput experienced by FTP packets significantly between (121%) at speed (5 m/s) to (169%) at speed (20 m/s). Next section extends the analysis to compare TCP Sintok with ELFN.

6.4.2 TCP Sintok versus TCP ELFN

TCP ELFN collects route state information from the network layer directly. Therefore, it is expected to be more accurate. The previous scenario (i.e., TCP Sintok vs ADTCP) is replicated with the difference in the transport protocol deployed. The comparison between TCP Sintok and ELFN is conducted using the same network configuration

and setting presented in this work [1].

6.4.2.1 Mobility Scenario

This scenario analyzes the performance of TCP Sintok and ELFN in random topology with single TCP flow and zero error rate. Figure 6.15 compares the average throughput of TCP Sintok and ELFN over (5 m/s), (10 m/s), (15 m/s), and (20 m/s) speeds. As far as TCP Sintok is concerned, the average throughput decreases slightly as the speed increases. In specific, the average throughput is (646.5 Kbps) at (5 m/s) speed and declines to (588.6 Kbps) at (20 m/s). By contrast, ELFN throughput suffers a significant drop at the same speed rates from approximately (146.5 Kbps) to (58.6 Kbps) at (20 m/s) speed. Interestingly, The graph showed that TCP Sintok achieves a considerable increase over ELFN from (126.1%) at (5 m/s) speed to (163.7%) at (20 m/s) speed.

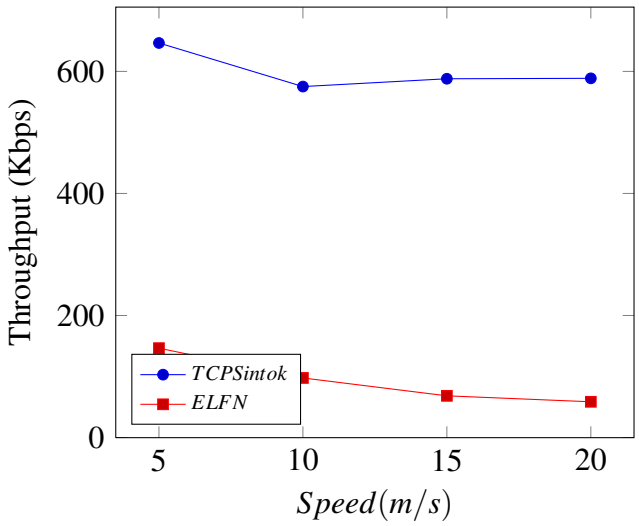


Figure 6.15: Throughput over Different Speeds

6.4.2.2 Mobility with 5% Channel Error Scenario

In this section, the previous scenario is repeated with (5%) channel error. Figure 6.16 shows the differences in throughput over four different speeds. The average throughput of TCP Sintok has been relatively stable over all speeds. In contrast, the throughput of

ELFN has suffered a severe decline, particularly in (20 m/s) speed when throughput fell dramatically to (39.1 Kbps) as compared to TCP Sintok (363.4 Kbps). In (5 m/s) speed, TCP Sintok achieves (109.1%) better/higher average throughput than ELFN, then the gap becomes wider as the node speed increases and reaches the highest at (20 m/s) with (161.1%).

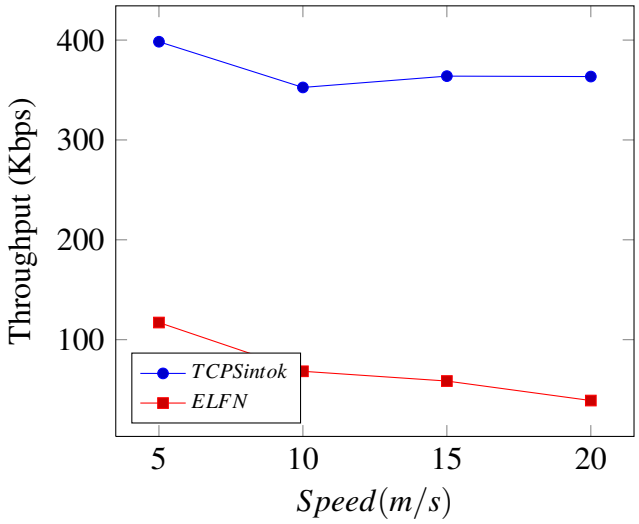


Figure 6.16: Throughput over Different Speeds and 5% Channel Error Rate

6.4.2.3 Mobility with 5% Channel Error and Three UDP Flows

In this scenario, three UDP flows are added to the last scenario to make it more realistic. Figure 6.17 shows the average throughput of TCP Sintok and ELFN over four different speeds. Over all speeds, the average throughput of TCP Sintok has steadily decreased while ELFN throughput dropped dramatically. In speed (5 m/s), TCP Sintok average throughput is (269.6 Kbps), almost (123%) better than ELFN throughput. As the node speed rises up, TCP Sintok continues to gain more throughput than ELFN with approximately (143.6%) at (10 m/s) and (156.2%) at (15 m/s), respectively. Interestingly, the performance improvement of TCP Sintok reaches (165%) more than ELFN at (20 m/s) speed.

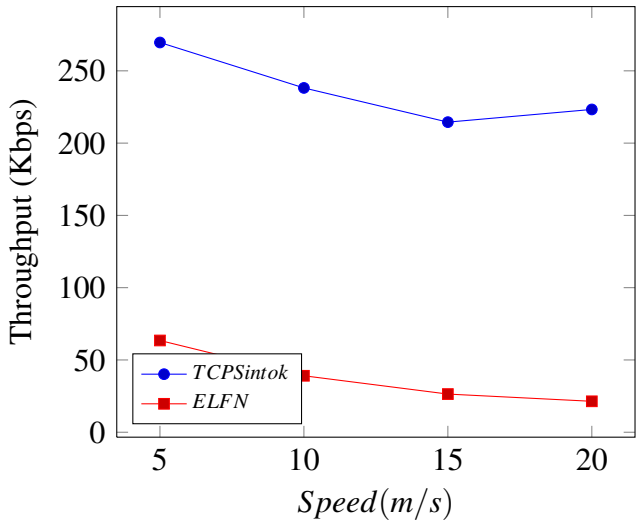


Figure 6.17: Throughput over Different Speeds, 5% Channel Error Rate, and Three UDP Flows

To conclude, TCP Sintok performance is compared to ELFN through this section and the results are presented in Figures 6.15 to 6.17. TCP Sintok achieves higher throughput than ELFN, which is between (109.1%) and (165%). Next section discusses the results that have been presented in Sections 6.3 and 6.4.

6.5 Discussion on TCP Sintok Performance

This section discusses and analyzes TCP Sintok performance in ad hoc networks based on the experimental results obtained in Section 6.3 and Section 6.4. It presents a summary of several tests conducted to look into the behavior of TCP Sintok under a variety of network conditions and scenarios. In specific, it illustrates the effects of each individual factor, namely error rate, number of hops, contention, routing protocol, speed, and network topology, as well as a combination of these factors on TCP Sintok performance, which are as follows:

Impact of different channel error rate: Channel errors are masked pretty well by the

back-off and retransmission mechanism of IEEE 802.11 MAC protocol. However, the presence of high channel error rate has negative impact on TCP performance. This is true for both TCP Sintok and TCP NewReno, as shown in Figures 6.4 and 6.5 where the performance drops as the error rate increases. However, TCP Sintok manages to gain higher throughput, which is in contrast to TCP NewReno in a very resource constrained topology (i.e., chain topology). The explanation is that since TCP Sintok reduces self-contention between its data and ACK segments that leads to reduce the impact of back-off mechanism of the underlying protocol. As a result, TCP Sintok throughput is improved.

Impact of different number of hops: Also, RTS/CTS plays an important role to avoid collision and hidden terminal, yet it becomes less effective as the number of hops between the source and the destination is increased. On the other hand, increasing the number of hops between the sender node and receiver node allows the sending more data simultaneously (i.e., one segment can run in four hops, and two in eight hops and so on). This will raise the probability of collision and contention resulting in degraded TCP performance. From the plot of Figure 6.7 and 6.8, it is clear that the performance of TCP Sintok and TCP NewReno drops as the number of hops increases. However, TCP Sintok achieves higher throughput and lower average delay due to its CAM mechanism that sets the congestion window size according to the contention level of the medium. More interestingly, the performance gap between TCP Sintok and TCP NewReno grows as the number of hops increases and TCP Sintok performs (33%) better throughput than TCP NewReno.

Impact of different routing protocols named AODV and DSR: The characteristics of different routing protocols have significant effects on TCP performance. Most studies reported in the literature considered one particular routing protocol during the eval-

uation phase. However, this does not give a clear view on the performance of the proposed transport protocol. In many simulation experiments carried out in this work, as shown in Figures 6.9, 6.10, 6.12, and 6.15, TCP Sintok performance has been tested using AODV and DSR over different scenarios and conditions. Even so, TCP Sintok performance is affected by the type of routing protocol, but it outperforms other protocols regardless of the routing protocols being used. This is due to the ability of TCP Sintok to detect mobility loss from congestion loss using its LDM without depending on the underlying protocols.

Impact of different speeds: When the mobility is high, the performance of TCP depends on the routing protocols. From the plot of Figures 6.12, 6.13, 6.15, and 6.16, it was observed that, as a general trend, the performance of TCP Sintok and those of ADTCP and ELFN performance would decrease as the mobility speed is increased. However, the drop in ADTCP and ELFN performance is much more dramatic when compared to that of TCP Sintok. This is because of TCP Sintok's fast response to topology changes as compared to the other which are wasting time in the backoff phase when a new effective path is discovered.

Impact of topologies: The ad hoc network could be represented in a variety of topologies depending on the implementation environments. Among the most popular static topologies are chain and grid. TCP Sintok was evaluated in 5-hop and 6-hop chain topology as well as a range from (5x5) to (9X9) grid topology. In static topology, TCP Sintok has achieved better performance as compared to TCP NewReno due to its CAM. Moreover, this study reports on random topology over a rectangle area of 400mX800m size and 30 mobile nodes moving using Random Way Point Model. Again, TCP Sintok outperforms TCP NewReno in random topology as shown in Figures 6.9, 6.10, and 6.11.

Impact of multiple factors: In order to emulate a real environment, different factors such as contention, mobility, and channel error, should appear simultaneously within the same scenario. Among these factors, contention has the most significant negative impact on TCP performance. It happens before congestion and indicates the first sign of network overload. In addition, increasing the speed from slow to fast affects TCP performance. As shown in Figures (6.12, 6.13, 6.14) and Figures (6.15, 6.16, 6.17), the improvement potential of ADTCP and ELFN has the tendency to decrease in the existence of competing flows as well as increasing node speeds. However, in contrast, the combination of LDM and CAM successfully increases TCP Sintok responsiveness to topology change and reduces the contention in the shared medium and increase special channel reuse, thus leading to a better overall performance by TCP Sintok, as compared to ADTCP and ELFN.

6.6 Summary

This chapter introduced a new transport protocol named TCP Sintok, specially for ad hoc networks. TCP Sintok was developed to extend the TCP NewReno functionalities. The design of TCP Sintok was discussed in this chapter. Then, the proposed protocol was implemented in the NS-2 simulation environment. Following this, the proposed protocol was evaluated inside the NS-2 simulator by observing different performance metrics over a variety of scenarios.

The evaluation was accomplished by examining TCP Sintok's average throughput, delay, and jitter as compared to that of TCP NewReno in different topologies. The evaluation criteria have also been fulfilled by comparing the average throughput of TCP Sintok over two recent proposals, named ADTCP and ELFN in random topology. Finally, the obtained results showed that TCP Sintok outperforms three other TCP proposals over ad hoc networks. Furthermore, TCP Sintok has been proven to be the

proper choice for developers and normal users in ad hoc networks.

CHAPTER SEVEN
CONCLUSION AND FUTURE WORKS

This thesis aimed at developing a new Transmission Control Protocol (TCP) called TCP Sintok for mobile ad hoc networks, and evaluated its performance extensively via simulation. This chapter provides the conclusion of the research work. It starts with Section 7.1, where the research findings and importance of the TCP Sintok are discussed with possible implementation and its benefits towards ad hoc networks. In Section 7.2, the contributions made by this research are highlighted. The limitations of the research is then presented in Section 7.3. Finally, Section 7.4 offers some suggestions for further studies.

7.1 Summary of the Research

The decentralized nature of ad hoc networks, in contrast to wired networks, comes with new features (pros) and challenges (cons) that violate the design concept of the Transmission Control Protocol (TCP), since TCP was well designed to work over wired networks where most packet losses occur due to network congestion at bottle-necked routers. However, in mobile ad hoc networks, packet losses mostly occur due to node mobility or link layer contention leading to the degradation of overall network TCP performance. Yet, end users expect TCP to continue to provide fast and reliable data delivery service without posing any drawback or side effects in ad hoc networks.

As was mentioned in Chapter 1, the research work presented in this thesis was motivated by the need for a new TCP to improve the performance of reliable data delivery over IEEE 802.11 DCF ad hoc networks. The main aim of this thesis was to introduce a new TCP, named TCP Sintok, which responds to mobility induced packet loss accurately, controls the growth of congestion window correctly according to the current

network conditions, and reacts to different types of packet loss appropriately.

It was observed and concluded that the current TCPs cannot handle the challenges caused by the mobility features and contention on wireless channels in mobile ad hoc networks, which have a negative impact on TCP performance. For instance, TCP NewReno, which is the default TCP recommended by IETF for the Internet, suffers from low throughput, high delay, high delay variance (jitter), and high packet loss, as explained in detail in Chapter 2. This has encouraged and prompted this research effort to introduce TCP Sintok for mobile ad hoc networks.

Design Research Methodology (DRM) was adopted as a framework and guideline to accomplish this research. In addition, network modeling and simulation process proposed by Guizani et al. [118] as well as Jain [9] were adopted to achieve the research objectives.

Firstly, delay-based Loss Detection Mechanism (LDM) over mobile ad hoc networks was introduced in Chapter Four. LDM determines the cause of packet loss by monitoring the trend of delay samples and reacts according to the packet loss type. Theoretical and simulation results showed that TCP with LDM improves loss detection accuracy.

Next, a novel Contention Avoidance Mechanism (CAM) was developed in Chapter Five. The proposed mechanism was inspired by Communication Accommodation Theory (CAT) in adapting the sending rate (congestion window). A new efficient link utilization method was proposed for use in adjusting the sending rate. Simulation experiments had confirmed that the proposed mechanism controls the growth of congestion window within the optimal window size. This proposed mechanism was validated using simulation experiments.

A series of experimental studies were conducted to validate the effectiveness of TCP Sintok in identifying the cause of packet loss and adapting the sending rate appropriately. Two variants of TCP protocol, known as ADTCP and ELFN, in addition to the traditional TCP were chosen to evaluate the performance of TCP Sintok through simulation. The results demonstrated that TCP Sintok improves jitter, delay, and throughput as compared to these current variants. The findings have significant implication in providing reliable data transfer within MANET and thus encourage the development of new ad hoc network applications. Thus, it aids in supporting the deployment of ad hoc networks on mobile device communications.

In completing the TCP Sintok performance evaluation, the results confidently emphasizes the fact that the framed objectives of this research, as presented in this thesis, have been well and wholly achieved.

7.2 Research Contributions

The overall contribution of this research was to develop a new TCP to improve the performance of ad hoc networks. A mathematical model for end-to-end delay was developed to investigate the impact of congestion on delay trends. Furthermore, a loss detection mechanism and contention avoidance mechanism were proposed to induce enhancements within TCP Sintok. The specific contributions of this thesis are as follows:

i. The development of a performance model for TCP over ad hoc networks can serve as a benchmark for any intended improvement and enhancement of TCP in this dynamic environment.
ii. The improvement in the accuracy of packet loss detection in TCP congestion control was achieved by introducing a new Loss Detection Mechanism (LDM).

a. The development of a mathematical model of end-to-end delay was performed to study the impact of congestion and contention network conditions on delay trend.
b. The validation of the proposed delay model was achieved by comparing the model results with results obtained from a valid network simulator.
c. The design of delay-based loss detection mechanism (LDM) was formulated which is capable of efficiently distinguishing mobility loss from congestion loss within ad hoc networks.
d. The verification and validation of the proposed LDM was performed by implementing it in NS-2 simulation environment and comparing the identified network state by LDM and the actual network state in trace file.

iii. The adaptation of the sending rate of TCP congestion control based on the current network condition was made by designing a new contention avoidance mechanism to help TCP control the growth of congestion window to avoid contention and enhance network resource utilization.

a. Exploring the applicability of Communication Accommodation Theory (CAT) was performed in computer communication.
b. The design of Contention Avoidance Mechanism (CAM) based on CAT features and properties was carried out, which is able to accurately adapt congestion window size according to the network condition leading to optimal resource utilization.
c. The verification and validation of the proposed CAM was achieved by implementing it in NS-2 and comparing the obtained results with other results produced by real test bed experiments.

iv. The development TCP Sintok for ad hoc networks was based on the standard TCP NewReno.

a. Incorporation of the proposed LDM and CAM in TCP sender was achieved.
b. The modification of TCP Congestion control actions according to network status was performed.
c. The evaluation of TCP Sintok over a variety of scenarios was illustrated by implementing it in NS-2 and showing significant improvement over the standard TCP NewReno (legacy TCP).
d. The comparison of TCP Sintok to recent related works (namely, ELFN and ADTCP) showing significant performance gained was performed.

7.3 Research Limitation

Although this study was conducted under careful selection and application of a set of supporting methods and guidelines, it is limited to specific and precise utilization. First of all, the proposed network topologies used in the implementation are widely used and proven in ad hoc networks. The research was conducted in specific network conditions and environments, and not including all the topologies and structures. Moreover, the number of nodes used in the validation and performance evaluation was limited and fixed, whereas in real ad hoc networks it is unpredictable and changeable. Furthermore, not all types of IEEE 802.11 standards were used between the nodes, the focus was only on IEEE 802.11 DCF b. Moreover, the random way point mobility model was not discussed since it was presumed to not affect the result at all. However, network behavior is always unpredictable. Also, only FTP was used during the evaluation phase, and not all applications were discussed due to the time limitation and experiment sophistication.

7.4 Future Works

The proposed mechanisms in this research improve the performance of TCP in a variety of scenarios. However, there are some limitations as well as pending works that can be pursued for future directions. This section outlines some possible extensions, which are as follows:

Extending end-to-end delay model:

The proposed delay model in Chapter 4 was designed with focus on contention, congestion, and random channel error (Uniform Error Model). More accuracy appears to be possible if another network state is adopted in the model. In addition, different error distribution could be evaluated to determine to what extent they may influence the model accuracy.

Automatic setting of congestion window size and RTO:

TCP Sintok was evaluated with fixed actions adopted at the sender to set *cwnd* and *RTO* values upon packet loss detection. However, to accelerate the responsiveness, *cwnd* and *RTO* values could be adaptive dynamically according to network conditions.

A customized increase factor:

The proposed CAM is currently focused on the sender side and maintained end-to-end semantics. It is well-known that cross layers can give more accurate information, but it would increase the complexity and deployability of the proposed mechanism. Therefore, it is worth to conduct a comprehensive investigation on the cross layer approach to deal with the effect of dynamic link layer contention.

Extending interoperability of TCP Sintok across and beyond the Internet:

The proposed TCP Sintok is completely customized to wireless ad hoc networks. Users may find it useful to be connected with the Internet. In such scenarios, the TCP in ad hoc network is required to be able to establish a connection with the Internet. Furthermore, it is necessary to have some sort of gateway to establish this interoperation. To investigate the presented changes and functionalities required in TCP Sintok is indeed a viable extension of this research.

Evaluation of TCP Sintok in a testbed:

Although TCP Sintok was evaluated comprehensively and extensively through a validated simulator, its implementation in a real testbed is definitely of interest. Furthermore, to evaluate TCP Sintok using real traffic is surely a good extension to be performed in extending the life of this research.

REFERENCES

[1] Z. Fu, B. Greenstein, X. Meng, and S. Lu, "Design and Implementation of a TCP-friendly Transport Protocol for Ad Hoc Wireless Networks," in *Network Protocols, 2002. Proceedings. 10th IEEE International Conference on*. IEEE, 2002, pp. 216–225.

[2] R. Jain, *Art of Computer Systems Performance Analysis: Techniques for Experimental Design Measurements Simulation and Modeling*. John Wiley & Sons, Inc., 1991.

[3] M. Małowidzki, "Network Simulators: A Developer's Perspective." Citeseer, 2004, pp. 1–9.

[4] A. Al Hanbali, E. Altman, and P. Nain, "A Survey of TCP over Ad Hoc Networks," *IEEE Communications Surveys & Tutorials*, vol. 7, no. 3, pp. 22–36, 2005.

[5] J. Kurose and K. Ross, *Computer Networks: A Top Down Approach Featuring the Internet*. Pearson Addison Wesley, 2012.

[6] M. Z. Oo and M. Othman, "The Effect of Packet Losses and Delay on TCP Traffic over Wireless Ad Hoc Networks," in *Mobile Ad-Hoc Networks: Applications*, X. Wang, Ed. InTech, 2011, pp. 425–450.

[7] F. Wang and Y. Zhang, "Improving TCP Performance over Mobile Ad-hoc Networks with Out-of-order Detection and Response," in *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*. ACM, 2002, pp. 217–225.

[8] A. Ghaleb-Seddik, Y. Ghamri-Doudane, and S. M. Senouci, "Coupling Loss and Delay Differentiation to Eenhance TCP Performance within Wireless Multi-hop Ad-hoc Networks," *Journal of Communications*, vol. 7, no. 12, pp. 859–872, 2012.

[9] M. Hassan and R. Jain, *High Performance TCP/IP Networking*. Pearson Prentice Hall, 2004.

[10] I. . L. S. Committee *et al.*, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, Std., march 2012.

[11] A. Balador, A. Movaghar, S. Jabbehdari, D. Kanellopoulos *et al.*, "A Novel Contention Window Control Scheme for IEEE 802.11 WLANs," *IETE Technical Review*, vol. 29, no. 3, p. 202, 2012.

[12] M. S. Corson and J. Macker, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations," Internet Engineering Task Force, RFC 2501, Sep. 1999. [Online]. Available: http://datatracker.ietf.org/doc/rfc2501/

[13] K. Leung and V. Li, "Transmission Control Protocol (TCP) in Wireless Networks: Issues, Approaches, and Challenges," *IEEE Communications Surveys & Tutorials*, vol. 8, no. 4, pp. 64–79, 2006.

[14] B. Crow, I. Widjaja, J. G. Kim, and P. Sakai, "IEEE 802.11 Wireless Local Area Networks," *IEEE Communications Magazine*, vol. 35, no. 9, pp. 116–126, 1997.

[15] S. Bluetooth, *Specification of the Bluetooth System, version 1.1*, Std., 2001.

[16] B. A. Miller, C. Bisdikian, and T. Foreword By-Siep, *Bluetooth Revealed*. Prentice Hall PTR, 2001.

[17] P. Mohapatra and S. Krishnamurthy, *Ad Hoc Networks: Technologies and Protocols*. Springer, 2005.

[18] J. Postel, "Transmission Control Protocol," Internet Engineering Task Force, RFC 0793, Sep. 1981. [Online]. Available: http://www.rfc-editor.org/rfc/rfc793.txt

[19] M. Allman and A. Falk, "On the Effective Evaluation of TCP," *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 5, pp. 59–70, 1999.

[20] M. Morshed, M. Rahman, M. Rahman, and M. Islaml, "Performance Comparison of TCP Variants over AODV, DSDV, DSR, OLSR in NS-2," in *Informatics, Electronics Vision (ICIEV), 2012 International Conference on*, 2012, pp. 1069–1074.

[21] A. Boukerche, *Algorithms and Protocols for Wireless and Mbile Ad Hoc Networks*. Wiley-IEEE Press, 2009, vol. 77.

[22] M.-Y. Park, S.-H. Chung, and C.-W. Ahn, "TCPs Dynamic Adjustment of Transmission Rate to Packet Losses in Wireless Networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2012, no. 1, p. 304, 2012.

[23] E. Larsen, "TCP in MANETs–Challenges and Solutions," Norwegian Defence Research Establishment (FFI), Tech. Rep., Sep. 2012. [Online]. Available: http://www.rfc-editor.org/rfc/rfc793.txt

[24] B. Soelistijanto and M. Howarth, "Transfer Reliability and Congestion Control Strategies in Opportunistic Networks: A Survey," *Communications Surveys Tutorials, IEEE*, vol. PP, no. 99, pp. 1–18, 2013.

[25] F. Lee, "Routing in Mobile Ad Hoc Networks," in *Mobile Ad-Hoc Networks: Protocol Design*, X. Wang, Ed. InTech, 2011, pp. 299–322.

[26] G. Holland and N. Vaidya, "Analysis of TCP Performance over Mobile Ad Hoc Networks," *Wireless Networks*, vol. 8, no. 2/3, pp. 275–288, 2002.

[27] S. M. Mirhosseini and F. Torgheh, "ADHOCTCP: Improving TCP Performance in Ad Hoc Networks," in *Mobile Ad-Hoc Networks: Protocol Design*, X. Wang, Ed. InTech, 2011, pp. 121–138.

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

[29] M. Conti and S. Giordano, "Multihop ad hoc networking: The Theory," *Communications Magazine, IEEE*, vol. 45, no. 4, pp. 78–86, 2007.

[30] 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 *INFOCOM 2003. Twenty-second annual joint conference of the IEEE Computer and Communications. IEEE Societies*, vol. 3. IEEE, 2003, pp. 1744–1753.

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

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

[33] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno and SACK TCP," *ACM SIGCOMM Computer Communication Review*, vol. 26, no. 3, pp. 5–21, 1996.

[34] S. Floyd, T. Henderson, and A. Gurtov, "The NewReno Modification to TCPs Fast Recovery Algorithm," Internet Engineering Task Force, RFC 2582, 1999. [Online]. Available: https://datatracker.ietf.org/doc/rfc2582/

[35] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," Internet Engineering Task Force, RFC 2001, January 1997.

[36] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgment Options," Internet Engineering Task Force, RFC 2018, Oct. 1996. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2018.txt

[37] S. Floyd, T. Henderson *et al.*, "RFC 3782: The NewReno Modification to TCP's Fast Recovery Algorithm," Internet Engineering Task Force, RFC 3782, 2004. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3782.txt

[38] M. Allman, V. Paxson, and E. Blanton, "TCP Congestion Control," Internet Engineering Task Force, RFC 5681, 2009. [Online]. Available: https://datatracker.ietf.org/doc/rfc5681/

[39] A. Al-Jubari, M. Othman, B. Mohd Ali, and N. Abdul Hamid, "An Adaptive Delayed Acknowledgment Strategy to Improve TCP Performance in Multi-hop Wireless Networks," *Wireless Personal Communications*, vol. 69, no. 1, pp. 307–333, 2013. [Online]. Available: http://dx.doi.org/10.1007/s11277-012-0575-9

[40] N. Mast and T. J. Owens, "A Survey of Performance Enhancement of Transmission Control Protocol (TCP) in Wireless Ad Hoc Networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2011, no. 1, pp. 1–23, 2011.

[41] U. Ibom, "TCP Performance over MANET," in *Information Networking, 2008. ICOIN 2008. International Conference on*. IEEE, 2008, pp. 1–5.

[42] J. Choi, S. Yoo, and C. Yoo, "An Enhancement Scheme for TCP over Mobile Ad Hoc Networks," in *Vehicular Technology Conference, 2003. VTC 2003-Spring. The 57th IEEE Semiannual*, vol. 3. IEEE, 2003, pp. 1522–1526.

[43] M. Gerla, R. Bagrodia, L. Zhang, K. Tang, and L. Wang, "TCP over Wireless Multi-hop Protocols: Simulation and Experiments," in *Communications, 1999. ICC'99. 1999 IEEE International Conference on*, vol. 2. IEEE, 1999, pp. 1089–1094.

[44] H. Wu, Y. Peng, K. Long, S. Cheng, and J. Ma, "Performance of Reliable Transport Protocol over IEEE 802.11 Wireless LAN: Analysis and Enhancement," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2. IEEE, 2002, pp. 599–607.

[45] B. Sreenivas, G. Bhanu Prakash, and K. Ramakrishnan, "L2DB-TCP: An Adaptive Congestion Control Technique for MANET Based on Link Layer Measurements," in *Advance Computing Conference (IACC), 2013 IEEE 3rd International*. IEEE, 2013, pp. 1086–1093.

[46] A. Gupta, I. Wormsbecker, and C. Wilhainson, "Experimental Evaluation of TCP Performance in Multi-hop Wireless Ad Hoc Networks," in *Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, 2004.(MASCOTS 2004). Proceedings. The IEEE Computer Society's 12th Annual International Symposium on*. IEEE, 2004, pp. 3–11.

[47] M. Allman, V. Paxson, W. Stevens *et al.*, "TCP congestion control," Internet Engineering Task Force, RFC 2581, 1999. [Online]. Available: http://datatracker.ietf.org/doc/rfc2581/

[48] Z. Fu, X. Meng, and S. Lu, "How Bad TCP Can Perform in Mobile Ad Hoc Networks," in *Computers and communications, 2002. Proceedings. ISCC 2002. Seventh international symposium on*. IEEE, 2002, pp. 298–303.

[49] D. Kouvatsos, *Network Performance Engineering: A Handbook on Convergent Multi-service Networks and Next Generation Internet*. Springer, 2011, vol. 5233.

[50] H. b. Liu and Y. Gu, "Survey on TCP Congestion Control for MANET," *Journal of Central South University (Science and Technology)*, vol. 44, no. 1, pp. 156–165, 2013.

[51] C. Sharma and B. Tyagi, "Performance Evaluation of TCP Variants Under Different Node Speeds Using OPNET Simulator," in *Advance Computing Conference (IACC), 2013 IEEE 3rd International*. IEEE, 2013, pp. 302–307.

[52] C. Lochert, B. Scheuermann, and M. Mauve, "A Survey on Congestion Control for Mobile Ad Hoc Networks," *Wireless Communications and Mobile Computing*, vol. 7, no. 5, pp. 655–676, 2007.

[53] M. Welzl, *Network Congestion Control: Managing Internet Traffic*. Wiley Online Library, 2005.

[54] M. Duke, R. Braden, W. Eddy, and E. Blanton, "A Roadmap for Transmission Control Protocol (TCP) Specification Documents," Internet Engineering Task Force, RFC 4614, 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4614.txt

[55] D. Papadimitriou, M. Welzl, M. Scharf, and B. Briscoe, "Open Research Issues in Internet Congestion Control," Internet Engineering Task Force, RFC 6077, Feb. 2011. [Online]. Available: https://datatracker.ietf.org/doc/rfc6077/

[56] V. Jacobson, "Congestion Avoidance and Control," in *ACM SIGCOMM Computer Communication Review*, vol. 18, no. 4. ACM, 1988, pp. 314–329.

[57] V. Paxson and M. Allman, "Computing TCP's Retransmission Timer," Internet Engineering Task Force, RFC 2988, Nov. 2000. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2988.txt

[58] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the Internet," *IEEE/ACM Transactions on Networking (TON)*, vol. 7, no. 4, pp. 458–472, 1999.

[59] H. Jiang, S. Cheng, and X. Chen, "TCP Reno and Vegas Performance in Wireless Ad Hoc Networks," in *Communications, 2001. ICC 2001. IEEE International Conference on*, vol. 1. IEEE, 2001, pp. 132–136.

[60] S. Xu, T. Saadawi, and M. Lee, "Comparison of TCP Reno and Vegas in Wireless Mobile Ad Hoc Networks," in *Local Computer Networks, 2000. LCN 2000. Proceedings. 25th Annual IEEE Conference on*. IEEE, 2000, pp. 42–43.

[61] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Reno Performance: A Simple Model and its Empirical Validation," *Networking, IEEE/ACM Transactions on*, vol. 8, no. 2, pp. 133 –145, Apr. 2000.

[62] W. Xu and T. Wu, "TCP Issues in Mobile Ad Hoc Networks: Challenges and Solutions," *Journal of Computer Science and Technology*, vol. 21, no. 1, pp. 72–81, 2006.

[63] A. M. Al-Jubari, M. Othman, B. M. Ali, and N. A. W. A. Hamid, "TCP Performance in Multi-hop Wireless Ad Hoc Networks: Challenges and Solution," *EURASIP Journal on Wireless Communications and Networking*, vol. 2011, no. 1, pp. 1–25, 2011.

[64] A. Gurtov and S. Floyd, "Modeling Wireless Links for Transport Protocols," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 85–96, 2004.

[65] G. Holland and N. Vaidya, "Impact of Routing and Link Layers on TCP Performance in Mobile Ad Hoc Networks," in *Wireless Communications and Networking Conference, 1999. WCNC. 1999 IEEE*. IEEE, 1999, pp. 1323–1327.

[66] A. Ahmed, S. Zaidi, and N. Ahmed, "Performance Evaluation of Transmission Control Protocol in Mobile aA Hoc Networks," in *Networking and Communication Conference, 2004. INCC 2004. International*. IEEE, 2004, pp. 13–18.

[67] S. Fong, "Loss Discrimination Algorithm for Wired/Wireless Networks," *Journal of Computer Science*, vol. 7, 2011.

[68] A. Habbal and S. Hassan, "Loss Detection and Recovery Techniques for TCP in Mobile Ad Hoc Network," in *Network Applications Protocols and Services (NETAPPS), 2010 Second International Conference on*. IEEE, 2010, pp. 48–54.

[69] D. Triantafyllidou, K. Al Agha, and V. Siris, "Adaptive setting of TCP's maximum window in ad hoc multihop networks with a single flow," in *Wireless Communications and Networking Conference, 2009. WCNC 2009. IEEE*. IEEE, 2009, pp. 1–6.

[70] K. Kim, P. Lorenz, and M. Lee, "A New Tuning Maximum Congestion Window for Improving TCP Performance in MANET," in *Systems Communications, 2005. Proceedings*. IEEE, 2005, pp. 73–78.

[71] K. Chen, Y. Xue, and K. Nahrstedt, "On Setting TCP's Congestion Window Limit in Mobile Ad Hoc Networks," in *Communications, 2003. ICC'03. IEEE International Conference on*, vol. 2. IEEE, 2003, pp. 1080–1084.

[72] R. de Oliveira and T. Braun, "A Smart TCP Acknowledgment Approach for Multihop Wireless Networks," *Mobile Computing, IEEE Transactions on*, vol. 6, no. 2, pp. 192–205, 2007.

[73] J. Chen, M. Gerla, Y. Lee, and M. Sanadidi, "TCP with Delayed ACK for Wireless Networks," *Ad Hoc Networks*, vol. 6, no. 7, pp. 1098–1116, 2008.

[74] M. Feeley, B. Cully, and S. George, "Understanding Performance for Two 802.11 Competing Flows," *Journal of Computer Science & Technology*, vol. 3, p. 006, 2008.

[75] D. Kim, J. Cano, P. Manzoni, and C. Toh, "A Comparison of the Performance of TCP-Reno and TCP-Vegas over MANETs," in *Wireless Communication Systems, 2006. ISWCS'06. 3rd International Symposium on*. IEEE, 2006, pp. 495–499.

[76] X. Zhang, N. Li, W. Zhu, and D. Sung, "TCP Transmission Rate Control Mechanism Based on Channel Utilization and Contention Ratio in Ad Hoc Networks," *Communications Letters, IEEE*, vol. 13, no. 4, pp. 280–282, 2009.

[77] X. M. Zhang, W. B. Zhu, N. N. Li, and D. K. Sung, "TCP Congestion Window Adaptation Through Contention Detection in Ad Hoc Networks," *Vehicular Technology, IEEE Transactions on*, vol. 59, no. 9, pp. 4578–4588, 2010.

[78] A. Singh and K. Kankipati, "TCP-ADA: TCP with Adaptive Delayed Acknowledgement for Mobile Ad Hoc Networks," in *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE*, vol. 3. IEEE, 2004, pp. 1685–1690.

[79] S. Fu and M. Atiquzzaman, "DualRTT: Detecting Spurious Timeouts in Wireless Mobile Environments," in *Performance, Computing, and Communications Conference, 2005. IPCCC 2005. 24th IEEE International*. IEEE, 2005, pp. 129–133.

[80] D. Kim, C. Toh, and H. Yoo, "The Impact of Spurious Retransmissions on TCP Performance in Ad Hoc Mobile Wireless Networks," in *Personal, Indoor and Mobile Radio Communications, 2007. PIMRC 2007. IEEE 18th International Symposium on*. IEEE, 2007, pp. 1–5.

[81] A. Ahuja, S. Agarwal, J. Singh, and R. Shorey, "Performance of TCP over Different Routing Protocols in Mobile Ad-hoc Networks," in *Vehicular Technology Conference Proceedings, 2000. VTC 2000-Spring Tokyo. 2000 IEEE 51st*, vol. 3. IEEE, 2000, pp. 2315–2319.

[82] M. Rahman and H. Tan, "Performance Evaluation of TCP over Routing Protocols for Mobile Ad Hoc Networks," in *Communications and Networking in China, 2006. ChinaCom'06. First International Conference on*. IEEE, 2006, pp. 1–3.

[83] N. Premalatha and A. Natarajan, "Congestion Control in Wireless Ad Hoc Networks by Enhancement of Transmission Control Protocol," *Journal of Computer Science*, vol. 7, no. 12, p. 1824, 2011.

[84] C. P. Sahu, P. S. Yadav, S. Ahuja, R. Prasad, and A. K. Garg, "Optimistic Congestion Control to Improve the Performance of Mobile Ad Hoc Network," in *Advance Computing Conference (IACC), 2013 IEEE 3rd International*. IEEE, 2013, pp. 394–398.

[85] H. Touati, I. Lengliz, and F. Kamoun, "Adapting TCP Exponential Backoff to Multihop Ad Hoc Networks," in *Computers and Communications, 2009. ISCC 2009. IEEE Symposium on*. IEEE, 2009, pp. 612–617.

[86] T. D. Dyer and R. V. Boppana, "A Comparison of TCP Performance over three Routing Protocols for Mobile Ad hoc Networks," in *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*. ACM, 2001, pp. 56–66.

[87] M. Kang, H. Park, and J. Mo, "Implementation and Evaluation of a New TCP Loss Recovery Architecture," *EURASIP Journal on Wireless Communications and Networking*, vol. 2012, no. 1, pp. 1–9, 2012.

[88] H. Touati, I. Lengliz, and F. Kamoun, "Performance of TCP Adaptive RTO in Ad-hoc Networks Based on Different Routing Protocols," in *Mobile Wireless Communications Networks, 2007 9th IFIP International Conference on*. IEEE, 2007, pp. 176–180.

[89] Q. Lin, K. Chan, K. Tan, and B. Yeo, "Partition-Aware TCP for Mobile Ad-Hoc Networks," in *Communications, 2006. ICC'06. IEEE International Conference on*, vol. 8. IEEE, 2006, pp. 3777–3782.

[90] W. Sun, T. Wen, and Q. Guo, "A Novel Protocol for Mobile-Induced Packet Reordering in Mobile Ad Hoc NetWorks," in *Information Science and Engineering, 2008. ISISE'08. International Symposium on*, vol. 1. IEEE, 2008, pp. 626–631.

[91] S. Yang and Y. Lin, "Tuning Rules in TCP Congestion Control on the Mobile Ad Hoc Networks," in *Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on*, vol. 1. IEEE, 2006, pp. 759–766.

[92] C. Xiong, J. Yim, J. Leigh, and T. Murata, "Energy-Efficient Method to Improve TCP Performance for MANETs," in *Proceedings of 2004 International Conference on Computing, Communications and Control Technologies (CCCT 04)*, 2004, pp. 327–331.

[93] S. Bhandarkar, N. Sadry, A. Reddy, and N. Vaidya, "TCP-DCR: A Novel Protocol for Tolerating Wireless Channel Errors," *Mobile Computing, IEEE Transactions on*, vol. 4, no. 5, pp. 517–529, 2005.

[94] M. Li, B. Song, and J. Liu, "An End-to-end TCP Enhanced Scheme for Ad Hoc Wireless Networks," in *Wireless, Mobile and Multimedia Networks, 2006 IET International Conference on*. IET, 2006, pp. 1–4.

[95] S. Gajjar and H. Gupta, "Improving Performance of Adhoc TCP in Mobile Adhoc Networks," in *India Conference, 2008. INDICON 2008. Annual IEEE*, vol. 1. IEEE, 2008, pp. 144–147.

[96] T. Yanping, W. Haizhen, J. Mei, and L. Dahui, "Improvement Scheme of End-to-end TCP Congestion Control in Ad Hoc Network," in *Computer Science and Network Technology (ICCSNT), 2012 2nd IEEE International Conference on*, 2012, pp. 1068–1071.

[97] C. Kai, Y. Chen, and N. Yu, "An Improvement Scheme Applied to TCP Protocol in Mobile Ad Hoc Networks," in *Mobile Technology, Applications and Systems, 2005 2nd International Conference on*. IEEE, 2005, pp. 1–6.

[98] S. Biaz and N. Vaidya, "Distinguishing Congestion Losses from Wireless Transmission Losses: A Negative Result," in *Computer Communications and Networks, 1998. Proceedings. 7th International Conference on*. IEEE, 1998, pp. 722–731.

[99] R. Braden, "Requirements for Internet Hosts - Communication Layers," Internet Engineering Task Force, RFC 1122, Oct. 1989. [Online]. Available: http://www.rfc-editor.org/rfc/rfc1122.txt

[100] S. M. ElRakabawy and C. Lindemann, "A Practical Adaptive Pacing Scheme for TCP in Multihop Wireless Networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 19, no. 4, pp. 975–988, 2011.

[101] K. Nahm, A. Helmy, and C. Kuo, "Cross-layer Interaction of TCP and Ad Hoc Routing Protocols in Multihop IEEE 802.11 Networks," *Mobile Computing, IEEE Transactions on*, vol. 7, no. 4, pp. 458–469, 2008.

[102] X. Wang, Y. Han, and Y. Xu, "APS-FeW: Improving TCP Throughput over Multihop Adhoc Networks," *Computer Communications*, vol. 32, no. 1, pp. 19–24, 2009.

[103] E. Altman and T. Jiménez, "Novel Delayed ACK Techniques for Improving TCP Performance in Multihop Wireless Networks," in *Personal Wireless Communications*. Springer, 2003, pp. 237–250.

[104] R. De Oliveira and T. Braun, "A Delay-based Approach Using Fuzzy Logic to Improve TCP Error Detection in Ad hoc Networks," in *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE*, vol. 3. IEEE, 2004, pp. 1666–1671.

[105] D. Los Angeles Thomas *et al.*, *Elementary signal detection theory*. Oxford University Press, 2001.

[106] N. A. Macmillan and C. D. Creelman, *Detection theory: A user's guide*. Psychology press, 2004.

[107] N. C. Howard Giles, Justine Coupland, "Accommodation Theory: Communication, Context, and Consequence," *Contexts of accommodation: Developments in applied sociolinguistics*, p. 1, 1991.

[108] H. Giles, J. Coupland, and N. Coupland, *Contexts of Accommodation: Developments in AppliedSociolinguistics*. Cambridge University Press, 1991.

[109] S. W. Littlejohn and K. A. Foss, *Theories of Human Communication*. Wadsworth Publishing Company, 2007.

[110] J. A. DeVito, *Essentials of Human Communication*. Longman, 2002.

[111] L. Christopherson, "Can u Help me Plz? Cyberlanguage Accommodation in Virtual Reference Conversations," *Proceedings of the American Society for Information Science and Technology*, vol. 48, no. 1, pp. 1–9, 2011.

[112] H. J. Ladegaard, "Pragmatic Cooperation Revisited: Resistance and Non-cooperation as a Discursive Strategy in Asymmetrical Discourses," *Journal of Pragmatics*, vol. 41, no. 4, pp. 649–666, 2009.

[113] P. Offermann, O. Levina, M. Schönherr, and U. Bub, "Outline of a Design Science Research Process," in *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology*. ACM, 2009, p. 7.

[114] L. Blessing and A. Chakrabarti, *DRM: A Design Research Methodology*. Springer Verlag, 2009.

[115] A. M. M. Habbal and S. Hassan, "A Model for Congestion Control of Transmission Control Protocol in Mobile Wireless Ad hoc Networks," *Journal of Computer Science*, vol. 9, no. 3, pp. 335–342, 2013.

[116] K. Zaini, A. Habbal, F. Azzali, S. Hassan, and M. Rizal, "An Interaction Between Congestion-Control Based Transport Protocols and MANET Routing Protocols," *Journal of Computer Science*, vol. 8, 2012.

[117] N. Haniza, M. Khambari, S. Shahrin, A. Habbal, and S. Hassan, "Topology Influence on TCP Congestion Control Performance in Multi-hop Ad Hoc Wireless," in *Proceedings of World Academy of Science, Engineering and Technology*, no. 61. World Academy of Science, Engineering and Technology, 2012.

[118] M. Guizani, A. Rayes, B. Khan, and A. Al-Fuqaha, *Network Modeling and Simulation: A Practical Perspective*. Wiley-Interscience, 2010.

[119] O. Balci, "Verification Validation and Accreditation of Simulation Models," in *Proceedings of the 29th conference on Winter simulation*. IEEE Computer Society, 1997, pp. 135–141.

[120] S. Kurkowski, T. Camp, and M. Colagrosso, "MANET Simulation Studies: The Incredibles," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 9, no. 4, pp. 50–61, 2005.

[121] O. Balci, "Validation, Verification, and Testing Techniques Throughout the Life Cycle of a Simulation Study," *Annals of operations research*, vol. 53, no. 1, pp. 121–173, 1994.

[122] S. Schlesinger, R. E. Crosbie, R. E. Gagne, G. S. Innis, C. Lalwani, J. Loch, R. J. Sylvester, R. D. Wright, N. Kheir, and D. Bartos, "Terminology for Model Credibility," *Simulation*, vol. 32, no. 3, pp. 103–104, 1979.

[123] R. G. Sargent, "Verification and validation of Simulation Models," in *Proceedings of the 37th conference on Winter simulation*. Winter Simulation Conference, 2005, pp. 130–143.

[124] S. Hassan, "Simulation-based Performance Evaluation of TCP-Friendly Protocols for Supporting Multimedia Application in the Internet," PhD Thesis, Computer Science Department, University of Leeds, 2002.

[125] J. Mo, *Performance Modeling of Communication Networks with Markov Chains*. Morgan & Claypool Publishers, 2010.

[126] J.-Y. Le Boudec, *Performance Evaluation of Computer and Communication Systems*. EPFL Press, Lausanne, Switzerland, 2010.

[127] O. B. Lynn, "A Hybrid Mechanism for SIP over IPv6 Macromobility and Micromobility Management Protocols," PhD Thesis, College of Arts and Sciences, Universiti Utara Malaysia, 2008.

[128] O. Ghazali, "Scaleable and Smooth TCP-Friendly Receiver-Based Layered Multicast Protocol," PhD Thesis, College of Arts and Sciences, Universiti Utara Malaysia, 2008.

[129] L. F. Perrone and Y. Yuan, "Modeling and Simulation Best Practices for Wireless Ad Hoc Networks," in *Simulation Conference, 2003. Proceedings of the 2003 Winter*, vol. 1. IEEE, 2003, pp. 685–693.

[130] T. R. Andel and A. Yasinsac, "On the credibility of MANET simulations," *Computer*, vol. 39, no. 7, pp. 48–54, 2006.

[131] S. Bajaj, L. Breslau, D. Estrin, K. Fall, S. Floyd, P. Haldar, M. Handley, A. Helmy, J. Heidemann, P. Huang, S. Kumar, S. McCanne, R. Rejaie, P. Sharma, K. Varadhan, Y. Xu, H. Yu, and D. Zappala, "Improving Simulation for Network Research," University of Southern California, Tech. Rep. 99-702b, Mar. 1999, revised September 1999, to appear in IEEE Computer. [Online]. Available: http://www.isi.edu/~johnh/PAPERS/Bajaj99a.html

[132] E. Weingartner, H. Vom Lehn, and K. Wehrle, "A Performance Comparison of Recent Network Simulators," in *IEEE International Conference on Communications, ICC'09*. IEEE, 2009.

[133] E. Schoch, M. Feiri, F. Kargl, and M. Weber, "Simulation of Ad Hoc Networks: NS-2 Compared to JiST/SWANS," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, p. 36.

[134] "The Network Simulator NS-2," http://www.isi.edu/nsnam/ns/.

[135] T. R. Henderson, S. Roy, S. Floyd, and G. F. Riley, "NS-3 Project Goals," in *Proceeding from the 2006 workshop on ns-2: the IP network simulator*. ACM, 2006.

[136] L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, and M. Gerla, "Glomosim: A Scalable Network Simulation Environment," *UCLA Computer Science Department Technical Report*, vol. 990027, p. 213, 1999.

[137] A. Sobeih, W.-P. Chen, J. C. Hou, L.-C. Kung, N. Li, H. Lim, H.-Y. Tyan, and H. Zhang, "J-sim: A Simulation Environment for Wireless Sensor Networks," in *Proceedings of the 38th annual Symposium on Simulation*. IEEE Computer Society, 2005, pp. 175–187.

[138] X. Chang, "Network Simulations with OPNET," in *Simulation Conference Proceedings, 1999 Winter*, vol. 1. IEEE, 1999, pp. 307–314.

[139] O. Modeler, "OPNET Technologies Inc," 2009.

[140] Q. N. Simulator, "Scalable network technologies," *Inc.[Online]. Available: www. qualnet. com*, 2011.

[141] A. Varga, "OMNet++." [Online]. Available: http://www.omnetpp.org/

[142] M. K0ksal, "A Survey of Network Simulators Supporting Wireless Networks," *lnea: http://www. ceng. metu. edu. trSurvey*, vol. 20, 2008.

[143] G. F. Lucio, M. Paredes-Farrera, E. Jammeh, M. Fleury, and M. J. Reed, "OPNET Modeler and NS-2: Comparing the Accuracy of Network Simulators for Packet-level Analysis Using a Network Testbed," *WSEAS Transactions on Computers*, vol. 2, no. 3, pp. 700–707, 2003.

[144] P. P. Garrido, M. P. Malumbres, and C. T. Calafate, "NS-2 vs. OPNET: A Comparative Study of the IEEE 802.11 e Technology on MANET Environments," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, p. 37.

[145] X. Xian, W. Shi, and H. Huang, "Comparison of OMNET++ and other Simulator for WSN Simulation," in *Industrial Electronics and Applications, 2008. ICIEA 2008. 3rd IEEE Conference on*. IEEE, 2008, pp. 1439–1443.

[146] K. Fall and K. Varadhan, "The Network Simulator (ns-2)," *URL: http://www. isi. edu/nsnam/ns*, 2007.

[147] S. Floyd and V. Paxson, "Difficulties in Simulating the Internet," *IEEE/ACM Transactions on Networking (TON)*, vol. 9, no. 4, pp. 392–403, 2001.

[148] T. Issariyakul and E. Hossain, *Introduction to Network Simulator NS2*. Springer Verlag, 2008.

[149] J. Heidemann, K. Mills, and S. Kumar, "Expanding Confidence in Network Simulations," *Network, IEEE*, vol. 15, no. 5, pp. 58–63, 2001.

[150] K. Pawlikowski, H.-D. Jeong, and J.-S. Lee, "On Credibility of Simulation Studies of Telecommunication Networks," *Communications Magazine, IEEE*, vol. 40, no. 1, pp. 132–139, 2002.

[151] E. Altman and T. Jiménez, "NS Simulator for Beginners," *Synthesis Lectures on Communication Networks*, vol. 5, no. 1, pp. 1–184, 2012.

[152] D. Cavalcanti, D. Agrawal, C. Cordeiro, B. Xie, and A. Kumar, "Issues in Integrating Cellular Networks WLANs, and MANETs: A Futuristic Heterogeneous Wireless Network," *IEEE Wireless Communications*, vol. 12, no. 3, pp. 30–41, 2005.

[153] I. . L. S. Committee *et al.*, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, Std., 1999.

[154] B. P. Crow, I. Widjaja, L. Kim, and P. T. Sakai, "IEEE 802.11 Wireless Local Area Networks," *IEEE Communications Magazine*, vol. 35, no. 9, pp. 116–126, 1997.

[155] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," Internet Engineering Task Force, RFC 3561, 1981. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3561.txt

[156] D. Johnson, Y. Hu, D. Maltz *et al.*, "The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4," Internet Engineering Task Force, RFC 4728, 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4728.txt

[157] M. Arefin, M. Khan, and I. Toyoda, "Performance Analysis of Mobile Ad-hoc Network Routing Protocols," in *Informatics, Electronics Vision (ICIEV), 2012 International Conference on*, 2012, pp. 935–939.

[158] M. Ikeda, E. Kulla, M. Hiyama, L. Barolli, M. Younas, and M. Takizawa, "TCP Congestion Control in MANETs for Multiple Traffic Considering Proactive and Reactive Routing Protocols," in *Network-Based Information Systems (NBiS), 2012 15th International Conference on*, 2012, pp. 156–163.

[159] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," *Kluwer International Series in Engineering and Computer Science*, pp. 153–179, 1996.

[160] S. Floyd, "Metrics for the Evaluation of Congestion Control Mechanisms," Internet Engineering Task Force, RFC 5166, Mar. 2008. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5166.txt

[161] D. Zhou, W. Song, and Y. Cheng, "A Study of Fair Bandwidth Sharing with AIMD-Based Multipath Congestion Control," *Wireless Communications Letters, IEEE*, vol. 2, no. 3, pp. 299–302, 2013.

[162] S. Manaseer, "On Backoff Mechanisms for Wireless Mobile Ad Hoc Networks," Ph.D. dissertation, University of Glasgow, 2010.

[163] J. S. Kim and R. J. Dailey, "Confidence Intervals and Sample Size," *Biostatistics for Oral Healthcare*, pp. 113–126, 2008.

[164] L. Ding, W. Zhang, and W. Xie, "Modeling TCP Throughput in IEEE 802.11 Based Wireless Ad Hoc Networks," in *Communication Networks and Services Research Conference, 2008. CNSR 2008. 6th Annual*. IEEE, 2008, pp. 552–558.

[165] H. Xiao, Y. Zhang, J. Malcolm, B. Christianson, and K. C. Chua, "Modelling and Analysis of TCP Performance in Wireless Multihop Networks," *Wireless Sensor Network*, vol. 2, no. 7, pp. 493–503, 2010.

[166] F. Azimi and P. Bertok, "An Analytical Model of TCP Flow in Multi-hop Wireless Networks," in *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*. IEEE, 2010, pp. 88–95.

[167] A. A. Kherani and R. Shorey, "Performance Modeling and Analysis of TCP over Wireless Ad Hoc Networks with IEEE 802.11 MAC," *manuscript available at http://wwwsop. inria. fr/mistral/personnel/Arzad-Alam. Kherani*.

[168] E. Ghadimi, A. Khonsari, A. Diyanat, M. Farmani, and N. Yazdani, "An Analytical Model of Delay in Multi-hop Wireless Ad Hoc Networks," *Wireless networks*, vol. 17, no. 7, pp. 1679–1697, 2011.

[169] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, pp. 535–547, 2000.

[170] C. S. Lent, *Learning to Program with MATLAB: Building GUI Tools*. Wiley, 2013.

[171] R. Pratap, *Getting Started with MATLAB*. Saunders College Publishing, 2002.

[172] V. Jacobson, R. Braden, and D. Borman, "TCP Extensions for High Performance," Internet Engineering Task Force, RFC 1323, 1992. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2988.txt

[173] M. Jain and C. Dovrolis, *Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput*. ACM, 2002, vol. 32, no. 4.

[174] S. Prasanthi, S.-H. Chung, and Y.-H. Jo, "A New Loss Recovery Algorithm for Increasing the Performance of TCP Over Wireless Mesh Networks," in *Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on*. IEEE, 2012, pp. 229–236.

[175] F. R. Armaghani and S. S. Jamuar, "TCP-MAC interaction in multi-hop ad-hoc networks," in *Mobile Ad-Hoc Networks: Applications*, X. Wang, Ed. InTech, 2011, pp. 401–426.

[176] H.-J. Lee and J.-T. Lim, "Fair Congestion Control over Wireless Multihop Networks," *Communications, IET*, vol. 6, no. 11, pp. 1475–1482, 2012.

[177] R. L. West and L. H. Turner, *Introducing Communication Theory: Analysis and Application*. New York, NY: McGraw-Hill, 2010.

[178] M. L. McLaughlin, *Communication Yearbook 10*. Routledge, 2012.