SCTP-BASED CONCURRENT MULTIPATH TRANSFER IN THE CONTEXTS OF MULTIHOP WIRELESS NETWORKS AND TCP-FRIENDLINESS

by

Ilknur Aydin

A dissertation submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science

Spring 2010

SCTP-BASED CONCURRENT MULTIPATH TRANSFER IN THE CONTEXTS OF MULTIHOP WIRELESS NETWORKS AND TCP-FRIENDLINESS

by

Ilknur Aydin

Approved: _____

David Saunders, Ph.D.
Chair of the Department of Computer and Information Sciences

Approved: _____

George H. Watson, Ph.D.
Dean of the College of Arts and Sciences

Approved: _____

Debra Hess Norris, M.S.
Vice Provost for Graduate and Professional Education

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Chien-Chung Shen, Ph.D.
Professor in charge of dissertation

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Paul D. Amer, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Adarsh Sethi, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Stephan Bohacek, Ph.D.
Member of dissertation committee

To my beloved babanne

(1938 – 2007)

ACKNOWLEDGEMENTS

I would like to thank my PhD adviser Prof. Chien-Chung Shen for his support and patience with me during my graduate studies at University of Delaware (UD). His hard-working character and dedication to his work have been an inspiration for me. I also thank to my PhD committee members Profs. Paul Amer, Adarsh Sethi, and Stephan Bohacek. In particular, Prof. Amer's comments on my work has been crucial for the scientific strength of my dissertation. I was fortunate be in contact with Prof. Bohacek over the last couple of years. Our discussions have been fruitful and helped a lot in terms of shaping and finalizing my work.

I thank my collaborators Janardhan Iyengar and Phillip Conrad. Janardhan has not only been a great person to have technical discussions but also has been a great friend since the early years of graduate school. Phill has always inspired me with his enthusiasm about research and teaching.

I thank past and current members of the DEGAS Networking group at Computer and Information Sciences department of UD: Chaiporn Jaikaeo (Art), Chavalit Srisathapornphat (Lit), Zhuochuan Huang (Joe), Sonny Rajagopalan (Sonny), Naveen Manicka, Justin Yackoski, Xiaofeng Han, and Ke Li. Art and Lit had always patiently answered my questions in the early years of graduate school. Sonny has been a great mentor towards the end of my studies, shaking and reminding me what it means to be a PhD student and how to cross the finish line.

I had two other great friends at UD, Sreedevi Sampath and Usha Rani Annepu, who made the first two years of graduate school bearable. We have gone through crazy

assignment and project deadlines together but always had great fun. I cherish our continuing friendship.

I have been fortunate to have many friends in US that helped me in several aspects during my graduate studies. Filiz Ucar-Ozkan and Betul Buyukkurt have been my dear friends since undergraduate and have only been a phone call away whenever I needed. Betul Arican was the first person I met in Newark and has been my “big sister” since then. Ali Kemal and Neslihan Derin have been real friends and helped me in many regards whenever I needed. Their little daughter Zeynep has been a constant source of joy for me. Seda Yilmaz had been a great roommate for several years. I thank Seda for teaching me how to drive, all the fun trips that we have had, and our continuing friendship. Tugba Soycan’s support and concern for me has been immense especially over the last couple of years. Tugba has been very thoughtful and always check on me to make sure that I am still alive and kicking.

I thank my parents for their love and support. I thank my dad for giving me and my siblings the freedom to do whatever we want and wherever. My mom has been a great source of support for me. Her cheerful manners, support and prayers brought me to these days. My sisters Ebru, Yasemin, and Seda have always been with me. It has always been a great fun to hang out with them whenever I visited Turkey. My brothers Maksut and Muharrem have supported me in other ways. Maksut has been very gracious and generous to help me out even if it meant to put his plans on hold. Muharrem’s two summer visits have given me a great morale boost when I most needed.

I now also have my Indian family. I particularly thank Baba, Ammaji, Nafiya Aapa, Nooman Bhaijan, and Haseena Didi for their wholehearted support and prayers to finish my work. The last but not the least, I thank my dearest husband Abdul Rumaiz. I have known you as a nice person and decent man for long number of years. I thank you for the love, support, and mentorship that you provided me. I did not start my PhD journey with you but I could only finish the PhD because of you!

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

A multihome-capable transport layer protocol allows an application to transmit data via multiple (disjoint) paths simultaneously, a scheme termed concurrent multipath transfer (CMT). SCTP is an IETF-standardized transport layer protocol with built-in multihoming capability. In prior work, a CMT protocol using SCTP multihoming (termed SCTP-based CMT) was proposed and investigated for improving application throughput in wired networks. In that effort, SCTP-based CMT was studied in (bottleneck-independent) wired networking scenarios with ns-2 simulations. This dissertation studies SCTP-based CMT in two specific contexts using QualNet simulations: (i) CMT over Multihop Wireless Networks (MWNs), and (ii) TCP-friendliness of CMT in the Internet.

CMT over MWNs: Given the recent advances in multiple-radio nodes, multi-channel radios, and multi-path routing, more multihomed nodes are deployed in the wireless networks. This trend motivated us to study two specific issues in the context of CMT over MWNs. The first issue concerns the performance of CMT over MWNs, where we studied how the contention-induced losses and the wireless channel characteristics impact the performance of CMT. We found that similar to the wired cases, CMT over MWNs showed better performance than one single-homed SCTP association and even the ideal AppStripe application, when the receiver buffer (rBuf) was unconstrained. For the cases of constrained rBuf, we showed that considering the bandwidth limitations of multihop wireless networks compared to their wired counterparts, rBuf sizes bigger than 128 KB can be sufficiently enough not to restrain the CMT performance. Overall, we concluded that applications will benefit from using CMT in the MWNs context when they have sufficiently large rBuf. The second issue concerns the acknowledgment (ACK) mechanism

of CMT, where we investigated different design choices for the ACK mechanism of CMT in MWN to mitigate the effects of contention-induced losses in the multihop wireless channels among data and ACK traffic. According to the ACK policy of the original CMT, an ACK packet is sent to the path where the latest DATA packet arrives from. Therefore, a CMT receiver may send ACKs packets to any one of the (return) paths. We evaluated an alternative ACK policy which sends ACK packets to the same (return) path during the entire association lifetime. We found out that especially as the multihop wireless path gets longer and the number of simultaneous CMT sub-flows increases, the alternative ACK policy shows better performance, even when the quality of the paths differ. This is because the inter-flow interference (between the DATA and the ACK packets) within a CMT flow using the alternative ACK policy is less than the inter-flow interference within a CMT flow with the original ACK policy, which causes the aggregated sending rate of the CMT flow using the alternative ACK policy to become higher than the aggregated sending rate of the CMT flow using the original CMT ACK policy.

TCP-friendliness of CMT: TCP has been the de facto transport layer protocol used for reliable communications in the Internet. Following the Internet's infamous congestion collapse in the late 80s, congestion control algorithms were incorporated into TCP. The doctrine of TCP-friendliness then appeared at the end of 90s as a response to the increase of non-TCP traffic in the Internet to protect the health and stability of the Internet. The TCP-friendliness doctrine states that the sending rate of a non-TCP flow should be approximately the same as that of a TCP-flow under the same conditions (RTT and packet loss rate). In addition, the doctrine states that non-TCP transport layer protocols should implement some form of congestion control to prevent congestion collapse. However, recent developments of multihoming and CMT challenge this traditional definition of TCP-friendliness which was introduced for single-homed (single-path) end-to-end connections. In this dissertation, we surveyed historical developments of the TCP-friendliness concept and argued that the original TCP-friendliness doctrine should be extended to incorporate

multihoming and SCTP-based CMT. Since CMT is based on (single-homed) SCTP, as a first step, we investigated TCP-friendliness of single-homed SCTP. We discovered that although SCTP's congestion control mechanisms were intended to be "similar" to TCP's, being a newer protocol, SCTP specification has some of the proposed TCP enhancements already incorporated which results in SCTP performing better than TCP. Therefore, SCTP can obtain larger share of the bandwidth when competing with a TCP flavor that does not have similar enhancements. We concluded that SCTP is TCP-friendly but achieves higher throughput than TCP, due to SCTP's better loss recovery mechanisms just as TCP-SACK or TCP-Reno performs better than TCP-Tahoe. Then, we designed an experimental framework to investigate the TCP-friendliness of CMT according to the traditional doctrine of TCP-friendliness. Via QualNet simulations, we measured the sending rate of one two-homed CMT flow (containing two CMT subflows) and two SCTP flows and the impact of CMT and two SCTP flows on the other TCP flows in the network while sharing a tight link. We found out that one two-homed CMT association has similar or worse performance (for smaller number of competing TCP flows) than the aggregated performance of two independent, single-homed SCTP associations while sharing the link with other TCP connections, for the reason that a CMT flow creates a burstier data traffic than independent SCTP flows. When compared to the aggregated performance of two-independent TCP connections, one two-homed CMT obtains higher share of the tight link bandwidth because of better loss recovery mechanisms in CMT (as CMT inherits all the built-in TCP enhancements in SCTP). In addition, sharing of ACK information makes CMT more resilient to losses. Although CMT obtains higher throughput than two independent TCP flows, CMT's AIMD-based congestion control mechanism allows other TCP flows to coexist in the network. Therefore, we concluded that CMT is TCP-friendly, similar to two TCP-Reno flows are TCP-friendly when compared to two TCP-Tahoe flows.

To facilitate the above research, we developed SCTP and SCTP-based CMT simulation modules in the QualNet network simulator. We also cross-checked the correctness

of our SCTP QualNet module with SCTP ns-2 module using wired networking scenarios. We showed that though these two simulators are different, the performance trends between the ns-2 and the QualNet implementations are similar, validating the QualNet implementation of SCTP and CMT.

This page intentionally left blank

Chapter 1

INTRODUCTION

A host is *multihomed*, if the host has multiple network addresses [29]. We are seeing more multihomed hosts connected to the networks and the Internet. For instance, PCs with one Ethernet card and one wireless card, and cell phones with dual Wi-Fi and 3G interfaces are already common realities. Nodes with multiple radios and radios operating over multiple channels are being deployed [1, 2]. In addition, Wi-Fi wireless interface cards became so cheap that nodes with multiple Wi-Fi cards and wireless mesh networks (or testbeds) with multiple radios become practical [76, 92].

A transport protocol supports multihoming, if it allows multihomed hosts at the end(s) of a single transport layer connection. That is, a *multihome-capable transport protocol* allows a *set* of network addresses, instead of a single network address, at the connection end points. When each network address is bound to a different network interface card connected to a different physical network, multiple physical communication paths become available between a source host and a destination host (Figure 1.1).

A multihome-capable transport protocol can accommodate *multiple paths* between a source host and a destination host within a *single* transport connection. Therefore, technically, a multihomed transport protocol allows simultaneous transfer of application data through different paths from a source host to a destination host, a scheme termed *Concurrent Multipath Transfer (CMT)*. Network applications can benefit from CMT in many ways such as fault-tolerance, bandwidth aggregation, and increased application throughput.

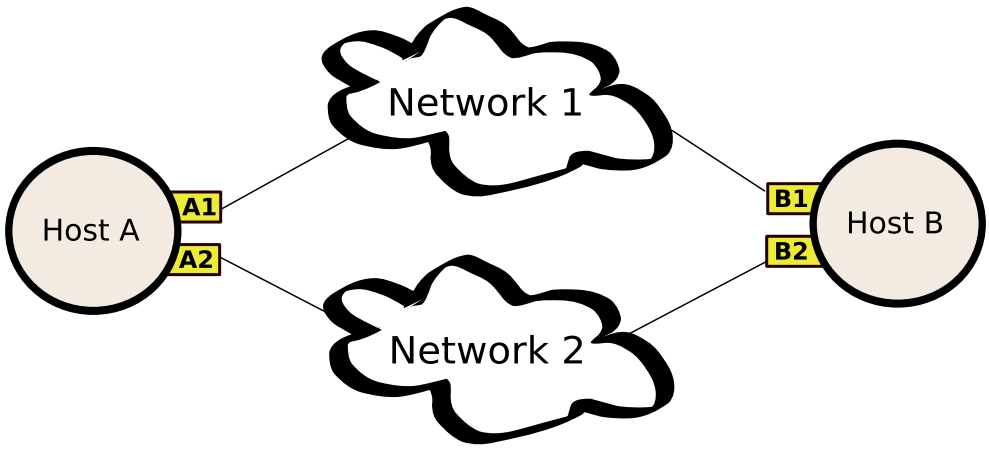


Figure 1.1: Example of multihoming (with multiple disjoint paths)

The current transport layer workhorses of the Internet, TCP and UDP, do not support multihoming. However, the *Stream Control Transmission Protocol (SCTP)* [98, 99] has built-in multihoming support. Since SCTP supports multihoming natively, SCTP has the capability to realize CMT for the network applications. In this dissertation, we study SCTP-based CMT [63] in two research contexts: multihop wireless networks (MWNs) and TCP-friendliness in the Internet.

The following section (Section 1.1) presents a primer on SCTP, followed by the scope of the problems studied in this dissertation (Section 1.2). We conclude this chapter by describing the organization of the dissertation in Section 1.3.

1.1 SCTP Primer

SCTP was originally designed to transport telephony signaling messages over IP networks. Later on, SCTP, supported by IETF, was found useful as a general purpose, reliable transport protocol for the Internet. SCTP provides services similar to TCP's (such as connection-oriented reliable data transfer, ordered data delivery, window-based and TCP-friendly congestion control, flow control) and UDP's (such as unordered data delivery, message-oriented). In addition, SCTP provides other services neither TCP nor

UDP offers (such as multihoming, multistreaming, protection against SYN flooding attacks) [100]. In the SCTP jargon, a transport layer connection is called an *association*. Each SCTP packet, or generally called *SCTP protocol data unit (SCTP-PDU)*, contains an SCTP *common header* and multiple data or control *chunks*[1].

1.1.1 SCTP Multihoming

One of the most innovative features of SCTP is its built-in multihoming capability where an association can be established between a *set* of local and a *set* of remote IP addresses as opposed to a *single* local and a *single* remote IP address as in a TCP connection. In an SCTP association, each SCTP endpoint chooses *a single port*. Although multiple IP addresses are possible to reach one SCTP endpoint, only one of the IP addresses is specified as the *primary* IP address to transmit data to the destination endpoint.

The *reachability* of the multiple destination addresses are monitored by SCTP with periodic *heartbeat* control chunks sent to the destination IP addresses. The application data is sent *only* to the primary destination address of an SCTP endpoint. However, if the primary address of an SCTP endpoint fails, one of the alternate destination addresses is chosen to transmit the data dynamically, a process termed *SCTP failover*.

In Figure 1.1, both Host A and Host B have two network interfaces, where each interface has one single IP address A1, A2 and B1, B2, respectively. Each interface is connected to a separate (i.e., physically disjoint) network (Network_1 and Network_2). Therefore, two end-to-end paths exist between Host A and Host B (A1 to B1 and A2 to B2). One SCTP association accommodates all of the IP addresses of each host and multiple paths between the hosts as follows: $([A1, A2 : portA], [B1, B2 : portB])$. Note that, two different TCP connections are needed to accommodate all the IP addresses and the two paths in the same figure, namely $([A1 : portA], [B1 : portB])$ and $([A2 : portA], [B2 : portB])$.

[1] Formats of SCTP-PDU, common header, data chunk, and some of the control chunks are presented in Appendix E.

1.1.2 Concurrent Multipath Transfer (CMT) with SCTP

Although the standard SCTP [98] supports multiple IP addresses to reach a destination host, only one of the IP addresses, named *the primary IP address*, is used as a destination at any time, to originally transmit application data to a destination host. The IP addresses other than the primary IP address are only used for retransmitting data during failover for the purpose of fault tolerance. Therefore, in reality, the standard SCTP does not fully utilize its potential to facilitate CMT for applications. Research efforts on the concurrent use of the multiple paths within an SCTP association continue [33, 44, 60, 75, 88, 108]. The SCTP-based CMT[2] proposed by Iyengar et. al. [60, 63] is the first SCTP research effort aiming to increase application throughput through concurrency.

Because paths may have different end-to-end delays, *naively*[3] transmitting data to multiple destination addresses (over different paths) within an SCTP association will often result in out-of-order arrivals at a multihomed SCTP receiver. Out-of-order arrivals have negative effects on SCTP throughput due to spurious fast retransmissions, and prevent congestion window growth even when ACKs continue arriving at the sender. CMT [63] proposed the following three algorithms to mitigate the effects of reordering at the receiver.

- *Split Fast Retransmit (SFR) algorithm.* Out-of-order arrivals at the receiver cause the receiver to send duplicate SACKs or SACKs with gap ack blocks[4] which in turn cause spurious fast retransmissions at the sender. The SFR algorithm addresses this issue by introducing a virtual queue per destination and deducing the missing

[2] From now on, any mention of *CMT*, *SCTP-based CMT*, or *SCTP CMT* refers to the CMT proposed in [60, 63].

[3] That is, simply using the standard SCTP without any modifications.

[4] "Each Gap Ack Block acknowledges a subsequence of TSNs received following a break in the sequence of received TSNs." [98]

reports per TSN (Transmission Sequence Number) correctly to prevent unnecessary fast retransmissions.

- *Cwnd Update for CMT (CUC) algorithm.* The cwnd evolution algorithm of the standard SCTP allows the cwnd of a path to be updated only when a new cumulative ACK arrives at the sender. Therefore, with CMT when ACK packets with unchanged cumulative ACKs (caused by the reordering due to the use of simultaneous paths) arrive at the sender, the cwnd value of the paths is not updated. CUC addresses this issue by tracking the latest TSN received in-order per destination and hence avoids unnecessary reduction in the congestion window updates.

- *Delayed ACK for CMT (DAC) algorithm.* The standard SCTP employs a built-in delayed SACK algorithm to reduce the ACK traffic. When reordering is observed at the receiver, the delayed ack algorithm of the standard SCTP states that the receiver should immediately send an ACK without waiting any further. However, with CMT, there is frequent reordering which will cause the ACK traffic not to be delayed. The DAC algorithm handles this issue by forcing the receiver to sometimes delay sending ACKs even when reordering is observed at the receiver to help reducing the ACK traffic.

The availability of multiple destination addresses in an SCTP association allows an SCTP sender to select one destination address for the retransmissions. However, in standard SCTP since only the primary destination address is used to send new data, there is no sufficient information about the condition of all other paths. On the other hand, since CMT simultaneously uses all the paths, a CMT sender maintains accurate information regarding the condition of all the paths. Therefore, a CMT sender can better select a path to send retransmissions. CMT includes the following retransmission policies.

- RTX-SAME: All of the retransmissions of a data chunk are always sent to the same destination address that the original transmission of the data chunk is sent to.

- RTX-CWND: A retransmission is sent to the active destination address with the highest cwnd value.
- RTX-SSTHRESH: A retransmission is sent to the active destination address with the highest ssthresh value.
- RTX-ASAP: A retransmission is sent to the active destination with space available in the cwnd of the path, at the time of the retransmission.

1.2 Scope of the Dissertation

In this dissertation, we study SCTP CMT in two research contexts: multihop wireless networks (MWNs) and TCP-friendliness in the Internet. The following subsections describe these two research efforts.

1.2.1 CMT over MWNs

In a MWN, nodes communicate with each other using wireless radios. Wireless medium (air) is a common resource shared among the nodes and a medium access control (MAC) protocol coordinates the access to the shared medium. In addition, communications between two end nodes go through multiple intermediate nodes (hops). Therefore, each node in the network is also a relay (router) for the other nodes in the network.

Iyengar et. al. proposed an SCTP-based CMT [63, 60] to achieve increased application throughput and studied the performance of CMT in various wired scenarios. In this dissertation, we study STCP-based CMT over MWNs. Recent advances in multiple radio nodes, multichannel radios [1, 2], and multipath routing protocols have fueled such research.

In this dissertation, we investigate the performance of CMT in IEEE 802.11 [12] based MWNs using QualNet simulations. In particular, we explore how contention-induced losses and multihop forwarding impact application throughput over CMT in the

context of MWN. In addition to evaluating the performance of the original CMT algorithms over MWNs, we also investigate how alternative designs in transmission of CMT acknowledgments (ACK) impact the performance of CMT.

1.2.2 TCP-Friendliness with CMT

TCP is the de facto reliable transport protocol used in the Internet. Following the infamous Internet *congestion collapse* in 1986 [65], several congestion control algorithms were incorporated into TCP to protect the stability and health of the Internet [65]. As a direct response to widespread use of non-TCP transport protocols, the concept of *TCP-friendliness* emerged [77]. Briefly, TCP-friendliness states that the sending rate of a non-TCP flow should be approximately the same as that of a TCP flow under the same conditions (RTT and packet loss rate) [55]. In addition, a non-TCP transport protocol should implement some form of congestion control to prevent congestion collapse. Since 1990s, new developments, such as multihoming and CMT, challenge this traditional definition of TCP-friendliness which was originally introduced for single-path end-to-end connections. For instance, recently, there are substantial activities in the IETF and the IRTF mailing lists (such as *tmrg*, *tsvwg*, *iccrg*, and *end2end-interest*) discussing the definition of TCP-friendliness and other related issues (such as compliance with TCP-friendly congestion control algorithms, what can cause congestion collapse in the Internet, Internet-friendly vs. TCP-friendly algorithms, fairness of flow rate fairness).

In this dissertation, we survey the historical development of TCP-friendliness and argue that the existing definition should be extended to incorporate SCTP CMT and multihoming. Since SCTP CMT is based on (single-homed) SCTP, we first investigate TCP-friendliness of single-homed SCTP[5]. We then study TCP-friendliness of SCTP CMT according to the traditional definition of TCP-friendliness [77] using QualNet simulations.

[5] Note that, although SCTP has "similar" congestion control mechanisms as TCP, there are subtle differences between (single-homed) SCTP and TCP.

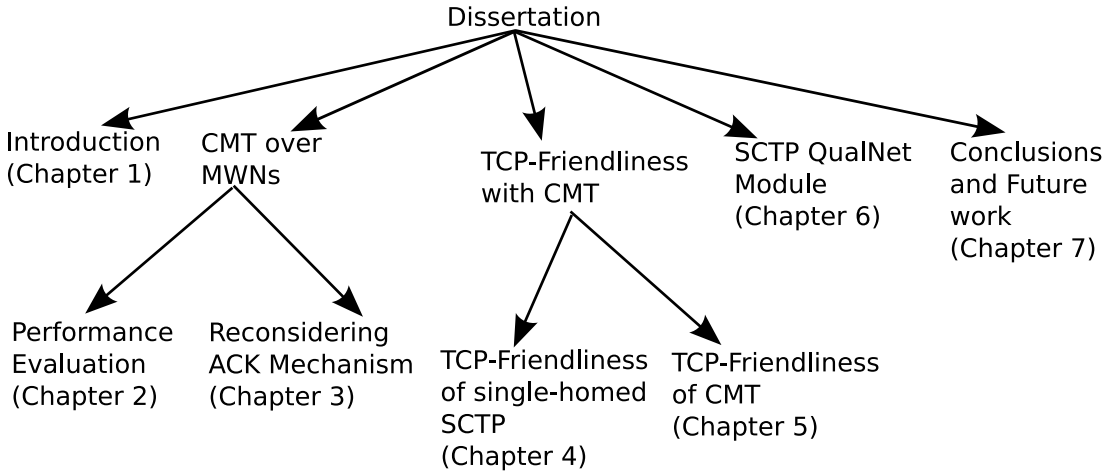


Figure 1.2: Organization of the dissertation

1.3 Organization of the Dissertation

The organization of this dissertation is depicted in Figure 1.2. We study transport layer multihoming, specifically SCTP multihoming and CMT, in two contexts, multihop wireless networks (MWNs) and TCP-friendliness in the Internet.

Chapter 2 and Chapter 3 are related to the first context, MWNs. In Chapter 2, we investigate the performance of SCTP CMT in the context of MWNs. We compare the performance of CMT over MWNs vs. CMT over wired networks and study how the characteristics of multihop wireless networks (such as multihop forwarding, hidden terminals, and contention-based losses) impact the performance of CMT. In the following chapter (Chapter 3), we focus on ACK mechanisms of CMT and evaluate alternative designs when sending ACKs to the CMT sender in the context of MWNs.

The following two chapters, Chapter 4 and 5, are related to the second context, TCP-friendliness in the Internet. The original design goal of SCTP CMT was to have application throughput at least as high as the aggregated application throughput of multiple independent (single-homed) SCTP associations. Note that, Iyengar et. al. studied the performance of CMT only in bottleneck-independent, wired network topologies. However, the aggressiveness of SCTP CMT without the assumption of bottleneck-independent

topology is still unknown. As a first step in understanding TCP-friendliness of SCTP CMT, we investigate TCP-friendliness of single-homed SCTP in a shared *tight link* [66] topology in Chapter 4. In Chapter 5, we study TCP-friendliness of CMT.

To facilitate our research, we implemented SCTP with various extensions of SCTP and SCTP CMT in the QualNet network simulator [3]. Chapter 6 describes the implemented SCTP QualNet module [21]. We also cross-checked the SCTP QualNet module with the SCTP ns-2 module in wired simulation scenarios to validate the correctness of the SCTP QualNet module.

Finally, Chapter 7 summarizes the contributions of the dissertation and future research plan.

Chapter 2

PERFORMANCE OF CMT OVER MWNS

In this chapter, we study the performance of CMT over multihop wireless networks (MWNs) [26]. We first describe our problem in Section 2.1. In Section 2.2, we give a background on the IEEE 802.11 DCF MAC protocol. Section 2.3 presents our simulation design followed by the hypotheses to be validated and the analysis of the simulation results in Section 2.4. Section 2.5 discusses related work. Section 2.6 concludes the chapter.

2.1 Problem Statement

Iyengar et. al. proposed a CMT protocol using SCTP multihoming to improve application throughput, and studied the performance of CMT over wired networks using ns-2 simulations. Given advances in multi-radio nodes, multichannel radios [1, 2], and multipath routing, more multihomed nodes are deployed in wireless networks, which motivates us to study SCTP-based CMT over wireless networks. In particular, we investigate application throughput of SCTP-based CMT over static IEEE 802.11-based multihop wireless networks using QualNet simulations. In this work, we considered a specific type of MWNs, where (i) all the nodes are stationary, (ii) there is no connection to a wired network or the Internet, and (ii) the medium access is orchestrated by the IEEE 802.11 DCF MAC protocol [12]. Such MWNs are motivated by *community mesh networks* [13], for instance. Within a community mesh network, although some of the nodes are directly connected to the Internet, nodes may directly communicate among each other over multihop paths to engage in peer-to-peer interactions.

is one type of network that motivates our CONfiguration. Some of the nodes within a community mesh network are connected to the Internet, but other nodes may talk to each other directly over multihop paths. We can consider a scenario where one neighbor node is downloading a file from another neighbor without the need to connect to the Internet.

2.2 Background

This section describes the details of the IEEE 802.11 DCF MAC protocol.

2.2.1 IEEE 802.11 DCF Primer

In wireless networks, spectrum is a shared and scarce resource that demands controlled access. The IEEE 802.11 Distributed Coordination Function (DCF) MAC protocol [12] is the *de facto* medium access standard used in (multihop) wireless networks.

IEEE 802.11 DCF is basically a carrier sense multiple access (CSMA) scheme with collision avoidance (CA) and positive acknowledgments (M-ACKs[1]). A node wanting to transmit data (M-DATA[2]) first senses the medium, termed *physical carrier sensing*. If the medium is not being used by the transmissions of other nodes, the node transmits its M-DATA. The receiver responds with an M-ACK after receiving the M-DATA. IEEE 802.11 DCF also uses *virtual carrier sensing* for collision avoidance. Basically, each IEEE 802.11 DCF PDU contains a duration field indicating how long it will take the sender node to transmit its M-PDU. Other nodes overhearing the transmitted M-PDU, look at the duration field and determine the minimum time that they need to defer their transmissions (maintained in the network allocation vector (NAV) of each node). IEEE 802.11 DCF also includes an optional RTS/CTS mechanism to reserve the channel before any M-DATA transmission. The sender node first sends an RTS (Request-To-Send)

[1] M-ACK is an M-PDU carrying acknowledgements at the MAC layer. *PDU* stands for protocol data unit and *M-PDU* stands for MAC layer PDU.

[2] M-DATA is an M-PDU carrying data at the MAC layer.

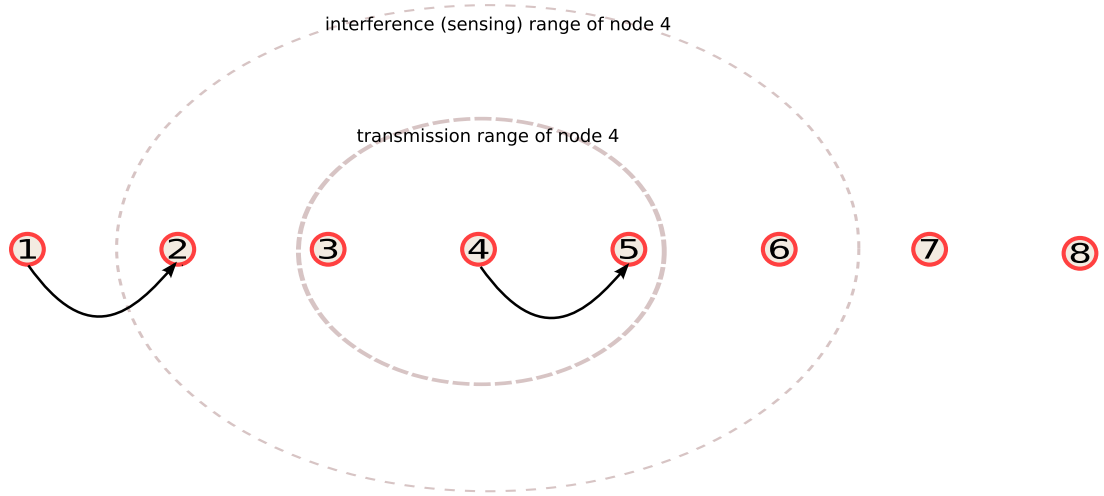


Figure 2.1: *A multihop wireless chain topology.* Node 4 is a hidden terminal for the transmission from node 1 to node 2.

message up to a number of times[3] to reserve the channel for its data transmission. If the sender does not receive a CTS (Clear-To-Send) after some number of tries, the sender drops the M-DATA and reports link failure to the upper layer. After getting a CTS, the sender then transmits the M-DATA up to a number of times[4] until the sender gets an M-ACK from the receiver. If the sender does not receive an M-ACK after so many tries, the M-DATA is dropped and an error is reported to the upper layer. The RTS and CTS messages also include the duration of the entire transmission so that any other node hearing the RTS/CTS exchange update its NAV accordingly to defer its transmissions, if any.

2.2.2 Hidden Terminals and Spatial Channel Reuse

Even though IEEE 802.11 DCF employs the RTS/CTS mechanism, it is still prone to the *hidden terminal problem*, which occurs due to the interference caused by another transmission in the neighborhood of a receiver node. In Figure 2.1, each node in the chain is equipped with an IEEE 802.11 wireless interface with transmission range of 250 meters

[3] *SHORT RETRY LIMIT*

[4] *LONG RETRY LIMIT*

and carrier sensing (and interference) range of 550 meters[5]. Nodes are 200 meters apart and each node can communicate only with its direct neighbors. Let's assume that there are two data transmissions in the network, one from node 1 to node 2 and the other from node 4 to node 5. Before starting the data transmission, node 1 sends an RTS to node 2, node 2 responds with a CTS. Note that node 4 can not hear (decode) the RTS and CTS messages because node 4 is outside the transmission range of nodes 1 and 2. Therefore, node 4 does not defer its transmission to node 5, while node 1 is transmitting to node 2. Thus, transmission at node 4 interferes with the reception at node 2 (since node 2 is within the interference range of node 4). Node 4 becomes a hidden terminal for the transmission from node 1 to node 2 and causes the loss of data (*contention-induced loss*).

The interference relationship among the nodes due to hidden terminals is the main bottleneck of IEEE 802.11 based multihop wireless networks. In particular, the use of the same channel by two different but simultaneous transmissions is possible only if the two transmissions are not interfering with each other (*spatial channel reuse*). For instance, in Fig. 2.1, transmissions between nodes 1 and 2 and between nodes 5 and 6 may occur simultaneously, but transmissions between nodes 1 and 2 and between nodes 4 and 5 can not happen at the same time.

2.3 Simulation Design

We implemented SCTP CMT in QualNet. Before running any CMT over MWNs, we validated the correctness of our SCTP CMT QualNet module with the SCTP CMT ns-2 module developed by A. Caro and J. Iyengar [32]. In this validation study, we repeated a subset of the CMT *over wired networks* ns-2 simulation experiments from [63] using our SCTP CMT QualNet module. The results confirmed that our SCTP CMT QualNet implementation is compatible to the SCTP CMT ns-2 implementation (please refer to Chapter 6 , and specifically Section 6.2.1).

[5] In a real wireless network, typically the transmission range is smaller than the interference (and sensing) range [106].

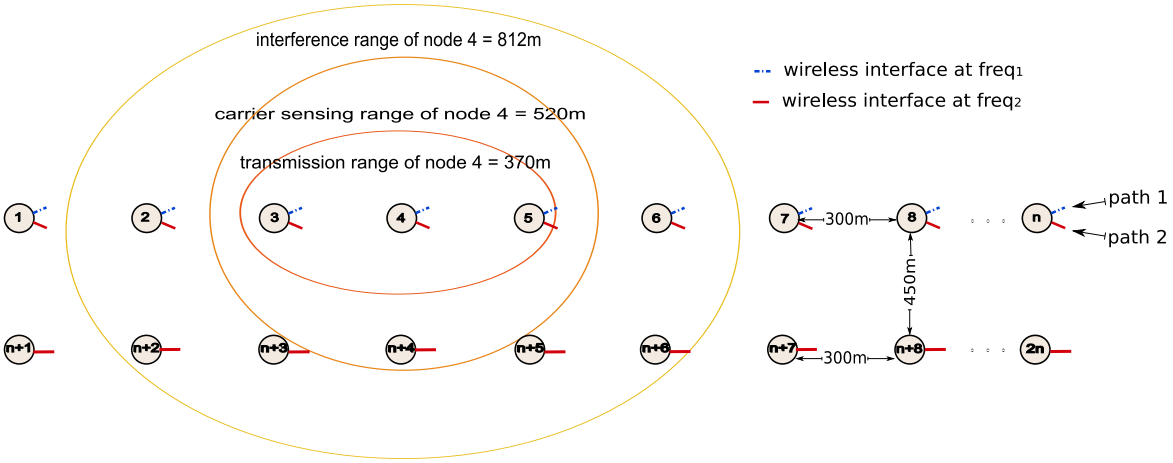


Figure 2.2: Simulation Topology

We then evaluated the performance of CMT in the context of multihop wireless networks using our SCTP CMT QualNet module[6]. We used a chain topology as depicted in Figure 2.2. The nodes in the first chain carry backlogged data traffic via CMT or SCTP associations. The second chain is to carry background traffic.

The first node in the first chain is the data source and the last node in the first chain is the data sink. We vary the number of hops in the chain. Each node in the first chain is equipped with *two* IEEE 802.11b wireless interfaces operating at *different* frequencies ($freq_1$ and $freq_2$) to ensure *two* independent (non-interfering) multihop wireless paths between the source and the destination nodes on the first chain. Adjacent nodes in each chain are located 300 meters away from each other. The transmission range is around 370 meters, the interference range is around 812 meters, and the carrier sensing range is around 520 meters for both of the wireless interfaces using the default values in QualNet version 4.5.1[7]. The data rate for IEEE 802.11b is 2 Mbps and the RTS/CTS mechanism is on. Each SCTP data chunk carries 1000 bytes of application data.

[6] The simulations in this chapter are conducted using svn revision 1 of the CMT implementation in QualNet 4.5.1.

[7] Note that with these settings each node in the chains can communicate only with its direct neighbors, but can interfere with the nodes up to 2 hops away.

The second chain is 450 meters away from the first chain. Each node on the second chain has only *one* wireless interface operating at $freq_2$, with the same wireless properties as the second wireless interface of the nodes in the first chain. The number of nodes in the second chain is the same as the number of nodes in the first chain for each simulation. To create background traffic (i.e., interference) for the CMT subflow running on path 2 of the first chain, we send a CBR (Constant Bit Rate) traffic on the second chain for the entire simulation time. The size of each CBR-PDU is 1000 bytes.

Although Figure 2.2 depicts one "physical" topology of MWNs, the following three aspects of the topology allow us to draw general conclusions. First, the specific chains of nodes represent a multihop topology. Second, the two orthogonal frequencies used along the first chain of nodes represent two independent multihop paths. Third, the traffic on the second chain of nodes may be used to represent the aggregated interference from neighboring nodes of other random topologies.

We used static routing in the simulations to eliminate any complications that might be introduced by the routing on the performance of the transport layer. The size of IP queue is set to be big enough to hold 50 SCTP-PDUs. Simulation time is 420 seconds. We measured the steady-state throughput at the receiving application between the 60th and the 360th seconds of the simulations. Each data point on the graphs is an average of 30 runs with a 95% confidence interval. In the simulations, SCTP flows and CMT with DAC employs delayed SACKs [29] with the delayed ACK factor of 2 (i.e., one T-ACK packet per SCTP-PDU carrying data) and the maximum T-ACK delay of 200 milliseconds.

We compared CMT against three other schemes as introduced in [60].

- **AwareApp**: an application that always picks the SCTP association that uses the *better* path to send data (i.e., one single-homed SCTP association over path 1 of the first chain in Figure 2.2).

- **UnawareApp**: an application that always picks the SCTP association that uses the

worst[8] path to send data (i.e., one single-homed SCTP association over path 2 of the first chain in Figure 2.2).

- **AppStripe**: an "ideal application" that has the best possible performance expected by an application that stripes data perfectly over multiple (independent) paths. Essentially, AppStripe represents the aggregated performance of multiple independent SCTP associations running over different paths. Note that, in our simulations, the throughput of AppStripe is the aggregated throughput of AwareApp and UnawareApp.

We investigated the performance of CMT in two settings: (i) with unconstrained receiver buffer (rBuf) at the transport layer (Section 2.4.1) and (ii) with constrained receiver buffer (Section 2.4.2). Our goal is to shed light on the following questions.

- How does CMT perform in MWNs as compared to AppStripe, AwareApp, and UnawareApp? How is CMT's performance in MWNs different from or similar to the CMT performance over wired networks and why (Section 2.4.1)?
- How influential the receiver buffer (rBuf) blocking problem is on the CMT performance over MWNs? Does rBuf blocking still have a big impact on the CMT performance over MWNs, as it does in the wired network case (Section 2.4.2)?
- How well do the RTX policies of CMT perform in MWNs especially under the constrained rBuf case (Section 2.4.2)?

2.4 Simulation Results and Analysis

In the following sub-sections, we present the simulation results for CMT over MWNs, first with an unconstrained rBuf and then with a constrained rBuf.

[8] Note that, the UnawareApp we defined in this dissertation is different from the UnawareApp introduced in [22], where the latter "sends data to one path selected from a set of available paths with equal probability".

2.4.1 CMT with Unconstrained rBuf

CMT's initial design goal was to obtain application throughput as good as the throughput of AppStripe (i.e., one CMT association is performing as good as the aggregated performance of multiple independent SCTP associations) [63]. However, studies of CMT over wired networks showed that, when the receiver buffer is unconstrained, one CMT flow performs better than the "ideal" data striping application AppStripe [63]. One of the main reasons for the surprisingly better performance of CMT in the *wired* networks as compared to AppStripe is that a CMT flow *shares a single sequence space*[9] across all of the CMT subflows. Therefore, CMT T-ACKs[10] returning from any of the paths to the CMT sender can simultaneously acknowledge data in all of the CMT subflows running over different paths (i.e., one T-ACK can increase the cwnd of all the CMT subflows simultaneously). That is, CMT is more resilient to ACK losses on the reverse paths.

While investigating the performance of CMT over MWNs, our initial hypothesis was that sharing the sequence space might not bring a clear advantage to CMT over AppStripe. We believe that over MWNs, CMT and AppStripe will have *different* spatial channel reuse because of the following two reasons (refer to Figure 2.3 by focusing on the transport layer mechanics, i.e., the shaded rectangle area in the figure).

- *(Hypo-1) Reduced interference between T-DATA and T-ACK packets:* First of all, in CMT, T-ACKs dynamically return from *any* of the paths to the CMT sender. While T-ACKs are returning to the CMT sender from a path, T-ACKs only contend for the channel with the T-DATA packets of the CMT subflow in that path. Whereas, in AppStripe, each SCTP receiver returns T-ACKs to its corresponding SCTP sender. That is, there is always contention between the T-ACKs and the T-DATAs of each

[9] TSN (Transmission Sequence Number) space used for congestion control and reliability.

[10] T-ACKs stands for the ACK packets sent at the transport layer.

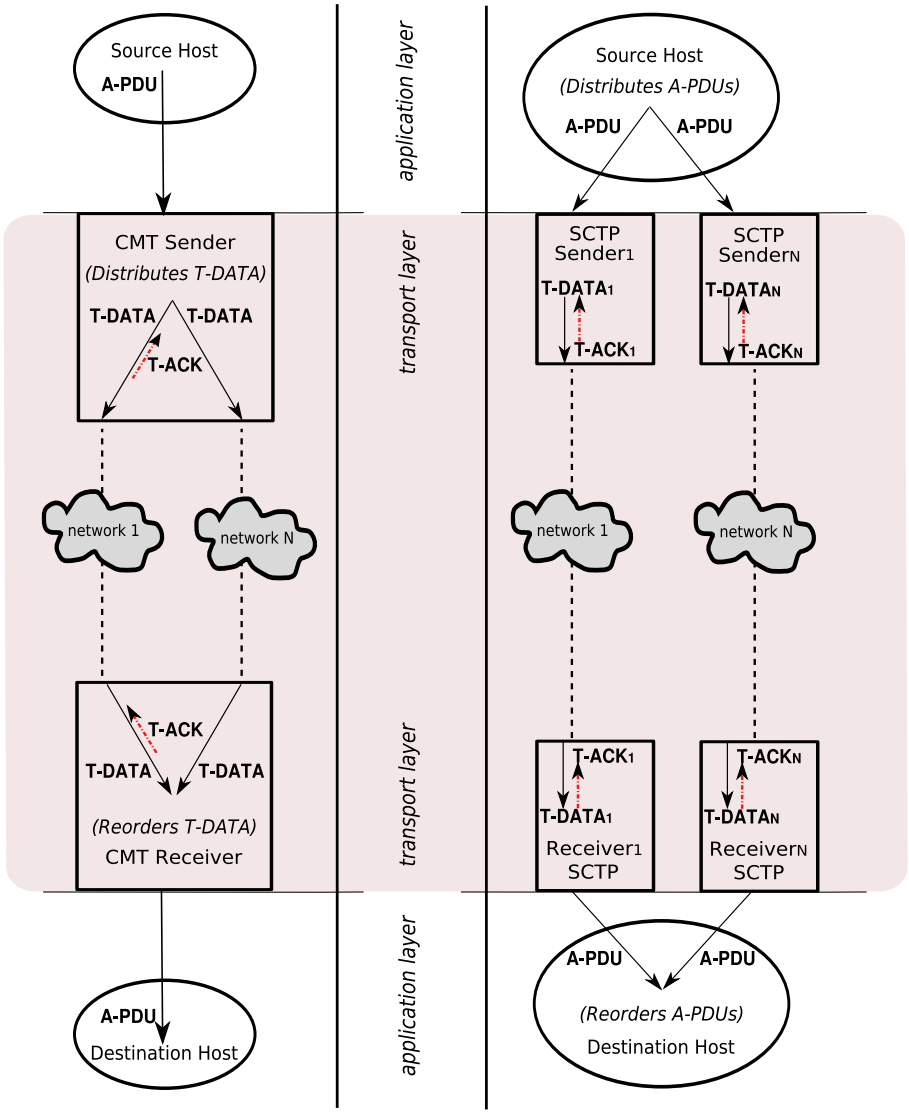
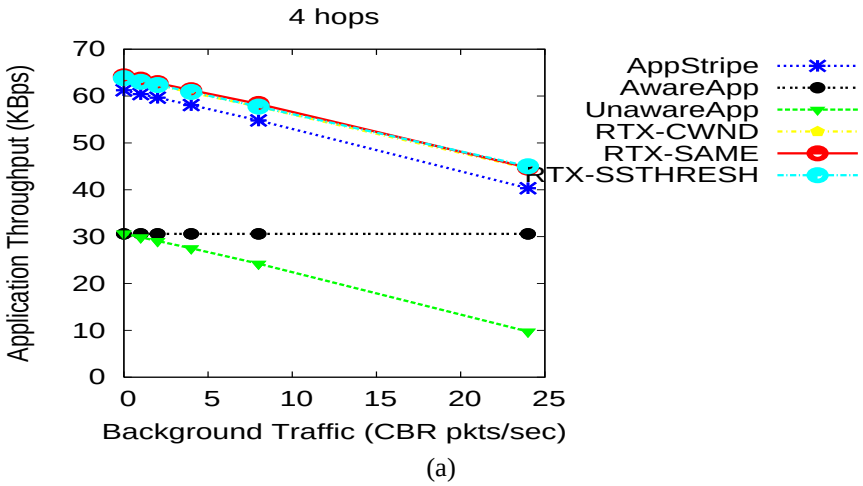
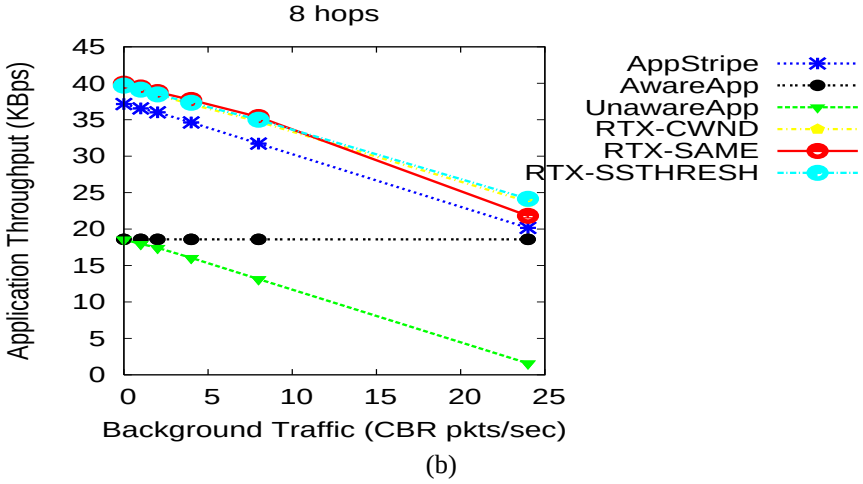


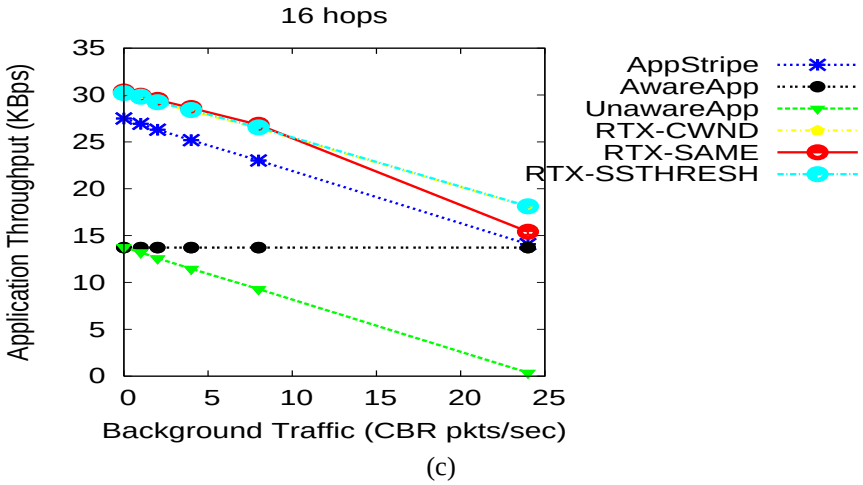
Figure 2.3: **CMT on the left**: T-ACKs are sent to the CMT sender via *any* of the return paths and acknowledge the data in *all* of the CMT subflows (i.e., one T-ACK can increase the cwnd of all of the CMT subflows simultaneously). **App-Stripe on the right**: T-ACKs are *per* SCTP association and acknowledge only the data of the corresponding SCTP association.



(a)



(b)



(c)

Figure 2.4: CMT vs. AppStripe, AwareApp, and UnawareApp with unconstrained rBuf

independent SCTP flow. Therefore, when we consider the channel contention between the T-DATA and the T-ACK packets, CMT has *a better spatial channel reuse* across all the paths compared to the aggregated spatial channel reuse of AppStripe subflows running across different paths. This is a clear advantage for CMT.

- *(Hypo-2) Increased self-interference between T-DATA packets:* However, since CMT T-ACKs simultaneously acknowledge multiple CMT subflows, the cwnd of each CMT subflow can grow *more and faster* compared to the cwnd growth of each independent SCTP flow. Cwnd growth reduces the spatial channel reuse (because as cwnd grows, more T-DATA packets are injected into the network and hence more T-DATA packets compete for the channel along the data forwarding path). Cwnd growth can cause performance degradation in TCP when the cwnd of TCP grows beyond the optimal value [59][11]. Therefore, extra increase in cwnd of each CMT subflow might hurt the throughput of each CMT subflow and hence might hurt the overall throughput of CMT compared to AppStripe.

Therefore, *sequence space sharing* can either increase (as explained in *Hypo-1* above) or decrease (as explained in *Hypo-2* above) the spatial channel reuse of CMT compared to AppStripe. Hence, the throughput of an application over CMT could be higher or lower compared to the throughput of AppStripe.

We have evaluated the performance of CMT with RTX-SAME, RTX-CWND, and RTX-SSTHRESH retransmission policies. Simulation results with unconstrained rBuf size for 4-, 8-, and 16-hop topologies are presented in Figure 2.4. We have the following observations.

- The throughput of CMT over MWNs is $\geq$ the throughput of AppStripe (i.e., aggregated throughput of AwareApp + UnawareApp). This is similar to the wired case

[11] A single-homed SCTP also shows *similar* symptoms [107] since SCTP's congestion control mechanics is "similar" to TCP's.

Table 2.1: CMT (with RTX-CWND) vs. AppStripe for unconstrained rBuf

	0 CBR pkts/sec	8 CBR pkts/sec	24 CBR pkts/sec
4-hop	4.20 %	5.31 %	10.55 %
8-hop	6.55 %	9.48 %	17.88 %
16-hop	9.61 %	14.91 %	28.44 %

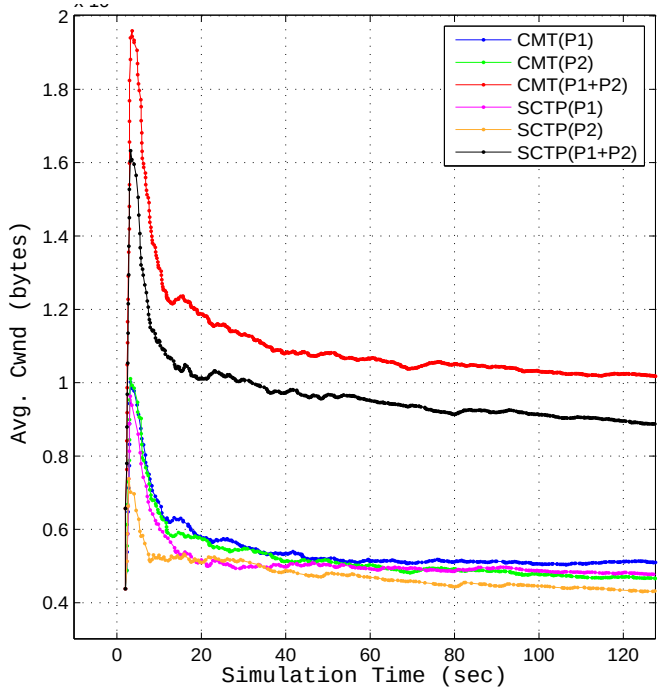


Figure 2.5: Average cwnd growths for the flows

as reported in [60]. Table 2.1 shows how much CMT outperforms AppStripe, as the number of hops and the loss (interference) in path 2 increases. The values in the table were calculated for CMT with the RTX-CWND policy, using Equation 2.1.

$$\frac{(CMT\ throughput - AppStripe\ throughput) * 100}{AppStripe\ throughput} \tag{2.1}$$

- Since we observed that CMT's throughput is $\geq$ AppStripe's in MWNs, we wanted to check *Hypo-2* further by examining several traces to understand how the cwnd's of CMT flow, AppStripe flow, and their subflows grow. Figure 2.5 shows a progression of the average cwnd's under moderate background traffic (8 CBR pkts/sec)

for a 16-hop configuration. What we observe in this figure is that cwnd of CMT subflow 1 grows slightly more (less than one data packet size) than cwnd of the SCTP flow on path 1. In the same way, cwnd of CMT subflow 2 grows slightly more than the cwnd of the SCTP flow on path 2. As we stated in *Hypo-2*, cwnd's of the CMT subflows grow more and faster compared to cwnd of the corresponding AppStripe subflows. However, for our simulation configurations, cwnd growth is not wild enough to hurt the throughput of individual CMT subflows. Hence, the overall cwnd growth of the CMT flow becomes more (almost one data packet size) than the cwnd growth of AppStripe, which leads to higher throughput for CMT.

- As the number of hops increases, the throughput of CMT, AppStripe, AwareApp, and UnawareApp all get smaller. We speculate that the main reason for throughput degradation is that the throughput of an SCTP association[12] decreases as RTT and loss rate of the path increase. Each hop increases RTT. In addition, as the number of hops increases, the simultaneous transmissions on the chain increase, and hence the contention for the channel (and loss rate of the path) increases.

2.4.2 CMT with Constrained rBuf

Next, we investigated the performance of CMT over MWNs with a limited rBuf size. Smaller rBuf sizes can be a performance bottleneck for CMT due to the *rBuf blocking problem*. The rBuf blocking problem of CMT is explained in [60]. A CMT data receiver maintains a single receiver buffer which is shared among CMT subflows. The receiver uses the rBuf (i) to store the application data arriving out-of-order and (ii) to store the data that the application is not ready to consume. To help flow control, a data receiver sends information about available rBuf space to the data sender, using the *arwnd* (advertised receive window) field in the SACK chunks. A data sender then calculates $peerRwnd$ value of the association using (i) the $a\omega rnd$ value in the SACKs and (ii) the

[12] is similar to the throughput of a TCP connection [86].

data that is sent but not yet acked. Data sender uses the $peer\,Rwnd$ to determine how much more data the rBuf at the receiver can hold. The sending rate of path$_i$ (destination address$_i$) at the data sender is then set to $min(cwnd_i, peer\,Rwnd)$[13].

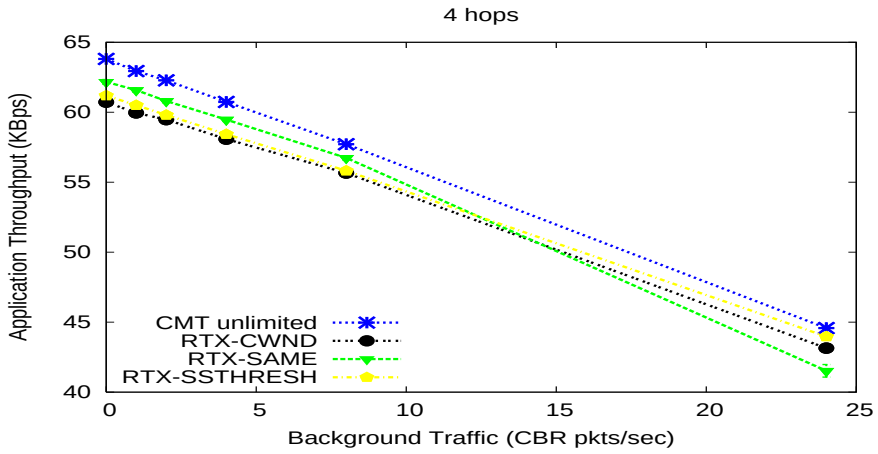
As the receiver keeps data arriving out-of-order, from different paths at the rBuf, the available rBuf space shrinks. While the receiver is waiting for missing data to come, out-of-order data can not be delivered to the application. In the meantime, the CMT sender calculates the $peer\,Rwnd$ to be very small or zero. This means the sending rate of *any* CMT subflow becomes very small or zero. Therefore, the data sending rate of the *entire* CMT association drops to zero, preventing CMT from sending data via any of the paths.

The rBuf blocking problem is unavoidable for CMT, especially if the rBuf size is small, or the delay, loss, or bandwidth characteristics of the paths CMT subflows run through differ greatly. We investigated the CMT performance for 128, 64, 32, and 16 KB rBuf sizes under light to heavy background traffic on path 2. The results for 64 KB, 32 KB, and 16 KB rBuf are depicted in Figures 2.6, 2.7, and 2.8, respectively. We observe the following.

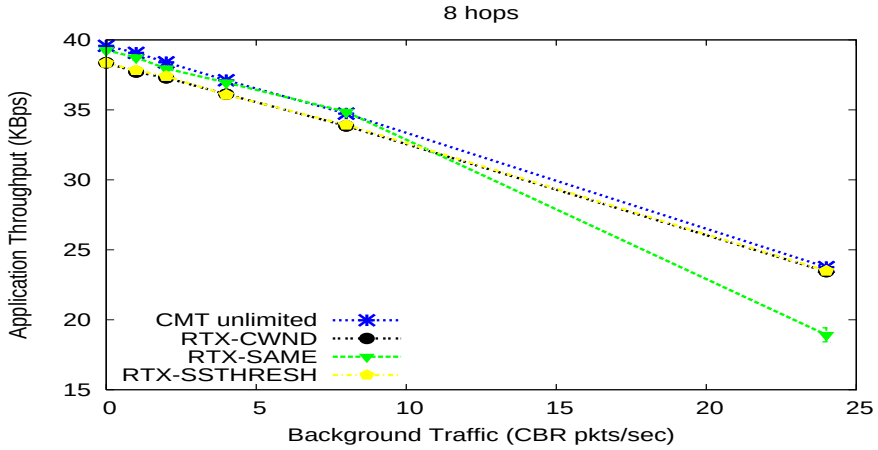
- As the rBuf size gets smaller, rBuf becomes a larger limiting factor in the overall CMT performance (comparing 64 KB vs. 16 KB configurations). In addition, it seems CMT is especially sensitive to the shorter hop (RTT) configurations (comparing 4 hop vs. 16 hop configurations). We did not see any deterioration in CMT throughput for the 128 KB rBuf except for the RTX-SAME under heavy background traffic (results are not presented here). Therefore, for the configurations studied in this dissertation, 128 KB seems to be a proper value for a rBuf size.

- Another comment is about the performance of the RTX policies of CMT over MWNs. The selection of a RTX policy is particularly important for the constrained

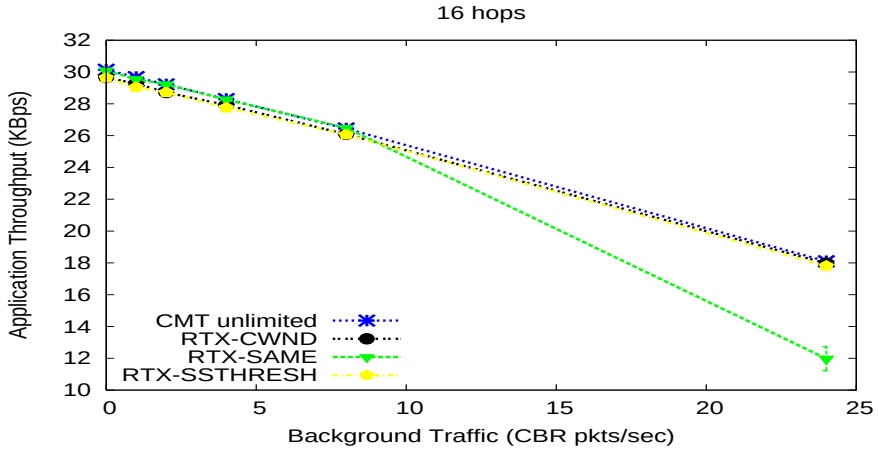
[13] An SCTP CMT sender maintains cwnd *per* destination address.



(a)

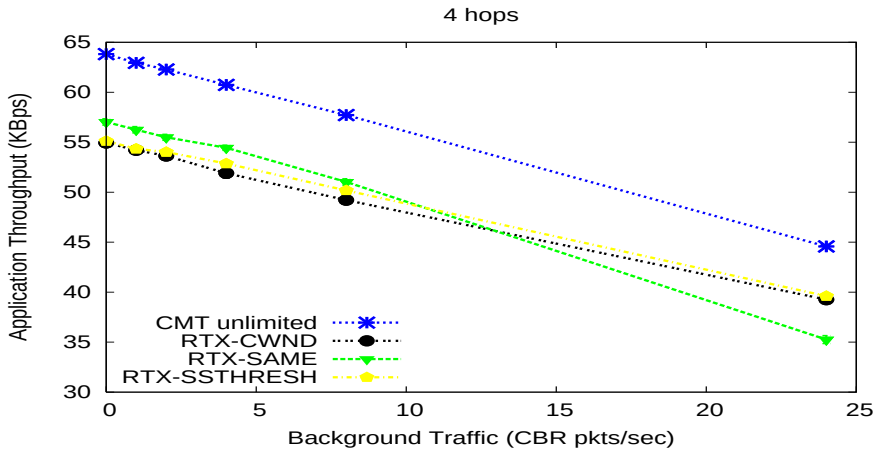


(b)

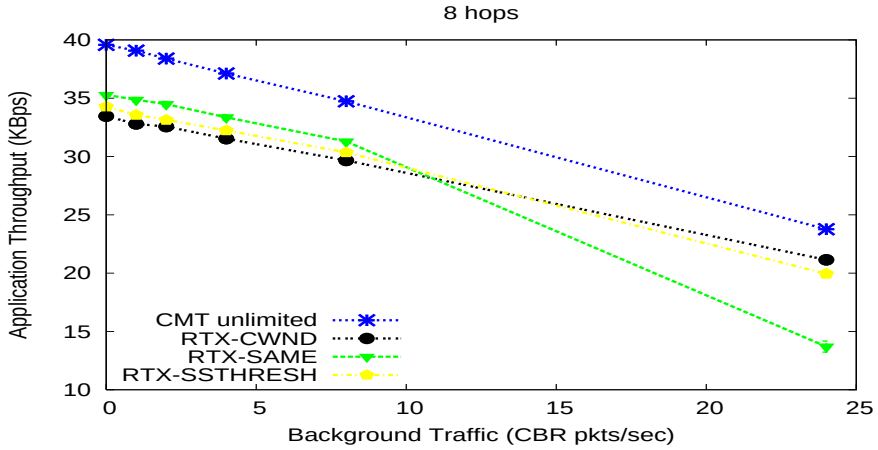


(c)

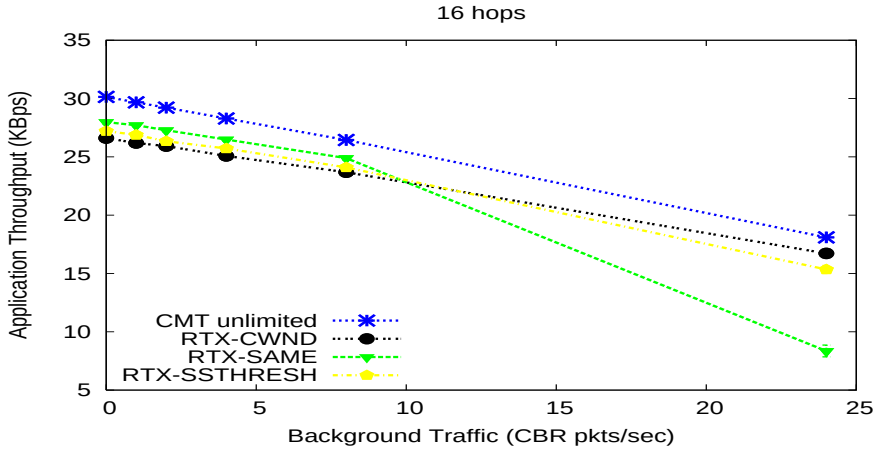
Figure 2.6: CMT with 64 KB rBuf vs. CMT(RTX-CWND) with unconstrained rBuf



(a)

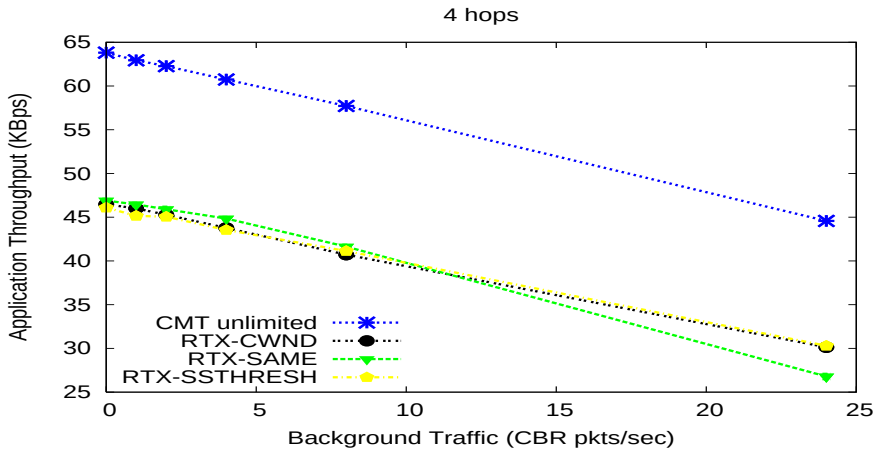


(b)

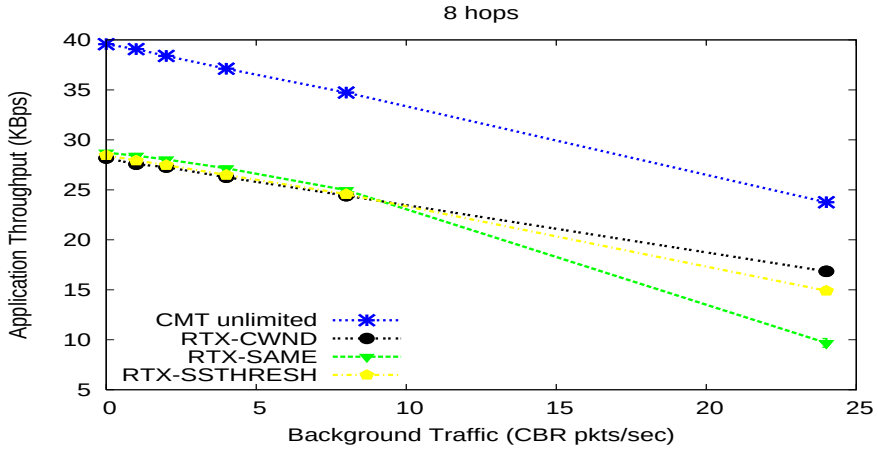


(c)

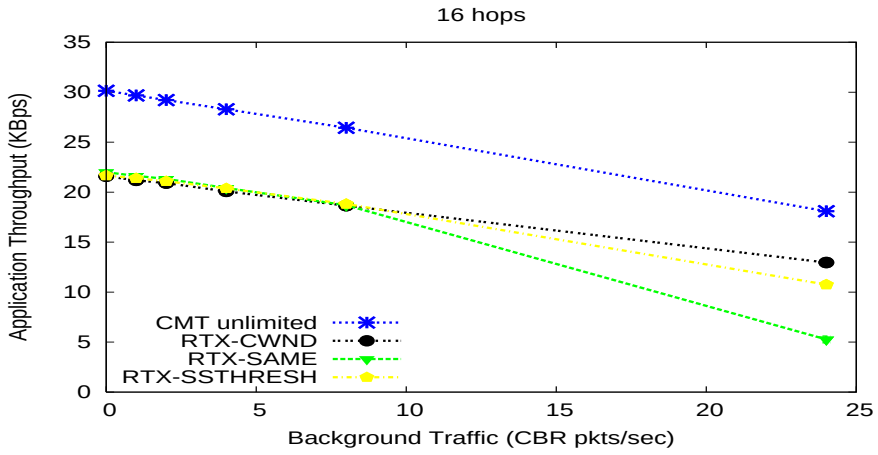
Figure 2.7: CMT with 32 KB rBuf vs. CMT(RTX-CWND) with unconstrained rBuf



(a)



(b)



(c)

Figure 2.8: CMT with 16 KB rBuf vs. CMT(RTX-CWND) with unconstrained rBuf

rBuf cases. This is because making sure that the retransmissions reach the receiver as early as possible and with minimal loss increases data delivery rate to the receiver application, which in turn empties the rBuf at the receiver faster. Iyengar et. al. studied the impact of RTX policies and rBuf blocking in [62, 61]. They concluded that rBuf blocking is unavoidable for CMT but the rBuf blocking problem can be mitigated with the selection of a proper RTX policy. They showed that CMT benefits more from *loss based RTX policies* (such as RTX-CWND and RTX-SSTHRESH) than the RTX-SAME policy. Basically, with a loss based RTX policy, retransmission of a data chunk is sent to the lowest loss path among all of the available paths. They suggested using cwnd (and ssthresh) of a path to approximate the loss rate of the path in the wired networks. That is, with the RTX-CWND (or RTX-SSTHRESH) policy, a retransmission of a data chunk is sent to the path with the highest cwnd (or ssthresh) value. We observed that for light to medium background traffic (0-8 CBR pkts/sec on path 2), RTX-CWND (or RTX-SSTHRESH) shows similar or slightly worse performance than RTX-SAME. However, under heavy background traffic (24 CBR pkts/sec), RTX-SAME is clearly worse than RTX-CWND (or RTX-SSTHRESH) especially as hop count (RTT) increases[14]. In addition, we observed that under heavy background traffic RTX-CWND is slightly better than RTX-SSTHRESH for longer hops. We speculate that this is because cwnd is a faster moving value compared to ssthresh and hence can keep up with the channel condition better.

2.5 Related Work

Understanding TCP and (single-homed) SCTP performance over MWNs is important to understand the performance of SCTP-based CMT in MWNs. The performance of TCP in IEEE 802.11 based multihop wireless networks has been studied extensively

[14] Similar to the reports in [61] for CMT over wired networks.

[59, 106, 36, 70]. In their seminal paper, Fu et. al. [59] discussed location-dependent contention and spatial channel reuse. The paper shows that the main reason for the losses TCP experiences in an IEEE 802.11 DCF based multihop wireless network is the contention-induced losses at the link-layer rather than the buffer overflow at the intermediate routers. They studied the TCP throughput both with ns-2 simulations and real network experiments, and showed that TCP's throughput is the highest when TCP operates at a specific window size that allows the best spatial channel reuse. However, cwnd of a TCP connection typically grows beyond this window size which in turn reduces the spatial channel reuse and hence causes the TCP connection to experience a below-optimal performance. They showed that for a simple chain topology the optimal value for the cwnd of a TCP flow to achieve the highest TCP throughput is h/4 (h is the number of hops) in a string topology. Kawadia et. al. [70] studied the performance of TCP in a real indoor experimental testbed with off-the-shelf IEEE 802.11b cards, and presented results similar to [59]. As TCP and single-homed SCTP use "similar" congestion control and flow control mechamisms, very few papers reported the performance of a single-homed SCTP over multihop wireless networks [107].

In addition to the SCTP-based CMT studied in this dissertation, there are other proposals aiming to exploit multihoming of SCTP to allow transmission of data through multiple paths simultaneously. IPCC-SCTP [108], LS-SCTP [44], SBPP-SCTP and Westwood SCTP [33, 88], WISE-SCTP [58], and cmpSCTP [75] are some of these efforts. To our best knowledge, none of these SCTP-based proposals or any other multihome-capable transport protocol is studied in the context of *multihop* wireless networks.

2.6 Summary and Conclusions

In this chapter, we studied the performance of SCTP-based CMT over IEEE 802.11 based MWNs with two configurations: without limiting the rBuf (receiver buffer) size and with limited rBuf. We compared our results with those of CMT over wired networks [60] to get insights on how influential the characteristics of multihop wireless networks (e.g.:

performance. We found that similar to the wired cases, CMT over MWNs shows better performance than a single SCTP association and even the ideal AppStripe application, when the rBuf is not constrained. For the cases of constrained rBuf, we showed that considering the bandwidth limitations of multihop wireless networks compared to their wired counterparts, rBuf sizes $\geq$ 128 KB can be sufficiently enough not to restrain the CMT performance. In addition, loss-based RTX policies such as RTX-CWND and RTX-SSTHRESH were successfully mitigating the effects of rBuf blocking compared to the RTX-SAME policy. Overall, we conclude that applications will benefit from using CMT in the MWNs context when they have sufficiently large rBuf. contention-induced losses, hidden terminals, limited bandwidth) are on the CMT

Chapter 3

RECONSIDERING ACK POLICY OF CMT OVER MWNS

In the previous chapter, we studied the performance of CMT in the context of multihop wireless networks (MWNs). In this chapter, we specifically focus on the acknowledgment (ACK) policy of CMT over MWNs. The organization of this chapter is as follows. Sections 3.1 and 3.2 present the background and the problem statement, respectively. Sections 3.3 and 3.4 describes our simulation design and results. Section 3.5 discusses other related work, and Section 3.6 summarizes our contributions.

3.1 Background

As mentioned in Section 2.2, IEEE 802.11 DCF is the de facto standard used in the (multihop) wireless networks to mediate access to the medium (channel). Although IEEE 802.11 employs both physical and virtual carrier sensing, and RTS/CTS based channel reservations to coordinate nodes' access to the channel, IEEE 802.11 can not completely eliminate the interference among nodes, the hidden terminal problem, and contention-induced losses. Contention-induced losses in an MWN basically result from the network topology and the flow pattern, and the TCP performance over MWNs is greatly decremented by the contention-induced losses [36, 59, 70, 106]. In MWNs, T-DATA[1] and T-ACK[2] packets of the same TCP connection self-compete for the same wireless channel. There are two aspects to this self-contention within a TCP connection.

[1] T-DATA is a T-PDU carrying data at the transport layer.

[2] T-ACK is a T-PDU carrying transport layer acknowledgments (ACK).

- *Intra-flow interference:* The intra-flow interference occurs when T-DATA packets (flow) of a TCP connection traveling in the forward direction from the source node to the destination node self-interfere among one another. Similarly, intra-flow interference exists among the T-ACKs traveling in the reverse direction from the destination node to the source node.

- *Inter-flow interference:* A TCP sender has a single destination address to send its T-DATA packets. Assuming that single path routing is used, there is a single forward path from the source node to the destination node to send T-DATA packets. Similarly, a TCP receiver will have a single reverse path to send T-ACK packets. When the forward and the reverse paths are the same or overlap[3], inter-flow interference occurs between the T-DATA flow and the T-ACK flow of the same TCP connection, traveling in opposite directions.

Inter-flow interference of TCP vs. CMT: In TCP, application data is sent as a single T-DATA flow from a source node to a destination node. Whereas, in SCTP CMT, application data is sent as simultaneous, multiple T-DATA flows from a source node to a destination node. With multipath routing[4], multiple forward paths may be created from a source node to a destination node to send multiple T-DATA flows. A CMT sender transmits T-DATA packets to all of the forward paths simultaneously, as much as the cwnd of each destination address allows. A CMT receiver on the other hand sends a T-ACK packet[5] to whichever path the *latest* T-DATA packet arrived from. For instance, if

[3] Well-known, single path, wireless routing protocols such as AODV [87] and DSR [68] produce the "same" or overlapping forward and reverse paths between two nodes.

[4] Multipath routing protocols such as [73, 78] can create multiple, disjoint paths between a source and destination pair.

[5] Note that CMT *shares* a single transmission sequence number (TSN) space across all the CMT sub-flows. Therefore, a T-ACK packet for a CMT T-DATA flow can acknowledge all the T-DATA packets sent by any CMT T-DATA sub-flow via any one of the forward paths.

the last T-DATA packet arrived to the CMT receiver from path 1, then a T-ACK packet is returned via path 1 to the CMT sender (see left-side of Figure 2.3). Therefore, a CMT T-ACK flow may interfere with *any* of the CMT T-DATA sub-flows (i.e., there exist an inter-flow interference between a single T-ACK flow and multiple T-DATA sub-flows in CMT).

3.2 Problem Statement

We reconsider the ACK policy of CMT in the context of MWNs. In particular, we focus on reducing the inter-flow interference (between the T-DATA flows and the T-ACK flow) of a CMT association. One observation is that, we can take advantage of availability of multiple paths in a CMT association to reduce the inter-flow interference within a CMT association.

As described above, the policy in the original CMT protocol is to return T-ACKs via *any* one of the paths to the CMT sender. Therefore, the CMT T-ACK flow may interfere with any one of the CMT T-DATA flows. An alternative policy would select *only one* of the return paths for transmitting T-ACKs during the lifetime of an association. With the alternative policy, the T-ACK flow would only interfere with the T-DATA sub-flow in that particular path.

We designed QualNet simulation scenarios to evaluate pros and cons of the two policies. Our goal is to understand which policy is more beneficial and under what conditions to reduce *inter-flow interference* within a CMT association. The following sections describe our simulation design, results, and analysis.

3.3 Simulation Design

We used QualNet 4.5.1[6] for simulations similar to what is presented in Section 2.3. For each simulation design, we have a chain topology where each node is equipped with

[6] We used svn revision 1 and 2 of the SCTP module in QualNet 4.5.1 for the simulations in this chapter.

either one, two, or three IEEE 802.11b interfaces (radios) and radios equipped on a common node operate at orthogonal frequencies. We set up static routes to facilitate routing between the radios operating at the same frequency. In this way, depending on the number of radios per node, we created one, two, or three disjoint (non-interfering) end-to-end paths between a source node and a destination node. We use default[7] wireless interface parameter values in QualNet 4.5.1 which result in the transmission range around 370 meters, the interference range around 812 meters, and the carrier sensing range around 520 meters for all the wireless interfaces. We set the distance between two neighboring nodes in a chain to be 300 meters to allow communications between the direct neighbors. The data rate for IEEE 802.11b is 2 Mbps and the RTS/CTS mechanism is on.

The first node in the chain is the data source while the last node is the data sink. The source node sends data chunks of size 1000 bytes to the sink for the entire simulation duration, continuously. The sending rate at the source node is not constrained by the receiving application's data consumption rate or by the receiver buffer size.

We also have a second chain of nodes used to create background traffic for some of the end-to-end paths in the first chain. We used Constant Bit Rate (CBR) traffic as the background traffic. The CBR chain is 450 meters aways from first chain. Each node in the CBR chain has one IEEE 802.11b interface with the same properties (and the same operational channel) as the 2nd or the 3rd wireless interface of the first chain. In this configuration, the background traffic reduces the quality of the channel for the T-DATA and the T-ACK flows running on path 2 or 3 of the first chain. CBR data packet size is also 1000 bytes.

The IP queues are configured to hold up to 50 SCTP-PDUs of application data. The simulation time is 420 seconds. We measured the steady-state throughput at the receiving application between the 60th and 360th seconds of the simulations. Each data

[7] The default wireless interface parameters in QualNet are based on the off-the-shelf wireless cards (such as Compaq WL110, Air Lancer MC-11b, Siemens SpeedStream SS1021, OrinocoLucent WaveLAN, and Microsoft MN-520).

point on the graphs is an average of 30 runs with a 95% confidence interval. In the simulations, SCTP and CMT use delayed SACKs [29] with the delayed ACK factor of 2 (i.e., one T-SACK packet for every other SCTP data packet) and the maximum T-ACK delay of 200 milliseconds.

We compared the application throughput of CMT against two other transport schemes, single-homed SCTP and multi-homed SCTP. We simulated one, two, and three-path configurations in 4-, 8-, and 16-hop chain topologies. The topology and data flows in one- and two-path schemes are depicted in Figure 3.1 while those of three-path schemes are shown in Figure 3.2. To name each scheme, h is for *homed*, a is for *acknowledgment*, d is for *data*, and P is for *path*. We have the following one-path schemes.

- **1hSCTP@P1**: One single-homed SCTP association on path 1 as in Figure 3.1(a). The T-DATA and the T-ACK flows of the SCTP association run on the same path 1. Therefore, the inter-flow interference between the T-DATA and the T-ACK flows is maximum in this scheme.

- **1hSCTP@P2**: One single-homed SCTP association running on path 2 as in Figure 3.1(b). This is similar to 1hSCTP@P1, but there can also be CBR traffic on path 2 to worsen the channel condition of path 2.

The two-path schemes are the following.

- **2hSCTP-dP1-aP2**: One two-homed SCTP association as in the SCTP standard [98]. However, data (including retransmissions) is only transmitted via path 1, while T-ACKs are only returned via path 2 – Figure 3.1(c).

- **2hSCTP-dP2-aP1**: One two-homed SCTP association similar to 2hSCTP-dP1-aP2. However, a T-DATA flow is on path 2 and a T-ACK flow is on path 1 – Figure 3.1(d).

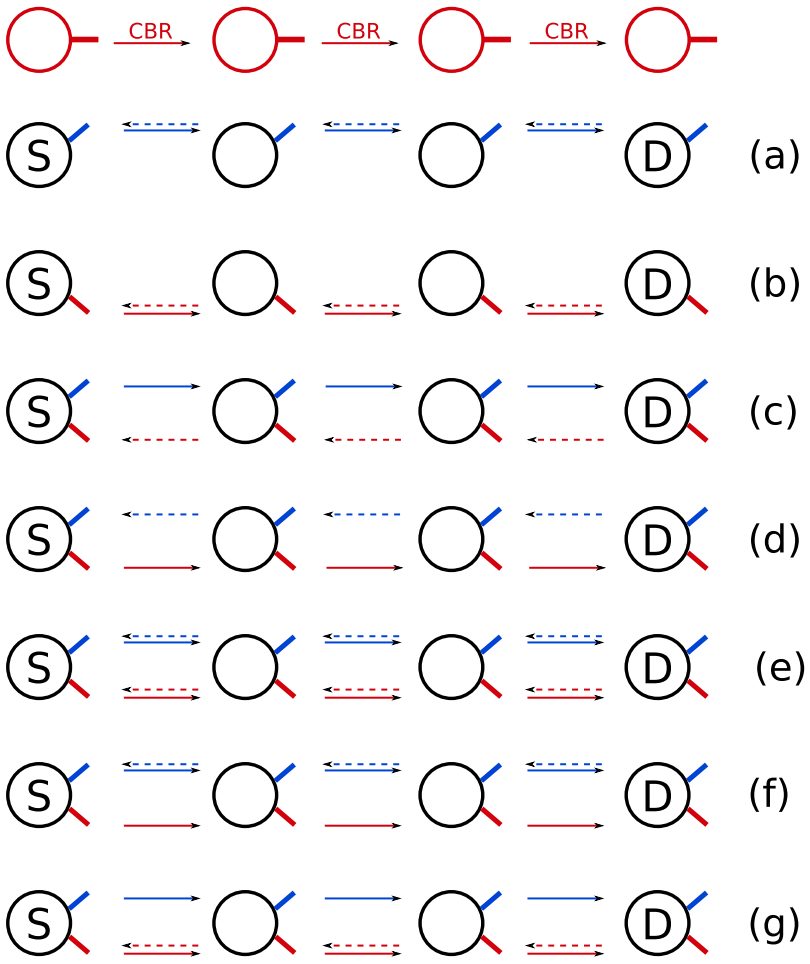


Figure 3.1: *One and two-path schemes*. Each wireless interface operating at a specific channel and each path is depicted with a different color. Solid-arrow lines are T-DATA packets, and dashed arrow lines are T-ACK packets (a) 1hSCTP@P1 (b) 1hSCTP@P2 (c) 2hSCTP-dP1-aP2 (d) 2hCMT@P1P2 (this is the original CMT where T-ACKs can return from *any* one of the paths) (e) 2hCMT-dP1P2-aP1 (f) 2hCMT-dP1P2-aP2

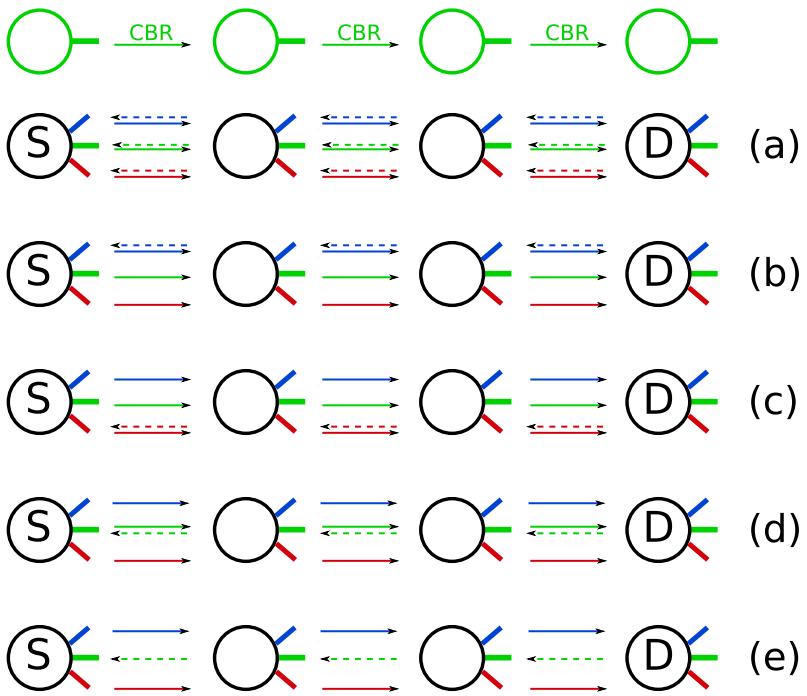


Figure 3.2: *Three-path CMT schemes*. (a) 3hCMT@P1P2P3 (b) 3hCMT-dP1P2P3-aP1 (c) 3hCMT-dP1P2P3-aP2 (d) 3hCMT-dP1P2P3-aP3 (e) 3hCMT-dP1P2-aP3

- **2hCMT@P1P2**: One two-homed CMT association with two simultaneous T-DATA flows on paths 1 and 2, respectively. T-ACKs can be sent to any one of the paths 1 and 2 – Figure 3.1(e). This is essentially the original CMT scheme as studied in [60]. Therefore, the T-ACK flow of the CMT association may interfere with any one of the T-DATA flow on paths 1 and 2.

- **2hCMT-dP1P2-aP1**: One two-homed CMT association with two T-DATA flows on paths 1 and 2, respectively. However, T-ACKs are only sent to path 1 – Figure 3.1(f). This way, the T-ACK flow may only interfere with the T-DATA flow on path 1.

- **2hCMT-dP1P2-aP2**: Similar to 2hCMT-dP1P2-aP1, except that the T-ACK flow may only interfere with the T-DATA flow on path 2 – Figure 3.1(g).

We also simulated three-homed CMT schemes similar to the two-homed CMT schemes, as mentioned above. **3hCMT@P1P2P3** is the original CMT with three T-DATA sub-flows on three disjoint paths. In **3hCMT-dP1P2P3-aP1**, **3hCMT-dP1P2P3-aP2**, and **3hCMT-dP1P2P3-aP3**, there are three data sub-flows on three different paths but only one return path is selected to send T-ACKs. Finally, in **3hCMT-dP1P2-aP3**, there are only two T-DATA flows on paths 1 and 2, respectively while path 3 is completely dedicated to the T-ACK flow (Figure 3.2).

We evaluated these schemes in two setups, (i) without background traffic where we completely look into the self-interference relationships between the T-DATA and the T-ACK flows within a scheme, and (ii) with background traffic, where the path quality of the T-DATA sub-flows and the T-ACK flow differs.

3.4 Results and Analysis

We first investigated cases where there is no background traffic (Section 3.4.1). Then, we studied the cases with background traffic (Section 3.4.2). We had the following hypotheses before we investigated each case.

- Cases with no background traffic
  - Comparing two schemes with *different number of T-DATA flows*: The scheme with a higher number of simultaneous T-DATA flows should perform better (i.e., achieve higher application throughput). For instance, we expected 2hCMT schemes to perform better than 1hSCTP schemes. Similarly, 2hCMT schemes should perform better than 2hSCTP-dP1-aP2.
  - Comparing two schemes with *the same number of T-DATA flows*: The scheme with less inter-flow interference should perform better (i.e., achieve higher application throughput). For instance, 2hSCTP-dP1-aP2 should perform better than 1hSCTP schemes. Another example is that 2hCMT-dP1P2-aP1 (i.e.,

2hCMT with a single T-ACK path) should perform better than 2hCMT@P1P2 (i.e., the original CMT).

- Cases with background traffic: Sending T-ACK flow of a CMT association to the *better* quality path should increase the performance of CMT.

3.4.1 No Background Traffic

The results without any background traffic are presented in Figure 3.3. As expected, there is no performance difference between 1hSCTP@P1 vs. 1hSCTP@P2, and between 2hCMT-dP1P2-aP1 vs. 2hCMT-dP1P2-aP2, as path 1 and path 2 have the same quality (no background traffic in either path). Similar is true for 3hCMT-dP1P2P3-aP1 vs. 3hCMT-dP1P2P3-aP2 vs. 3hCMT-dP1P2P3-aP3, where paths 1, 2 and 3 have the same quality. Comparing CMT with the single- and the two-homed SCTP, we have the following observations.

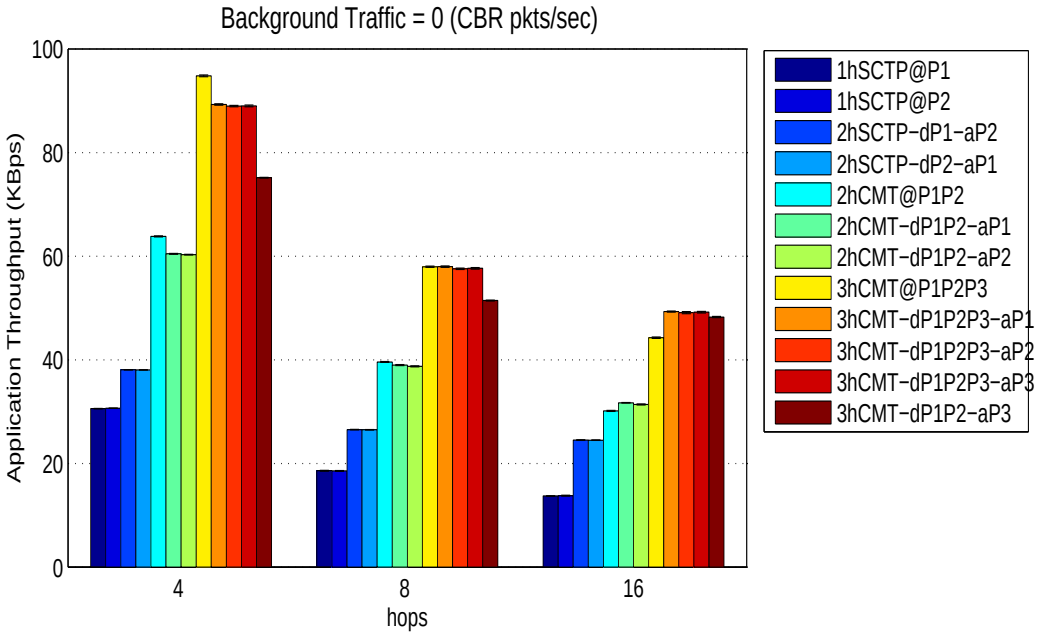
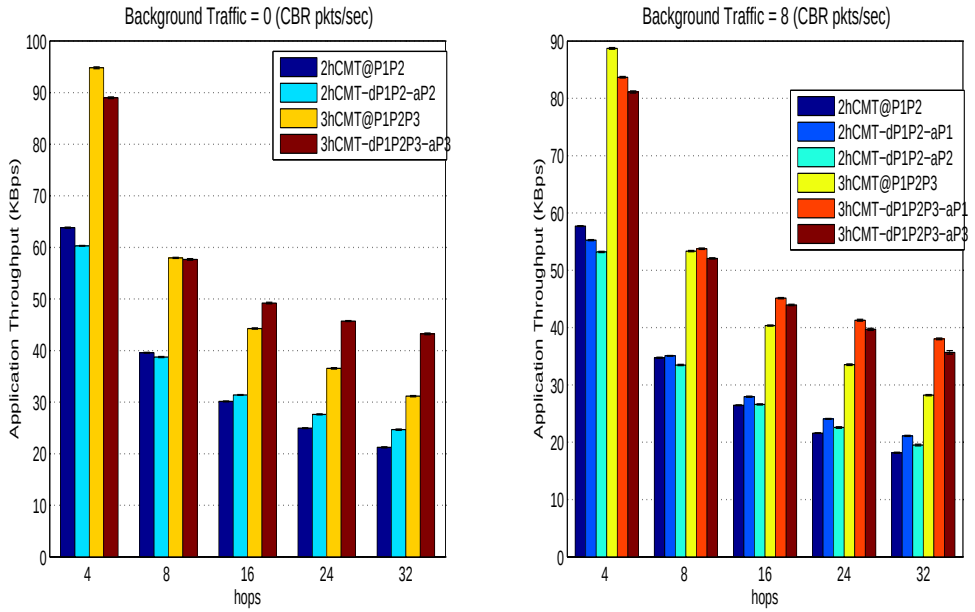


Figure 3.3: Comparing one, two, and three path transport schemes when there is no background traffic.



(a) (b)

Figure 3.4: Comparing CMT schemes where ACKs are sent to only *one* (2hCMT-dP1P2-aP2, 3hCMT-dP1P2P3-aP3) vs. *any* of the paths (2hCMT@P1P2, 3hCMT@P1P2P3) as hop count increases (a) with no background traffic (b) with background traffic on path 2 (for 2hCMT) or path 3 (for 3hCMT).

- *1hSCTP vs. 2hCMT*: The throughput of the two-homed CMT schemes is double or slightly more than double of the performance of the single-homed SCTP schemes on path 1 or path 2. This result is expected, as the design goal of CMT is to have the aggregated performance of multiple, independent, single-homed SCTP associations running on multiple paths [60][8].

- *1hSCTP vs. 2hSCTP vs. 2hCMT*: First of all, throughput of the two-homed SCTP scheme 2hSCTP-dP1-aP2 (or 2hSCTP-dP2-aP1) is better than the throughput of one-homed SCTP scheme 1hSCTP@P1 (or 1hSCTP@P2), as expected. In both one- and two-homed SCTP schemes, there is a single T-DATA flow and a single

[8] Also, refer to our previous work on the performance of CMT over MWNs [26], for further details.

T-ACK flow. However, in the two-homed SCTP schemes, each T-DATA and T-ACK flow has a dedicated path. Therefore, interference between the T-DATA and the T-ACK flows are eliminated. Whereas, in the single-homed SCTP schemes, the same path is used for both T-DATA and T-ACK traffic and hence the interference between the T-DATA and the T-ACK flows is maximum. Secondly, the throughput of two-homed SCTP schemes is lower than the throughput of the two-homed CMT schemes. The main reason is that, in two-homed CMT schemes, the application data is sent into *two* paths simultaneously (i.e., there are two T-DATA flows), while in two-homed SCTP schemes application data is transmitted via only one path.

- *2hCMT with a single ACK path vs. 2hCMT*: Reviewing Figure 3.3, we observe that for the 4-hop chain, 2hCMT@P1P2 has a higher throughput than 2hCMT-dP1P2-aP1 (or 2hCMT-dP1P2-aP2) by about 5.8%. However, as the hop count increases, the throughput of 2hCMT@P1P2 decreases to the values below the throughput of 2hCMT-dP1P2-aP1 (or 2hCMT-dP1P2-aP2). We validated this trend with higher numbers of hops as shown in Figure 3.4. The throughput of 2hCMT-dP1P2-aP2 eventually becomes about 16% higher than the throughput of 2hCMT@P1P2 for a 32-hop chain, when there is no background traffic.

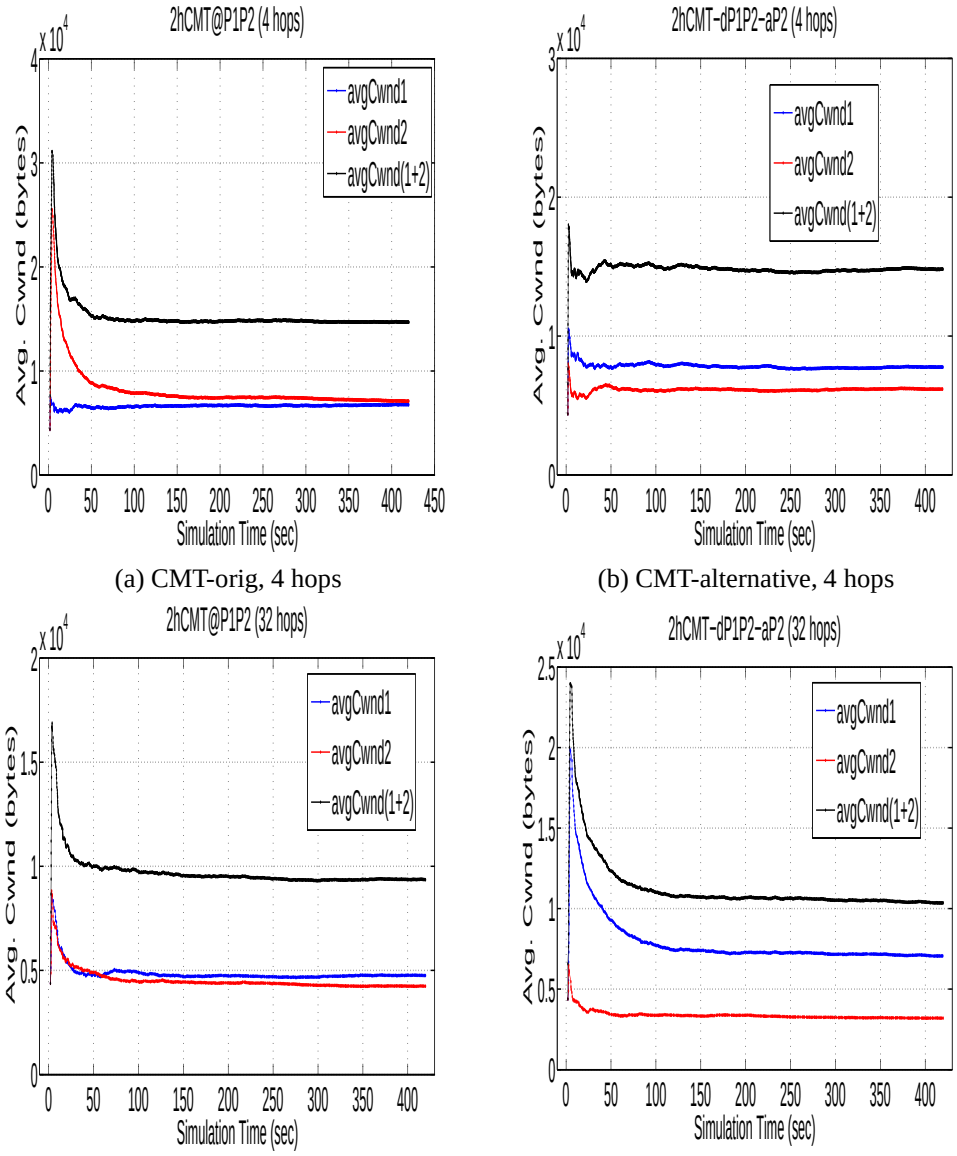
We also compared the two- and the three-homed CMT schemes. The first observation is that, as expected, having three disjoint paths is more beneficial than having two disjoint paths. Additional observations are as follows.

- *2hCMT vs. 3hCMT with two data paths*: The 2hCMT schemes and the 3hCMT-dP1P2-aP3 scheme have two T-DATA flows and a single T-ACK flow. However, in 3hCMT-dP1P2-aP3, the T-ACK flow has a dedicated path and hence does not interfere with any of the T-DATA flows of the CMT association. This is the main reason for the higher throughput of 3hCMT-dP1P2-aP3 compared to the other 2hCMT schemes.

- *3hCMT with a single ACK path vs. 3hCMT*: We observe that the throughout of 3hCMT@P1P2P3 is about 6.5% higher than the throughput of 3hCMT-dP1P2P3-aP1 (3hCMT-dP1P2P3-aP2, or 3hCMT-dP1P2P3-aP3), where the ACK path is fixed as path 1 (path 2, or path 3), for a 4-hop chain. However, as the number of hops increases, the performance of 3hCMT-dP1P2P3-aP1 (3hCMT-dP1P2P3-aP2, or 3hCMT-dP1P2P3-aP3) starts outperforming 3hCMT@P1P2P3. The results with no background traffic and higher hop counts are shown in Figure 3.4(a). This is similar to the case of 2hCMT@P1P2 vs. 2hCMT-dP1P2-aP1 (or 2hCMT-dP1P2-aP2). However, in the three-path 3hCMT case, sending T-ACKs to a single path is even more beneficial as the number of hops increases, compared to two-path 2hCMT. For a 32-hop chain, 3hCMT-dP1P2P3-aP3 has 38.8% more throughput than 3hCMT@P1P2P3.

In layman's term, we conclude that for a CMT association over MWNs, sending the T-ACK flow to a *single* path (i.e., the alternative ACK policy) is better than sending the T-ACK flow to *any* one of the paths (i.e., the original CMT ACK policy), especially (i) as the number of hops increases, and (ii) as the number of simultaneous T-DATA flows within a CMT flow increases. We checked the individual simulation runs to better explain our conclusion. Figures 3.5 and 3.6 show the average congestion window growths of the original CMT and the CMT with the alternative ACK policy for two-homed and three-homed CMT associations, respectively, with short (4-hop) and long (32-hop) paths. Comparing the average congestion windows in the figures, we conclude the following.

- Interflow-interference is the main factor that causes the performance difference between different CMT ACK policies. When the T-ACK flow is sent to only one path (as in the alternative ACK policy), the T-ACK flow only affects the cwnd (and hence the sending rate) of that particular path. On the other hand, when the T-ACK flow is sent to any one of the paths (as in the original CMT case), T-ACK flow interferes



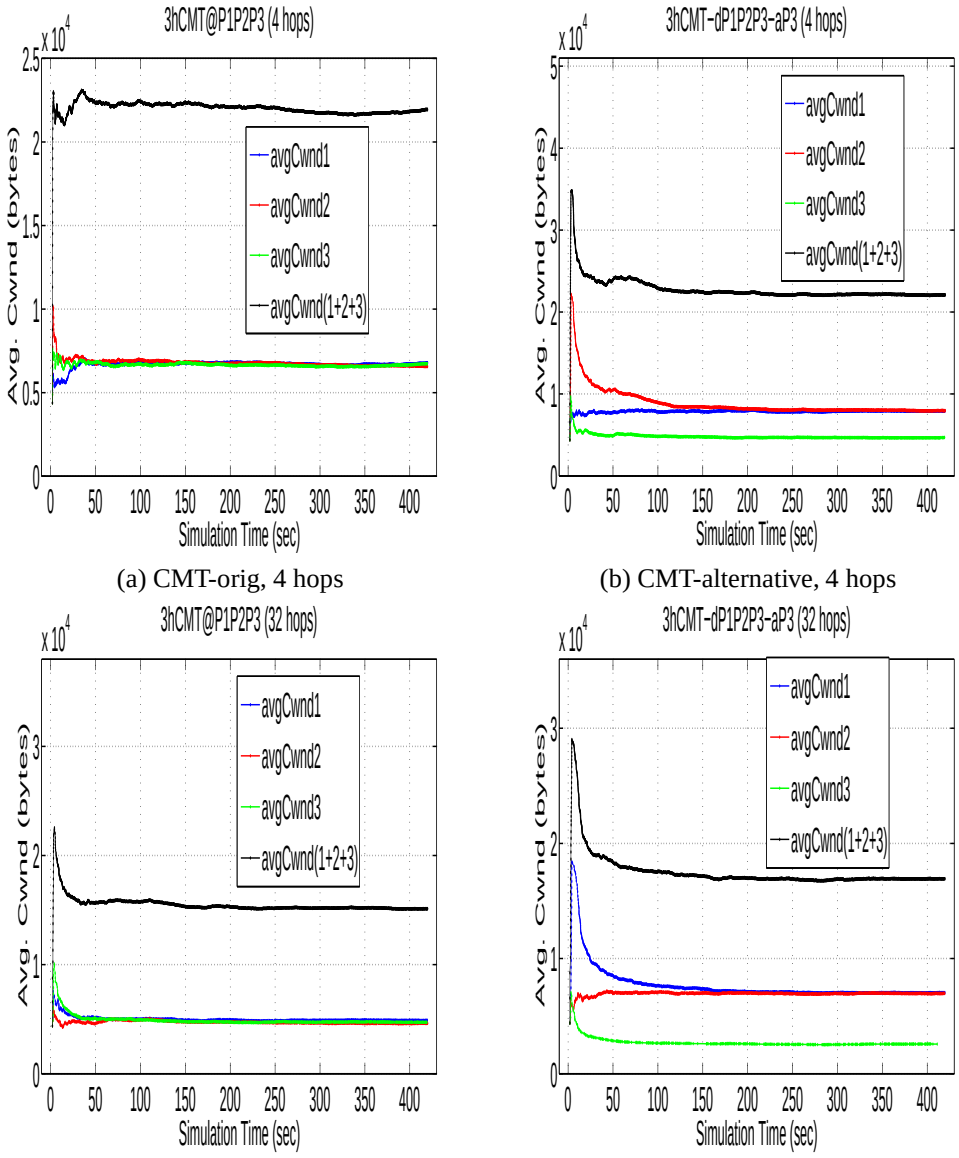
(a) CMT-orig, 4 hops

(b) CMT-alternative, 4 hops

(c) CMT-orig, 32 hops

(d) CMT-alternative, 32 hops

Figure 3.5: Average congestion windows of two-homed original CMT (left-side) and the CMT with alternative ACK policy (right-side) for 4-hop (top) and 32-hop (bottom) paths



(a) CMT-orig, 4 hops

(b) CMT-alternative, 4 hops

(c) CMT-orig, 32 hops

(d) CMT-alternative, 32 hops

Figure 3.6: Average congestion windows of three-homed original CMT (left-side) and the CMT with the alternative ACK policy (right-side) for 4-hop (top) and 32-hop (bottom) paths

with all the T-DATA flows of CMT and reduces the cwnd of all the T-DATA flows, which may collectively decrease the performance of the CMT association.

- Comparing the performance of the original CMT and the CMT with the alternative ACK policy in both shorter and longer paths: Longer multihop wireless paths have higher delay and loss rates. Therefore, when running on longer paths, CMT sub-flows have smaller sending windows[9]. CMT flows with smaller cwnd's are more sensitive to the (interference-induced) losses. Therefore, tampering with fewer number of cwnds within a CMT association (as in the case of CMT with the alternative ACK policy) increases the sending rate.

- Comparing the performance of the original CMT and the CMT with the alternative ACK policy with two or three CMT sub-flows: When the number of T-DATA flows in a CMT association is higher, more T-DATA flows are affected from the inter-flow interference using the original CMT ACK scheme. Therefore, the overall performance of the original CMT degrades.

3.4.2 With Background Traffic

The results with background traffic (8 CBR pkts/sec) in either path 2 (for one- and two-homed schemes) or path 3 (for three-homed schemes) are depicted in Figure 3.7.

- *Single-homed SCTP with background traffic*: Comparing the throughput of 1hSCTP@P1 and 1hSCTP@P2, we validated that having background traffic in the path affects SCTP throughput adversely.

- *Two-homed SCTP with background traffic*: When there is background traffic on path 2, 2hSCTP-dP1-aP2 shows better performance than 2hSCTP-dP2-aP1. This

[9] Similar to the sending rate of a TCP flow, sending rate of a CMT sub-flow is inversely proportional to the RTT and the square root of loss [86].

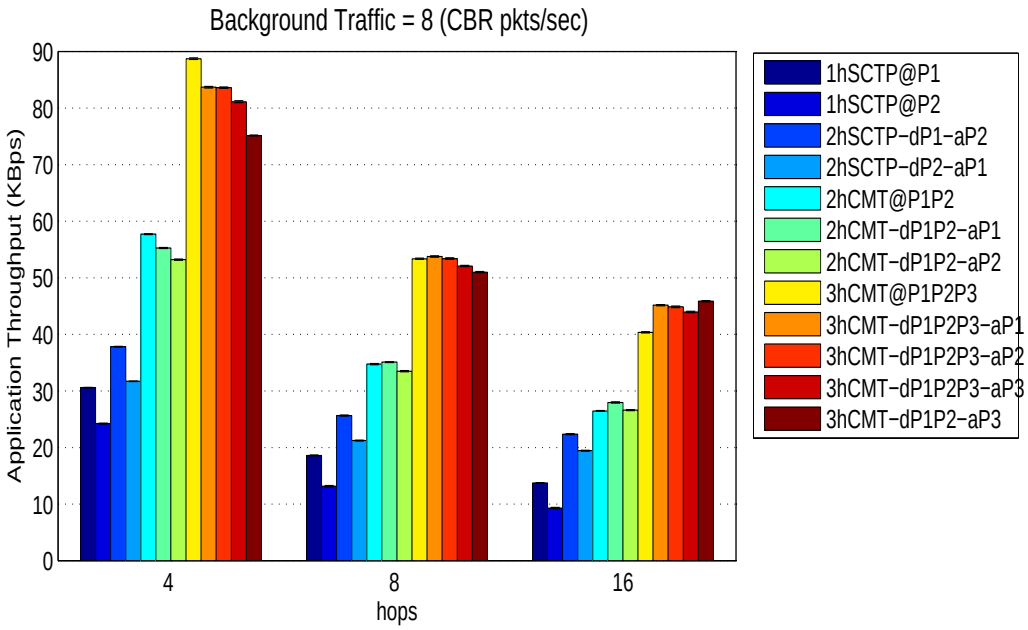


Figure 3.7: Comparing one, two, and three path transport schemes when there is background traffic in path 2 (for 2-homed schemes) or path 3 (for 3-homed schemes).

suggests that a single-flow SCTP association is more sensitive to the data loss than the ack loss.

- *2hCMT and 3hCMT with background traffic*: Here, we studied sending ACKs to the *better* path vs. to *any* of the paths for two- and three-homed CMT associations – Figures 3.7 and 3.4(b). Comparing 2hCMT@P1P2 with 2hCMT-dP1P2-aP1, as the hop count increases, it is more crucial to use the better path for sending ACKs. 2hCMT-dP1P2-aP1 shows 16% better performance than 2hCMT@P1P2 for a 32-hop chain. Similarly, the throughput of 3hCMT-dP1P2P3-aP1 is about 34% better than the throughput of 3hCMT@P1P2P3.

3.5 Related Work

There are a few papers in the literature which studied the *self*-interference between T-DATA and T-ACK flows of a TCP connection in multihop wireless networks. COPAS

[38] is a routing scheme which modified the AODV routing protocol to create two disjoint paths between the sending and the receiving nodes. COPAS transmits TCP DATA through one path from the sender to the receiver, while transmitting the ACK packets through the other path from the receiver back to the sender. Since COPAS operates on a singler-radio, single-channel network, the forward and the backward paths ends up interfering with each other in the vicinity of both the source node and the destination node. [105] also considers transmitting the T-DATA and the T-ACK packets of a TCP connection through disjoint paths between the source node and the destination node. Similar to COPAS, [105] modifies AODV to create multiple disjoint paths between the sender and the receiver. Unlike COPAS though, their channel assignment and routing schemes work with multiple radios per node running on orthogonal channels to reduce the intra-path interference within a single path and inter-path interference between the multiple paths.

Note that both [38] and [105] use multipath routing schemes. However, application data over TCP is transferred as a single T-DATA flow in a single path from the source node to the destination node. In contrast, CMT transmits the application data as multiple T-DATA flows from the source node to the destination node, and works best when the T-DATA paths are disjoint. Another option for an application over TCP is to transfer a single T-DATA flow with multipath routing over multiple paths from the source to the destination. However, multipath routing hurts the TCP performance since application data on different paths may arrive at the same TCP receiver out of order. Out-of-order data arrivals cause the receiver to send duplicate T-ACKs, which result in spurious retransmissions at the TCP sender. On the other hand, CMT mitigate the effects of out-of-order data arrivals over different paths at the CMT receiver. [27] tackled the TCP self-interference at the MAC layer. The paper proposed two MAC layer schemes which modify IEEE 802.11 to reduce the intra-flow interference between the T-DATA (or T-ACK) packets and the inter-flow interference between the T-DATA and the T-ACK packets.

An enumeration of the multi-radio multi-channel multi-path multi-hop (m^4) wireless networks is given in [103, 104]. In m^4 networks, not only inter-path interference but also intra-path interference is tackled by having the radios within a single path operate at orthogonal channels. Of course, how much inter- and intra-path interference reduced depends on how successful the proposed channel switching, channel assignment, and/or routing protocols are [43, 95, 101, 103, 104]. In our CMT route setup, we used multiple radios per node operating at orthogonal channels. In addition, radios within a path operate on the same channel (no channel switching and assignment). Therefore, we created multiple disjoint paths with zero inter-path interference, but the intra-flow interference within each path is maximum.

3.6 Summary and Conclusions

We study CMT over static IEEE 802.11-based multihop wireless networks (MWNs). One observation is that the availability of multiple (return) paths within a single association gives a CMT receiver an opportunity to *select* the transmission path of the returning ACK packets. According to the ACK policy of the original CMT, an ACK packet is sent to the path where the latest DATA packet arrives from. Therefore, a CMT receiver may send ACKs packets to *any* one of the (return) paths. We evaluated an alternative ACK policy which sends ACK packets to the *same* (return) path during the entire association lifetime. We found out that especially as the multihop wireless path gets longer and the number of simultaneous CMT sub-flows increases, the alternative ACK policy shows better performance, even when the quality of the paths differ. This is because the inter-flow interference (between the DATA and the ACK packets) within a CMT flow using the alternative ACK policy is less than the inter-flow interference within a CMT flow with the original ACK policy, which causes to the aggregated sending rate of the CMT flow using the alternative ACK policy to become higher than the aggregated sending rate of the CMT flow using the original CMT ACK policy.

Chapter 4

TCP-FRIENDLINESS OF SINGLE-HOMED SCTP

This chapter investigates TCP-friendliness of single-homed SCTP via QualNet simulations. The organization of the chapter is as follows. Section 4.1 reviews the background and the "formal" definition of TCP-friendliness. Section 4.2 states our motivation to study TCP-friendliness of single-homed SCTP. Section 4.3 elaborates the differences between the protocol specifications of TCP and SCTP as well as the conformance of QualNet TCP and SCTP simulation models with respect to the protocol specifications. Sections 4.4 and 4.5 describe the experimental framework and results and analysis, respectively. Section 4.6 reviews related work, and Section 4.7 concludes the chapter.

4.1 TCP-Friendliness: Background and Definition

In a computer network, *congestion* occurs when the demand (load or traffic the data sources pump into the network) is close to or larger than the network capacity. As a result of congestion, (i) the network throughput in terms of what the traffic sinks receive, decreases even though the load in the network increases, (ii) the packet delay in the network increases (as the router queues become longer), and (iii) packet loss increases (since router queues become full and start dropping packets). When no action is taken to prevent or reduce congestion, the network can be pushed into a state called *congestion collapse*, where little or no useful end-to-end communication occurs.

Congestion collapse was first defined and described as a possible threat for TCP/IP-based networks by Nagle in 1984 [81]. The first congestion collapse of the Internet was

observed in 1986 when data throughput between Lawrence Berkeley Lab to UC Berkeley significantly dropped to pathetic levels [65]. The original TCP specification [89] only included a *flow control* mechanism to prevent a transport sender from overflowing a transport receiver. TCP did not have any mechanism to reduce the (total traffic) load in the network, when network is yielding signs of congestion. In 1988, V. Jacobson et. al. proposed several algorithms (including *slow start* and *congestion avoidance*) based on the *conservation of packets* principle and *AIMD (Additive Increase, Multiplicative Decrease)* mechanisms to address the TCP flaws to prevent congestion collapse [65].

The conservation of packets principle states that "*once the system is in equilibrium (i.e., running stably with full data transit rate), a new packet will not be put into the network unless an old packet leaves the network*" [65]. Jacobson used *ACK clocking* to estimate if an old packet has left the network so that a new packet can be put into the network. TCP's *slow start* algorithm helps TCP come to an equilibrium point (i.e., starting the ACK clocking) quickly by increasing the sending rate of the data source by 1 MSS per received T-ACK[1]. Once the system is in equilibrium, the *congestion avoidance* algorithm takes over. During congestion avoidance, if there is no sign of congestion (i.e., no packet losses), a TCP source increases its sending rate by $(1 \times MSS/cwnd)$ per received ACK[2] (what is called *additive increase*). When there is a sign of congestion though, the TCP source reduces its sending rate to half of the previous sending rate (what is called *multiplicative decrease*). In their seminal paper [37], Chiu and Jain explain that if all the traffic sources in the network obey the AIMD principle, the network will not have congestion collapse and the bandwidth in the network will be "equally" shared among the flows in the network. The TCP's congestion control algorithms developed by Jacobson

[1] That is during slow-start, TCP doubles its sending rate per RTT. Therefore, in contrast to its name, during slow start TCP's congestion window opens up exponentially.

[2] Note that during the congestion avoidance phase, TCP congestion window is incremented a total of 1 MSS per RTT, i.e., a linear increase.

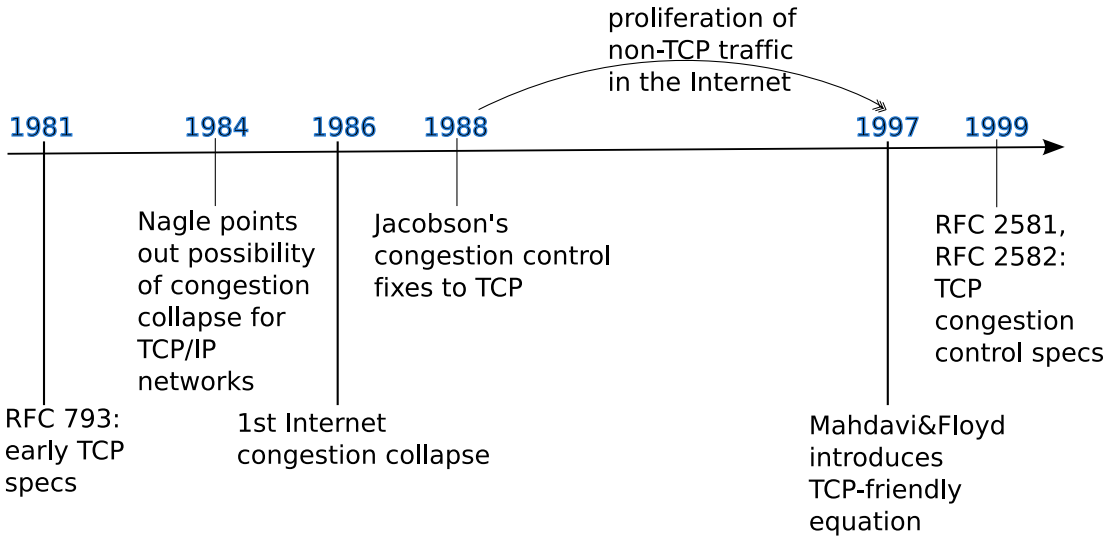


Figure 4.1: History of events that led to the doctrine of TCP-friendliness

were later revised and standardized by IETF as RFCs 2581 [18] and 2582 [51][3].

In 1980s the traffic in the Internet was mostly composed of applications running over TCP. Therefore, the congestion control mechanisms of TCP (as explained above) were sufficient to control the congestion in the Internet. However, as the Internet evolved, non-TCP traffic (such as streaming and multimedia applications running over UDP) began consuming a larger share of the overall Internet bandwidth, competing unfairly with the TCP flows, and essentially threatening the Internet's health and stability. As a response, the notion of *TCP-friendliness* emerged [77].

Definition: The TCP-friendliness doctrine [55] states that *"a non-TCP flow should not consume more resources (bandwidth) than what a conforming TCP flow would consume under the same conditions (segment size, loss, and RTT)"*. In addition, a non-TCP transport protocol should implement some form of congestion control to prevent congestion collapse.

[3] Note that RFC 2582 is obsoleted by RFC 3782 [52] in 2004.

In 1997, Floyd and Mahdavi introduced the *TCP-friendly equation*[4] [77] (Equation 4.1) which roughly calculates the bandwidth consumed by a TCP flow (conforming with the TCP congestion control algorithms). In 1998, Padhye et. al. extended this equation to include timeout events [86]. Figure 4.1, summarizes the chronology of events that led to the doctrine of TCP-friendliness.

$$bandwidth\ consumed = \frac{1.22 * MSS}{RTT * \sqrt{loss}} \tag{4.1}$$

4.2 Motivation

This chapter studies TCP-friendliness of single-homed SCTP, and the experiments conducted in this chapter aim to investigate the basic case where only one *competing* pair of TCP and single-homed SCTP[5] flows exist in the network. As mentioned earlier, one of our main goals in this dissertation is to investigate "TCP-friendliness" of SCTP CMT [63, 60]. Since CMT is based on single-homed SCTP, we believe that the first step in understanding TCP-friendliness of CMT is to understand TCP-friendliness of single-homed SCTP. Therefore, the results in this chapter serve as the first step of the experimental framework in Chapter 5. In addition, there exists little work in the literature about the basics of SCTP vs. TCP in the context of TCP-friendliness. This work also intends to bridge this gap. Furthermore, the comparison work in this chapter can also be considered as a model for the question of *"how to compare two transport protocols (especially from the congestion control perspective)?"*. In this chapter, we consider a topology where a single *tight link* [66] is shared by the flows in the network.

[4] Note that [28] defined another term called *TCP-compatible flow*. However, based on the definition given in the document, TCP-compatible flow is the same as what was earlier defined as TCP-friendly in [77].

[5] In this chapter, unless otherwise stated, SCTP refers to *single-homed* and *single-stream* SCTP associations as in [98].

4.3 SCTP vs. TCP Mechanics

SCTP's congestion control algorithms are designed to be "similar" to TCP's. However, there are subtle differences between the two that can make one transport protocol behave more aggressively than the other under certain circumstances. A few reports provide the similarity and the differences between SCTP and TCP mechanisms [14, 15, 30, 100]. In this section, we highlight *some* of such subtle differences between single-homed SCTP (based on RFC 4960 [98]) and TCP flavors (based on RFCs 2581 [18], 2582 [51], and 2018 [80]) that we believe are directly related to the discussion of TCP-friendliness.

1. **Comparing Transport Protocol Overheads**

   - *Transport PDU headers*[6]– A TCP-PDU has 20 bytes of header (without any options), whereas, an SCTP-PDU has 12 bytes of common header plus (data and/or control) chunk headers. For example, an SCTP data chunk header[7] has 16 bytes. If an SCTP-PDU carries a single data chunk, the total header size will be 28 bytes, which is 40% larger than the header of TCP-PDU (without any options).

   - *Message-based vs. byte-based transmission*– For SCTP, a *chunk* is the basic unit of transmission. SCTP sender wraps each A-PDU in one chunk[8]. SCTP receiver delivers each received A-PDU in the same way to the receiving application. That is, SCTP preserves message (A-PDU) boundaries during transmission and delivery. In contrast, TCP does byte-based transmission. A TCP sender does not maintain message (A-PDU) boundaries, and for example can concatenate the end portion of one A-PDU with the beginning portion

[6] Format of TCP and SCTP PDU's and headers are given in Appendix E.

[7] Refer to Figure E.3 in Appendix E.

[8] Note that, if the size of A-PDU is bigger than MTU, then an SCTP sender fragments the A-PDU into multiple chunks. Then, the SCTP receiver reassembles the A-PDU before delivering it to the receiving application.

of another A-PDU as the bytes fit into one single TCP-segment (TCP-PDU) during transmission. In the same way, a TCP receiver delivers some or all of an A-PDU to the receiving application, with one system call.

The impact of message-based vs. byte-based transmission on the relative performance of SCTP vs. TCP is that, as A-PDU size decreases, the overhead of SCTP per A-PDU will increase compared to TCP[9]. However, for the simulations in this chapter, we try to make the SCTP and TCP to be as similar as possible for the sake of TCP-friendliness discussion. Therefore, we do not consider the impact of A-PDU size[10].

- *Transport protocol ACKs*– SACK[11] is a built-in feature in SCTP while TCP needs to use *SACK option* [80]. SCTP defines a special chunk called *SACK chunk*[12] as the way to report the gaps in the received data. There are two issues with SCTP's SACK chunk compared to TCP's SACK option. (i) SACK chunk has a relatively large size compared to TCP SACK option. The size of a SACK chunk header without any *gap ACK blocks* is 12 bytes. For example, if there is no gap observed in the received data, SCTP-PDU carrying only a single SACK chunk will be 24 bytes[13]. On the other hand, TCP-PDU carrying no data (and options) will be 20 bytes. (ii) TCP SACK option can have at most 4 SACK blocks[14] (limiting the size of TCP header to 60 bytes in total),

[9] Assuming that SCTP does no bundling, and application over TCP connection does not use PUSH flag in TCP header.

[10] Interested readers can look into [69] for the impact of A-PDU size on SCTP vs. TCP throughput.

[11] In addition to the cumulative ACK, transport receiver also selectively sends other missing TSNs information.

[12] See Figure E.4 for the format of SCTP SACK chunk.

[13] 12 bytes for common header, 12 bytes for SACK chunk.

[14] Note that, when the TCP PDU also carries a TCP timestamp option, the limit of SACK blocks within a TCP SACK option becomes 3. Time stamp option is activated

while SCTP SACK chunk can have larger number of gap ACK blocks, as long as the size of the SCTP-PDU is smaller than Path MTU. Hence, as the path loss (especially in a high bandwidth path) gets higher, SCTP SACK can better inform the SCTP sender about the missing data compared to TCP, at the expense of increased overhead.

- In addition to the differences of protocol overhead between the basic SCTP and TCP specifications, as mentioned above, we note that QualNet 4.5.1 implements RFC 1323 [64] for high performance TCP. Therefore, the *TCP window scaling* option[15] is implemented together with the *TCP timestamps* option, which adds 12 extra[16] bytes to the TCP header of every TCP-PDU, making the TCP header 32 bytes.

2. **Comparing Congestion Control Mechanisms**

   - *How to increase cwnd*: Per RFC 2581, a TCP sender increases its congestion window (cwnd) based on the *number of ACK packets* received[17]. In contrast, SCTP counts the *number of bytes* acknowledged within each received ACK packet. Counting the number of ACK packets received rather than the number of bytes acknowledged within each ACK packet causes bigger performance issues for TCP especially when delayed ACKs [29] are used[18].

   - *When to increase cwnd:* During congestion avoidance, SCTP increases its cwnd only if the cwnd is in full use. This can make SCTP to be less aggressive

in our simulations for TCP.

[15] , which lets us to have send and receive buffer sizes $\geq$ 64K.

[16] 10 bytes for the timestamps option, 2 bytes for two *TCP no operation* option.

[17] Note that the ABC (Appropriate Byte Counting) enhancement for TCP is later introduced with RFC 3465 [16]. However, QualNet 4.5.1 does *not* implement ABC in TCP.

[18] Note that, we used delayed ACKs in our simulations.

in sending data.

- *Initial cwnd size*: Initial TCP cwnd size is 1-2 segments according to RFC 2581[19]. SCTP's initial cwnd size at slow start or after long idle periods is set to $min(4 * MTU, max(2 * MTU, 4380\ bytes))$, which will be higher than TCP's initial window size.
- *When to apply slow start vs. congestion avoidance:* SCTP increases its cwnd according to the slow start algorithm when $cwnd \leq ssthresh$, and applies the congestion avoidance algorithm, otherwise. On the other hand, RFC 2581 lets an implementation choose between slow start and congestion avoidance when cwnd and ssthresh are equal[20].

In summary, messaging overhead of SCTP might be higher compared to TCP (especially if no TCP options used). However, SCTP is a newer protocol compared to TCP; hence, some of TCP's enhancements (such as SACKs, ABC [16], initial congestion window size [17]) that came after RFCs 2581 and 2582 are already built-in features in SCTP. Therefore, it should not be surprising to see that SCTP throughput may be better than TCP's under identical conditions (further on this issue in Section 4.5).

4.4 Experimental Framework

In the following sub-sections, we describe different aspects of the simulation framework used this chapter.

---

[19] Note that, RFC 3390 [17] later on updated TCP's initial cwnd size to be up to 4K; however, QualNet 4.5.1 does not implement RFC 3390 and keeps TCP initial cwnd size at 2 segments.

[20] QualNet 4.5.1 applies the congestion avoidance algorithm when cwnd and ssthresh are equal. Hence, this is the same behavior as in the SCTP specification.

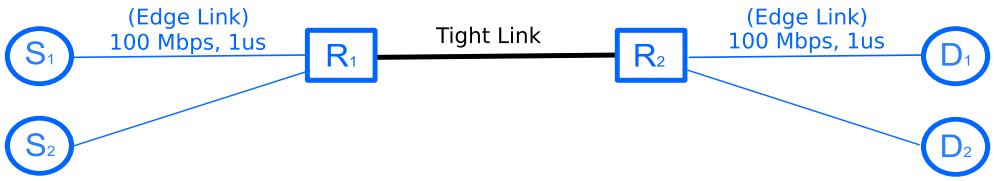


Figure 4.2: Simulation Topology

4.4.1 Topology

We designed an experimental framework to explore TCP-friendliness of SCTP in a single shared tight link topology as depicted in Figure 4.2. In the figure, the edge links use fast Ethernet (with 100 Mbps bandwidth capacity and 1 micro second one-way propagation delay). The tight link is modeled as a full-duplex point-to-point link, with a 45 msec one-way propagation delay[21]. The bandwidth of the tight link is varied as 5, 10, and 20 Mbps. Note that no artificial packet losses are introduced in the tight link or the edge links. Therefore, all the losses in the simulations are due to buffer overflows of congested traffic at routers R1 and R2. The buffer size at routers R1 and R2 is set to the bandwidth-delay product of the path. We use drop tail queues at the routers. We run simulations with QualNet[22].

4.4.2 Network Traffic

The traffic in the network is composed of two flows. **Flow 1** and **flow 2** are applications transmitting data over an SCTP association or a TCP connection from S_1 to D_1 and S_2 to D_2, respectively. The traffic flows are greedy (i.e., the applications at the sending hosts S_1 and S_2 always have data to send). In addition, the receiving applications at hosts D_1 and D_2 are always ready to consume whatever the transport layer protocol can

[21] Note that, the RTT of the paths in the network is 90 msec, similar to US coast-to-coast [93] values.

[22] using svn revision 10 of the SCTP module in QualNet 4.5.1. Note that, QualNet's TCP model uses code converted from the FreeBSD 2.2.2 source code implementation of TCP.

deliver. Therefore, the sending rate of the traffic sources is not limited by the application but by the network. The size of each application message (or A-PDU) is 1200 bytes to create SCTP-PDUs close to the size of path MTU (1500 bytes). Similarly, TCP-MSS (maximum segment size) is set to 1212 bytes[23].

4.4.3 Transport Protocol Parameters

While comparing SCTP and TCP[24], we tried our best to make the transport protocol parameters to be as close as possible in the simulations. Table 4.1 lists what parameters are used in common and per transport layer protocol, respectively. For TCP, we studied both TCP SACK (TCPS) [80] and TCP NEWRENO (TCPNR) [51]. We assumed unlimited send[25] and receiver buffer[26] size at the transport layer so that buffer size is *not* a limiting factor for the transport protocol throughput.

4.4.4 The Framework

Our goal is to understand how two flows (TCP and/or SCTP) share the available bandwidth in the network. We investigate two cases.

- **Case-I**: The two flows in the network are *started at the same time*[27]. We use Case-I to investigate *how two flows grow together* by looking into all possible TCP-SCTP

[23] Note that, QualNet 4.5.1 complies with Section 4.2.2.6 of RFC 1122 [29] and calculates the maximum data that can be put into an TCP-PDU based on the *effective-MSS*. Since every TCP-PDU included timestamps option (extra 12 bytes) in our simulations, we set the TCP-MSS to 1212, to let TCP effectively send 1200 bytes of data in each PDU, similar to SCTP.

[24] See Section 4.3, for the subtle differences between single-homed SCTP and TCP flavors.

[25] Send buffer size of each transport protocol is set to $2 \times bandwidth\text{-}delay$ product.

[26] The receiver buffer size of each transport protocol is set to a large value such as, 65535*2^{14} bytes.

[27] In the simulations, we started the two flows at random times within $[0...RTT]$ to get different randomized results with the repetition of the experiments.

Table 4.1: Transport protocol parameters and their values used in the simulations

Scheme	Parameter	Value
TCP specific	Window scaling option[a]	YES
	Timestamps option[b]	YES
	Other TCP parameters	QualNet 4.5.1 default values
SCTP specific	SCTP-RTO-INITIAL	3 sec
	SCTP-RTO-MIN	1 sec
	SCTP-RTO-MAX	60 sec
	SCTP-RTO-ALPHA	0.125
	SCTP-RTO-BETA	0.25
	Heartbeats	OFF
	Bundling	NO
common in SCTP & TCP	Send Buffer	unlimited
	Receive Buffer	unlimited
	Clock Granularity	500 msec
	Initial ssthresh	$65535*2^{14}$
	Delayed ACKs [18, 29]	YES[c]

[a] The *window scaling option* is required for TCP to have a receiver buffer size bigger than 64K. We activated the window scaling option for TCP flows so that TCP sending rate is not limited by the receiver buffer size.

[b] This parameter is automatically activated by QualNet, since QualNet 4.5.1 implements both window scaling and timestamps options (i.e., [64] together).

[c] The transport receiver sends a T-ACK for every other in-sequence TSN received or when the delayed ACK timer expires. The delayed ACK timer is set to 200 msec.

combinations.

(i) Start **two TCP** flows at the same time

(ii) Start **two SCTP** flows at the same time

(iii) Start **one SCTP** and **one TCP** flow at the same time

- **Case-II**: Initially only one flow is started[28]. Then, we introduce another flow *when the earlier flow operates at steady-state*[29]. Hence, we explore *how one flow gives way to another flow*. We simulated four combinations in Case II.

  (i) Start **one TCP** flow then start **another TCP** flow

  (ii) Start **one SCTP** flow then start **another** SCTP flow

  (iii) Start **one SCTP** flow then start **one TCP** flow

  (iv) Start **one TCP** flow then start **one SCTP** flow

The simulation time for Case-I is 720 seconds, and the simulation time for Case-II is 2140 seconds. For Case-I, we looked into performance metrics between 60^{th} and 660^{th} seconds. For Case-II, we looked into performance metrics between 10^{th} and 70^{th} seconds (when there is only one flow in the network) as well as between 280^{th} and 2080^{th} seconds (when both flows operate at steady-state). Note that, we used both TCPS and TCPNR for the TCP-SCTP combinations above.

4.4.5 Performance Metrics

The performance metrics we measured in the simulations are presented below. We looked into the long-term (steady-state) values of these metrics. In addition, we looked into the throughput metric over short time durations (1, 10, and 100 RTT) – see Figures

[28] at a random time between $[0...RTT]$.

[29] In the simulations, the latter flow is started at a random time between $80sec + [0..RTT]$.

4.8, 4.17, and 4.16. The long-term metric values in Figures 4.3, 4.4, 4.5, 4.6, 4.7, 4.9, 4.10, 4.11, 4.12, 4.13, 4.14, and 4.15 and Tables 4.2 and 4.3 are averages of 30 runs with 95% confidence intervals.

- **Throughput** – Application-layer data (including original, fast-retransmitted, and re-transmitted data that can be potentially lost) sent to the network by the transport protocol of the sending host per unit time. Throughput of a transport flow is shown as a *fraction* of the tight link bandwidth obtained by the flow in the graphs (i.e., throughput per flow is *normalized* with the tight link bandwidth).

- **Load by the transport protocol or Transport Load (T-Load)** – The actual number of bits sent to the network by the transport protocol per unit time. This includes all the transport layer headers, original, fast-retransmitted, and re-transmitted application-layer data and transport layer ACKs that can be potentially lost. T-Load per transport flow is normalized with the tight link bandwidth.

- **Goodput** – The application layer throughput measured at the receiving host. That is the number of bits delivered to the application layer of the receiving host by the transport layer per unit time. Goodput per transport flow is normalized with the tight link bandwidth.

While the metrics above (throughput, t-load, and goodput) are measured per flow in the network, the following metrics (fairness index, link utilization, and system utilization) are aggregated metrics and measured per configuration.

- **Fairness Index** – This metric is defined by R. Jain [67] to show fairness (i.e., the "equality" of resource sharing) in a system. Fairness index is a value between 0 and 1, with 1 showing the most fair (equal) allocation of the resources in the system. Assuming λ_i is the rate (throughput) of transport flow i, the fairness index

of the network is given by Equation 4.2, where n is the total number of flows in the network.

$$FIndex = \frac{(\sum_{i=1}^{n} \lambda_i)^2}{n * (\sum_{i=1}^{n} \lambda_i^2)} \tag{4.2}$$

- **Link Utilization** – We use Equation 4.3 to calculate link utilization (*for the tight link*), where λ_i is throughput of transport flow i and n is the total number of flows in the network. Our aim in using this metric is to see if the transport flows pump enough data traffic into the network. We want the link utilization to be high so that the network operates close to its capacity[30].

$$LinkUtil = \frac{(\sum_{i=1}^{n} \lambda_i)}{Tight\ Link\ Bandwidth} \tag{4.3}$$

- **System Utilization** – This metric is calculated using Equation 4.4, where n is the number of flows in the network, α_i is the goodput, and γ_i is the t-load of the transport flow i. Essentially, this metric shows how much of the total load in the network is converted to useful work (i.e., the data received by the applications). One of the signs of congestion collapse is, although there is traffic (load) in the network, the load is not converted into useful work and the network is busy transmitting unnecessary data traffic. Therefore, the higher the system utilization, the further away the system is from congestion collapse.

$$SysUtil = \frac{(\sum_{i=1}^{n} \alpha_i)}{(\sum_{i=1}^{n} \gamma_i)} \tag{4.4}$$

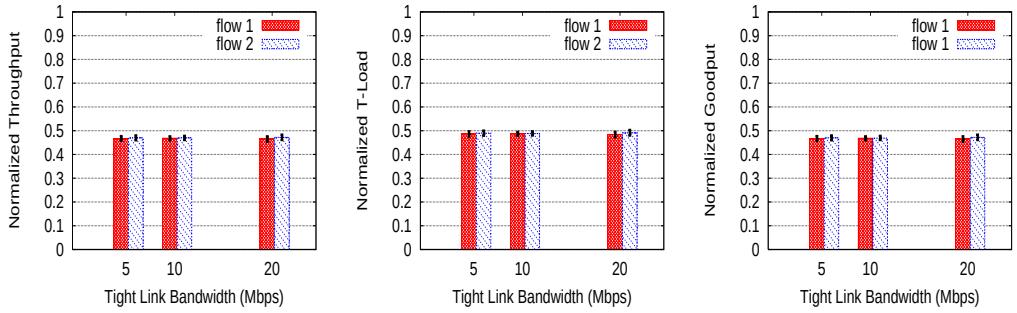
4.5 Simulation Results and Analysis

In this section we present the results from two sets of experiments we performed. Sections 4.5.1 and 4.5.2 discuss the results (i) when both flows in the network start at the same time, and (ii) when one flow starts after the other flow at steady-state, respectively.

[30] That is close to the "knee" as Chiu and Jain suggested [37].

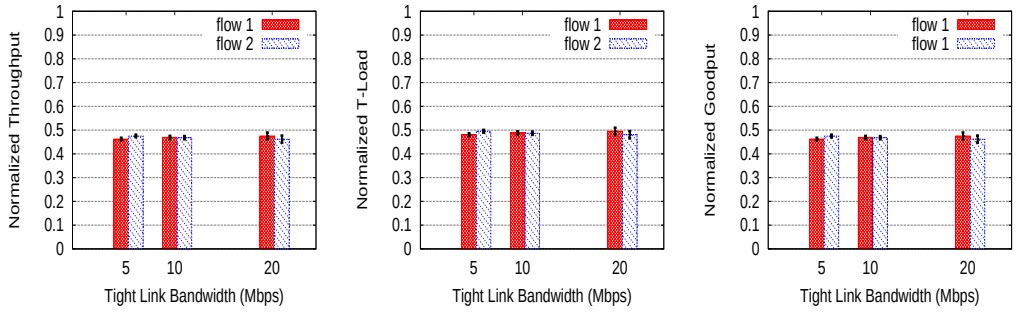
4.5.1 Flows Starting at the Same Time

Results for the flows starting at the same time are presented in Figures 4.3, 4.4, 4.5, 4.6, 4.7, and 4.8 and Table 4.2.



(a) (b) (c)

Figure 4.3: flow 1: TCPS, flow 2: TCPS, starting at the same time



(a) (b) (c)

Figure 4.4: flow 1: TCPNR, flow 2: TCPNR, starting at the same time

- *Two TCP flows start together*: From Figures 4.3 and 4.4 and Table 4.2, we observe that two TCP flows (for both TCPS and TCPNR flavors) share the link bandwidth pretty equally, irrespective of increase in bandwidth, as depicted with close individual throughput values per flow in the network as well as the high fairness index values. TCP congestion control algorithms allow aggregated flows to pump enough

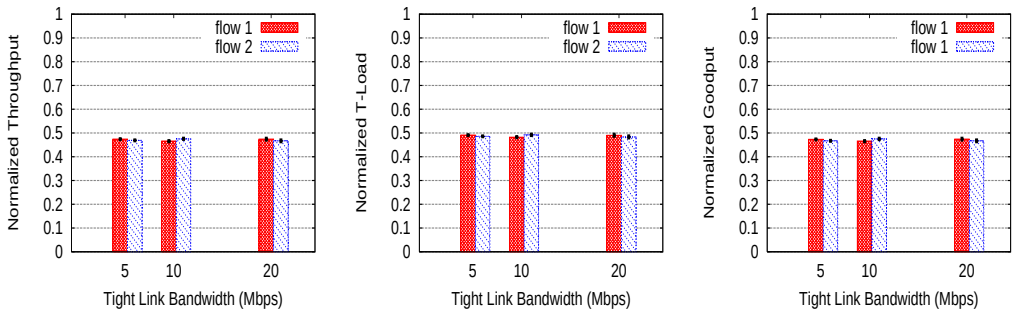


Figure 4.5: flow 1: SCTP, flow 2: SCTP, starting at the same time

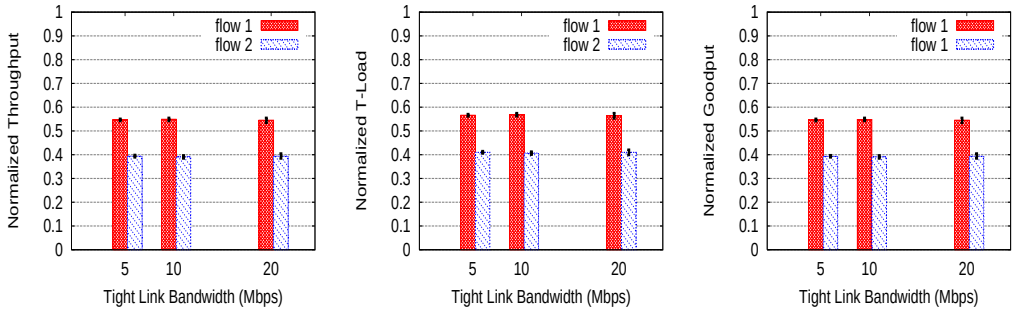


Figure 4.6: flow 1: SCTP, flow 2: TCPS, starting at the same time

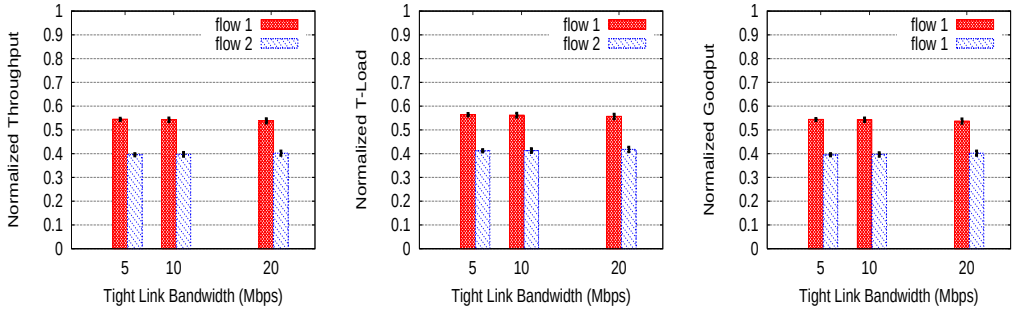
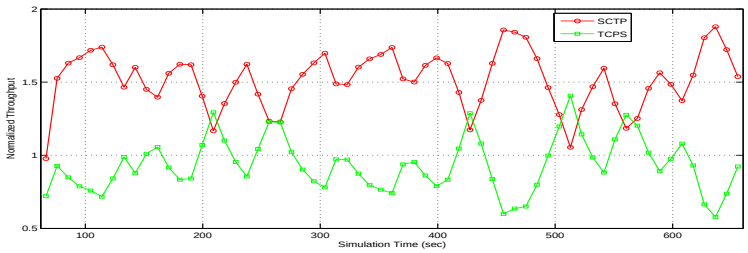
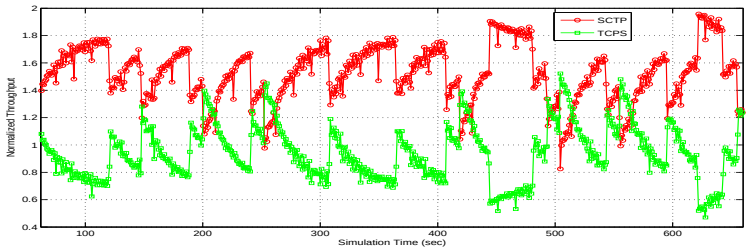


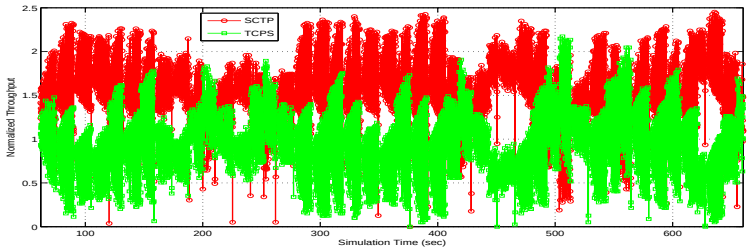
Figure 4.7: flow 1: SCTP, flow 2: TCPNR, starting at the same time



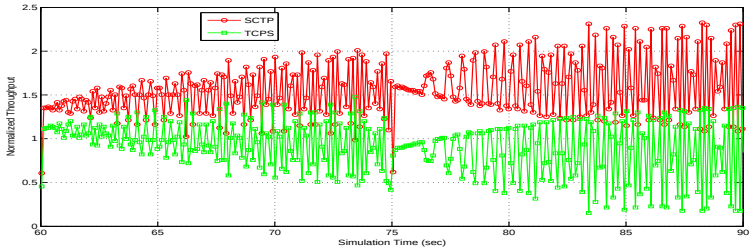
(a) 100 RTT



(b) 10 RTT



(c) 1 RTT



(d) 1 RTT, 60-90 seconds

Figure 4.8: Throughput of TCPS (green) and SCTP (red), starting at the same time, for different time intervals

Scheme	BW	FI	LinkUtil	SysUtil
TCPS, TCPS	5	0.997	0.938	0.960
	10	0.998	0.938	0.961
	20	0.996	0.937	0.961
TCPNR, TCPNR	5	0.999	0.938	0.961
	10	0.998	0.938	0.961
	20	0.993	0.937	0.961
SCTP, SCTP	5	1.0	0.943	0.963
	10	0.999	0.941	0.965
	20	0.999	0.941	0.966
SCTP, TCPS	5	0.974	0.940	0.962
	10	0.971	0.940	0.963
	20	0.972	0.939	0.964
SCTP, TCPNR	5	0.975	0.940	0.962
	10	0.974	0.939	0.963
	20	0.977	0.939	0.964

Table 4.2: Fairness Index, Link Utilization, and System Utilization when *both flows start at the same time*

traffic into the network (where link utilization values are more then 93%). In addition, the system utilization is high confirming that TCP is busy sending useful data traffic (i.e., no signs of congestion collapse).

- *Two SCTP flows start together*: Similar to the two TCP flow case, we observe that two SCTP flows starting at the same time also share the bandwidth equally (Figure 4.5), irrespective of increase in tight link bandwidth. The transport load values for the two SCTP flows are also close to the transport loads of the two TCP flows, showing that SCTP and TCP protocol overheads are similar for the configurations we have in the simulations. The link and system utilities are high ($\geq$94% and $\geq$96%, respectively) proving that SCTP congestion control algorithms causing the network to operate at high capacity without any threat of congestion collapse.

- *One SCTP and one TCP flows start together*: From Figures 4.6 and 4.7, we observe that irrespective of the increase in the tight link bandwidth, on average SCTP gets

35%-41% larger share of the bandwidth compared to TCPS (or TCPNR). However, the link and the system utility values are still high showing a stable network (see Table 4.2). We looked further into how the throughput of SCTP and TCPS changes over 1, 10, and 100 RTT intervals – Figure 4.8. We picked the *worst case* simulation run where SCTP throughput is largest[31] compared to TCPS among all 30 runs. Figure 4.8 validates that although SCTP is able to achieve higher throughput than TCPS, even *in the worst case*, SCTP responds to TCP traffic by increasing and decreasing its throughput. That is, even in the most aggressive and imbalanced case of 30 runs, SCTP does not simply take as much bandwidth as it can get; rather, over time SCTP gives and takes in a sharing with TCP. Therefore, the figure helps arguing for SCTP being TCP-friendly.

4.5.2 Latter Flow Starts after Earlier Flow Is at Steady-State

Results for Case II, where one flow starts earlier and the latter flow starts after the earlier flow is at steady state, are depicted in Figures[32] 4.9, 4.10, 4.11, 4.12, 4.13, 4.14, 4.15, 4.16, and 4.17 and Table 4.3.

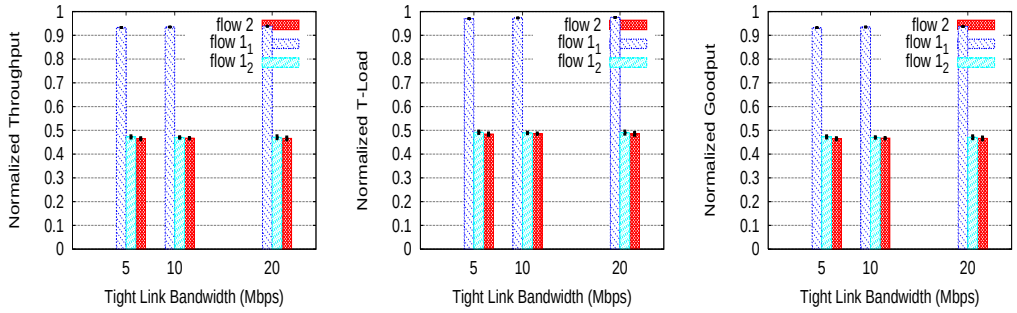


Figure 4.9: flow 1: TCPS followed by flow 2: TCPS

[31] In this particular run SCTP gets for about 62% more bandwidth than TCPS.

[32] Note that, in these figures *flow* 1_1, refers to the state of flow 1 when there is no other flow in the network, and *flow* 1_2 refers to the state of flow 1 after a second flow is introduced into the network.

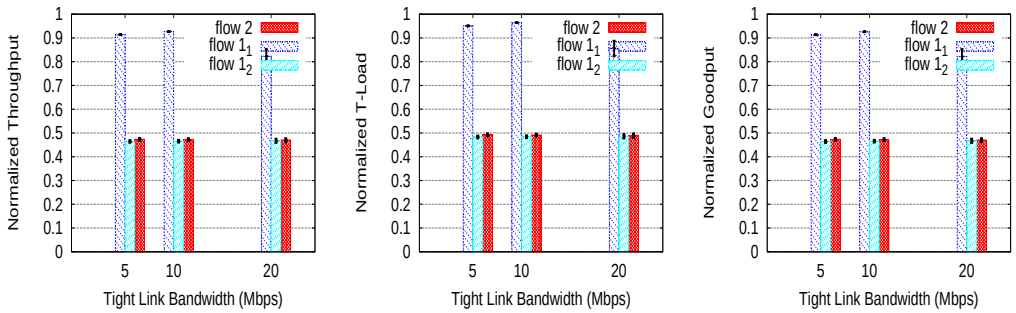


Figure 4.10: flow 1: TCPNR followed by flow 2: TCPNR

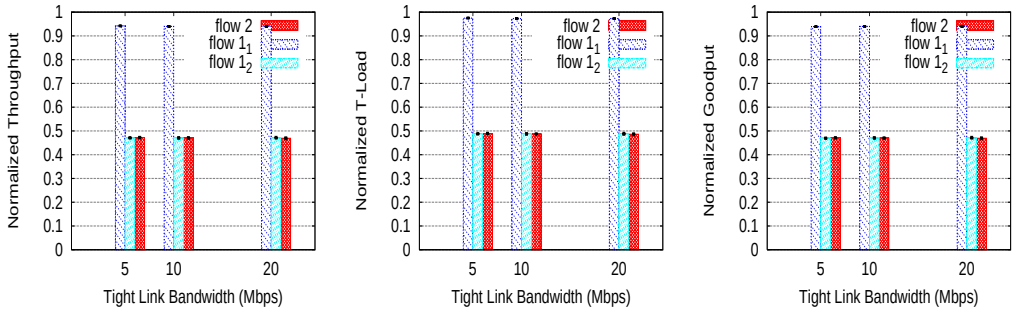


Figure 4.11: flow 1: SCTP followed by flow 2: SCTP

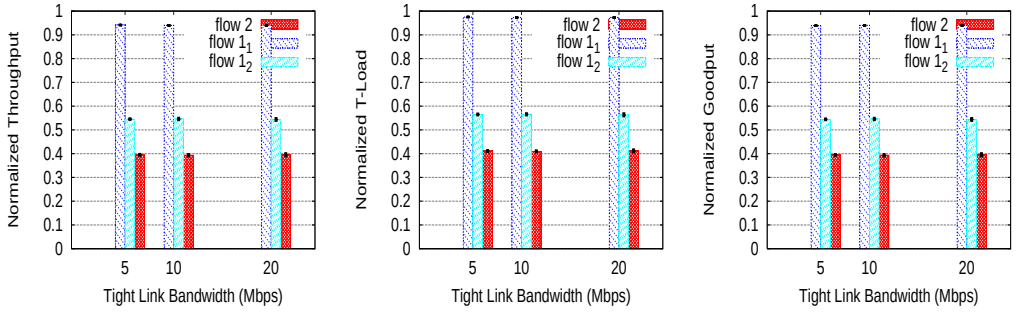
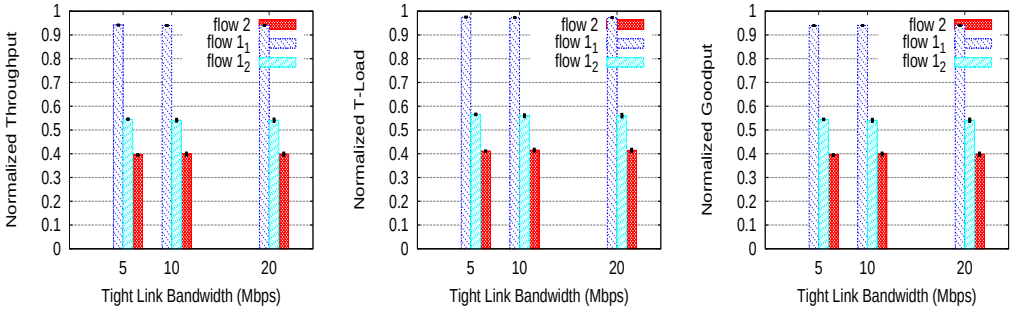
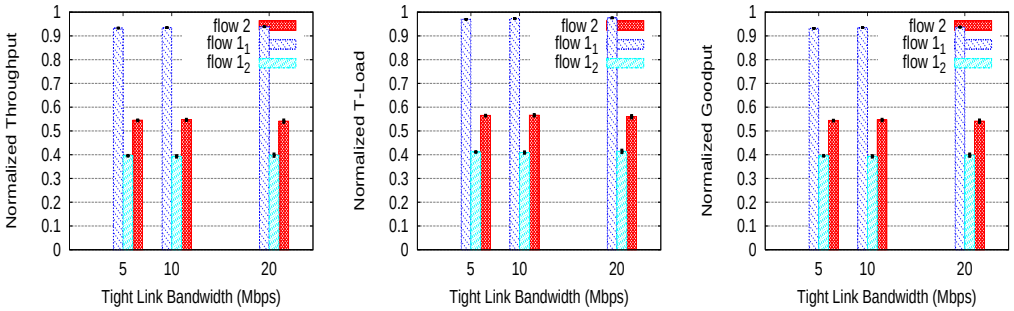


Figure 4.12: flow 1: SCTP followed by flow 2: TCPS



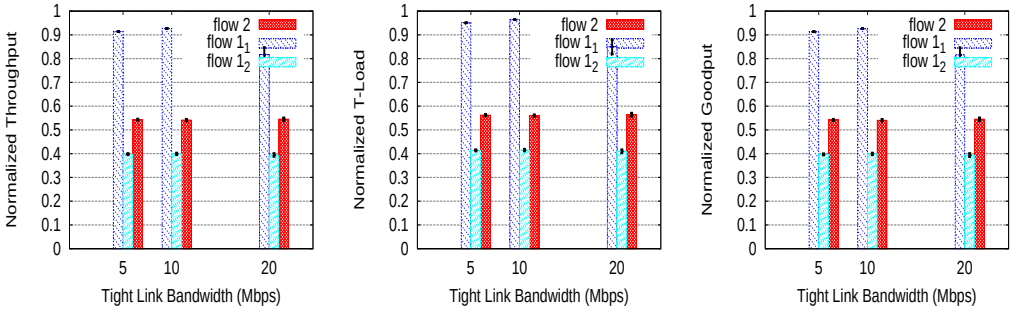
(a) (b) (c)

Figure 4.13: flow 1: SCTP followed by flow 2: TCPNR



(a) (b) (c)

Figure 4.14: flow 1: TCPS followed by flow 2: SCTP

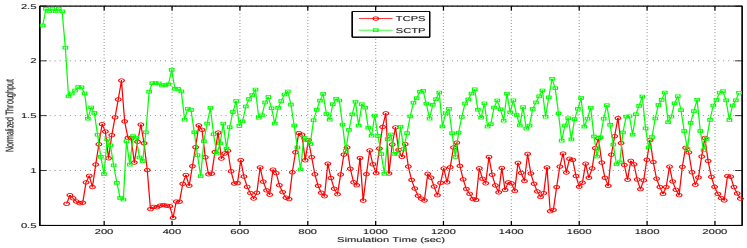


(a) (b) (c)

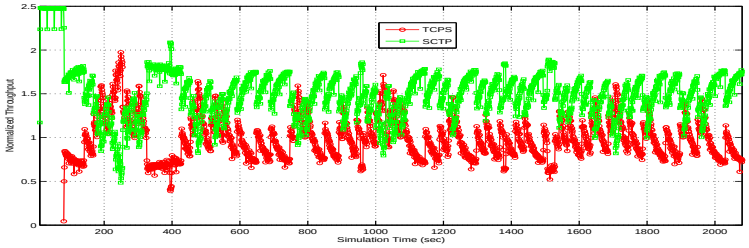
Figure 4.15: flow 1: TCPNR followed by flow 2: SCTP

Scheme	BW	FI	LinkUtil	SysUtil
TCPS follw. by TCPS	5	0.999	0.938	0.960
	10	0.999	0.938	0.961
	20	0.998	0.938	0.961
TCPNR follw. by TCPNR	5	0.999	0.938	0.961
	10	0.999	0.938	0.961
	20	0.998	0.937	0.961
SCTP follw. by SCTP	5	1.0	0.943	0.963
	10	1.0	0.941	0.965
	20	1.0	0.941	0.966
SCTP follw. by TCPS	5	0.975	0.940	0.962
	10	0.974	0.940	0.963
	20	0.975	0.939	0.964
SCTP follw. by TCPNR	5	0.975	0.940	0.962
	10	0.977	0.939	0.963
	20	0.976	0.939	0.964
TCPS follw. by SCTP	5	0.975	0.940	0.962
	10	0.974	0.940	0.963
	20	0.976	0.939	0.964
TCPNR follw. by SCTP	5	0.976	0.940	0.962
	10	0.977	0.939	0.963
	20	0.974	0.939	0.964

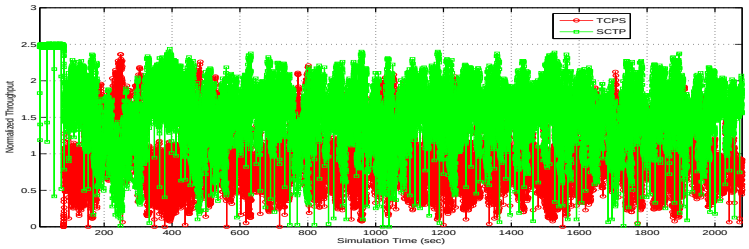
Table 4.3: Fairness Index, Link Utilization, and System Utilization when 2^{nd} flow starts after the 1^{st} flow is at steady-state



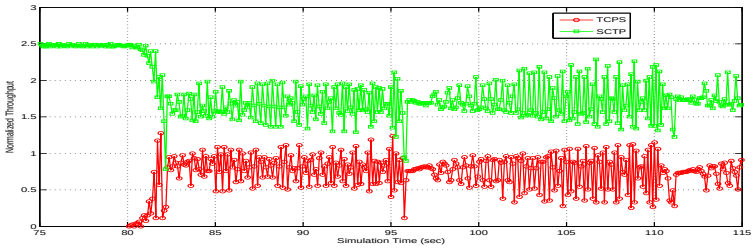
(a) 100 RTT



(b) 10 RTT

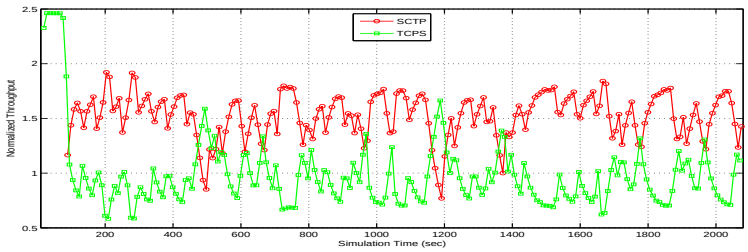


(c) 1 RTT

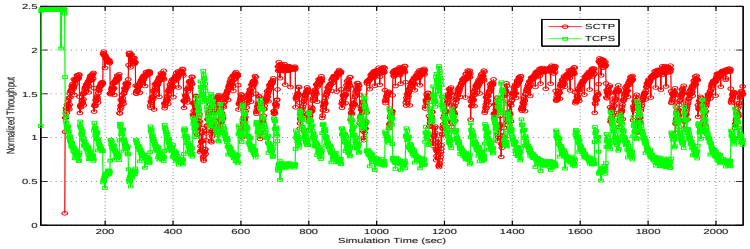


(d) 1 RTT, 75-115 sec

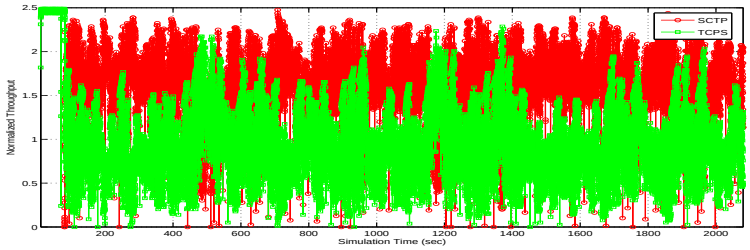
Figure 4.16: Throughput of SCTP (green) followed by TCPS (red), for different time intervals



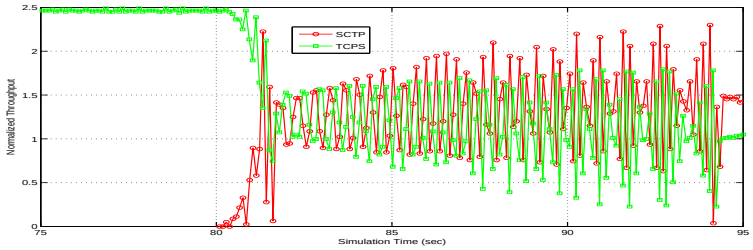
(a) 100 RTT



(b) 10 RTT



(c) 1 RTT



(d) 1 RTT, 75-95sec

Figure 4.17: Throughput of TCPS (green) followed by SCTP (red), for different time intervals

- *One TCP flow followed by another TCP flow*: By examining Figures 4.9 and 4.10, we first observe that as expected, the throughput of the first TCP flow drops after the second TCP flow is introduced to (more or less) half of its previous value (for both TCPS-TCPS and TCPNR-TCPNR cases). The fairness index of the system is also high for both TCPS-TCPS (more than 0.99) and TCPNR-TCPNR (more than 0.99), suggesting that even when the TCP flows start at different times, they still share the link fairly.

- *One SCTP flow followed by another SCTP flow*: By examining Figure 4.11 and comparing the results with those of the TCP flows, we observe that the earlier SCTP flow gives way to the latter SCTP flow, and both share the link fairly (see fairness index values in Table 4.3).

- *One SCTP flow followed by a TCP flow*: In contrast, SCTP does not give a way in an "equally-shared" manner to a later TCP flow – by observing Figure 4.12 (with TCPS) and 4.13 (with TCPNR). Although, the SCTP flow's throughput drops after a TCP flow is introduced, on average SCTP ends up getting a 36%-39% larger share of the bandwidth than TCPS or TCPNR. We believe that the proposed TCP improvements that have been added[33] into the SCTP's congestion control mechanism are responsible to a (faster and) better loss recovery in SCTP and hence SCTP's larger share of the throughput compared to TCP [15, 82, 83, 84]. Figure 4.16 shows the evolution of throughputs when an SCTP flow is followed by a TCPS flow. For this figure, we ploted the *worst* simulation result out of 30 runs where SCTP vs TCP throughput was most imbalanced[34]. The graph helps argue for SCTP being TCP-friendly as although SCTP gets higher throughput compared to TCP, SCTP still gives way to TCP and shares the bandwidth with the newly introduced TCPS in a give-and-take manner.

[33] Refer to Section 4.3.

[34] In this particular run, SCTP gets 53% more share of the bandwidth than TCPS.

- *One TCP flow followed by an SCTP flow*: When it comes to SCTP getting its share of bandwidth from an already stabilized TCP flow (Figures 4.14 and 4.15), SCTP again achieves higher throughput than an existing TCP flow. Moreover, the bandwidth obtained is put into useful work by SCTP, as the high system utility values in Table 4.3 suggest. The evolution of throughput for TCPS and SCTP flows for different time intervals for the most *imbalanced* run out of 30 runs where SCTP achieves 61% more bandwidth than TCPS is depicted in Figure 4.17. The figure shows how TCPS gives way to SCTP and how SCTP shares the bandwidth with TCPS in give-and-take manner. The figure again helps argue for SCTP being TCP-friendly.

4.6 Related Work

Although SCTP has been standardized by IETF in 2000, there is *little* work comparing the performance of (single-homed) SCTP with competing TCP flavors. Reference [30] used the *Opnet simulator* to perform initial simulation studies to find possible flaws in the early version of SCTP specification and implementation in 2001. Reference [69] looked into the throughput of competing SCTP and TCP connections in a shared link topology using *SCTP reference implementation* from 1999 on a test-bed (with Linux machines and NIST network emulator). Reference [69] focused on the impact of the message (A-PDU) size in comparing TCP and SCTP throughputs and showed that SCTP is *not more aggressive than TCP* when sharing a link. Reference [15] studied competing SCTP and TCP flows over a satellite (high bandwidth-delay product) link using *ns-2*. Reference [15] found out that the subtle enhancements in the SCTP congestion control mechanism help SCTP to recover faster after a loss and hence increase the throughput of SCTP compared to TCP. The results of [15], although for higher bandwidth-delay product links, align with our simulation results presented in this chapter.

4.7 Conclusions

In this chapter, we investigate the dynamics of two competing SCTP and TCP flows in a *shared* tight link topology using QualNet simulations. Our goal was to investigate SCTP–TCP competition somehow in the micro scale to identify the main subtle differences between SCTP and TCP's TCP-friendly-related-mechanics. We discovered that although SCTP's congestion control mechanisms were intended to be "similar" to TCP's, being a newer protocol, SCTP specification has some of the proposed TCP enhancements already incorporated which results in SCTP performing better than TCP. Therefore, SCTP can obtain larger share of the bandwidth when competing with a TCP flavor that does not have similar enhancements (as in the case of QualNet's TCP implementation). We conclude that SCTP is TCP-friendly but achieves higher throughput than TCP, due to SCTP's better loss recovery mechanisms[35] just as TCP-SACK or TCP-Reno perform better than TCP-Tahoe.

[35] Reports in [15, 82, 83, 84] also support our conclusion.

Chapter 5

TCP-FRIENDLINESS OF CMT

This chapter investigates the TCP-friendliness of CMT by designing an experimental framework and studying performance results of QualNet simulations. The organization of the chapter is as follows. Section 5.1 introduces the problem statement. Sections 5.2 and 5.3 describe the experimental framework, and the simulation results and analysis, respectively. Section 5.4 reviews related work and the recent controversies over the TCP-friendliness doctrine. Section 5.5 concludes the chapter.

5.1 Problem Statement

In Chapter 4, we described the definition and the goals of the TCP-friendliness doctrine. Traditionally, the notion of TCP-friendliness was defined for end-to-end transport connections over a single path. Our goal is to understand and quantify the TCP-friendliness of SCTP-based CMT for transport connections over multiple paths.

The design goal of CMT was to achieve a performance similar to the aggregated performance of multiple, independent, single-homed SCTP associations (called App-Stripe). Iyengar et. al. showed that the throughput of CMT can be similar or greater[1] (especially when the receiver buffer is unconstrained and the paths are showing similar characteristics) than AppStripe [60]. They studied the performance of CMT under the assumption that the network paths which CMT subflows run over are *bottleneck-independent* (similar to Figure 6.1). Since we are interested in the TCP-friendliness of

[1] Due to the sharing of the TSN space, CMT is more robust to ACK losses.

CMT, we revise this assumption and investigate how CMT behaves when a *tight link*[2] is shared among the CMT subflows and together with other TCP flows.

5.2 Experimental Framework

In the following sub-sections, we describe different aspects of the experimental framework used this chapter. Experiments are simulated in QualNet[3].

5.2.1 Topology

We designed an experimental framework to explore the TCP-friendliness of CMT in a single shared tight link topology[4] as depicted in Figure 5.1. The tight link has 100 Mbps bandwidth capacity and 2 msec one-way propagation delay. Each edge link has 100 Mbps bandwidth capacity and 14 msec one-way propagation delay[5]. No artificial packet losses are introduced in the tight link or the edge links. Therefore, all the losses in the simulations are due to buffer overflows at routers R1 and R2 caused by network traffic. We used RED queues [57] in our simulations. Note that, when a group of TCP flows share a tight link with drop-tail queues, *global synchronization* and *phase effects* can cause the TCP flows not to get "equal" share of the bandwidth [53, 57, 90, 91, 94] (i.e., TCP becomes TCP-unfriendly). Introducing randomness to the network helps reducing the impact of global synchronization and phase effects [57]. We calibrated RED parameters in our simulations so that TCP flow show TCP-friendly behavior (i.e., all TCP flows in the network get "equal" share of the tight link). As recommended by references [57, 54], $\min_{th}$ is set to 5 packets, $\max_{th}$ is set to three times $\min_{th}$, w_q is set to 0.002, and

[2] We prefer to use the term *tight link* [66] rather than *bottleneck*, in this dissertation.

[3] In this chapter, simulations run with svn revision 10 of the SCTP module in QualNet 4.5.1.

[4] Our topology is similar to the access link scenario in [20].

[5] Note that, the RTT of the paths in the network is 60 msec, similar to US coast-to-coast [93].

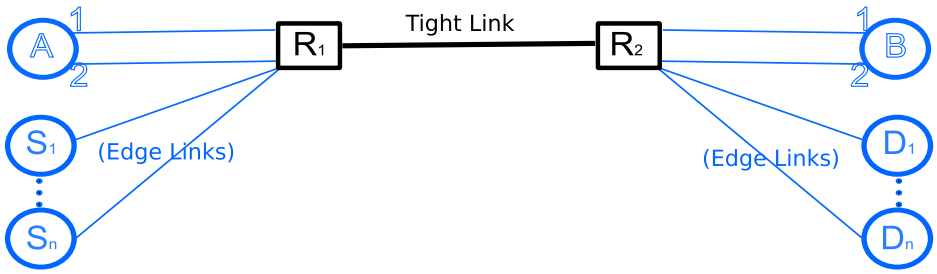


Figure 5.1: Simulation Topology

$\max_p$ is set to 0.02 in our simulations. The buffer size at routers R1 and R2 is set to the bandwidth-delay product (BDP).

5.2.2 Network Traffic

In our experimental topology (Figure 5.1), nodes A and B are multihomed hosts while nodes S_i and D_i are single-homed hosts. We first run n TCP flows, from source nodes S_i to destination nodes D_i, $1 \leq i \leq n$. We then add one of the following traffic loads into the network.

- **Flows TCP1 and TCP2:** Two additional TCP flows running over the network paths A1-R1-R2-B1 and A2-R1-R2-B2, respectively.
- **Flows SCTP1 and SCTP2:** Two single-homed SCTP[6] flows running over the network paths A1-R1-R2-B1 and A2-R1-R2-B2, respectively.
- **CMT flow:** a two-homed CMT flow from host A to host B, with two subflows *CMT-sub1* and *CMT-sub2*, running over the network paths A1-R1-R2-B1 and A2-R1-R2-B2, respectively.

For our experiments, n is varied as 8, 16, 32, 48, and 64 yielding a total of 10, 18, 34, 50, and 66 network flows[7]. All flows in the network are greedy (i.e., the applications

[6] From here on in this chapter, *SCTP* means single-homed SCTP.

[7] We counted each CMT subflow as one flow.

at the sending hosts always have data to send). In addition, the receiving application at each host is always ready to consume whatever the transport layer protocol can deliver. Therefore, the sending rates of the traffic sources are not limited by the applications but by the network. The size of each application message (or A-PDU) is 1200 bytes to create SCTP-PDUs close to the size of path MTU (1500 bytes). Similarly, TCP-MSS is set to 1212 bytes.

5.2.3 Transport Protocol Parameters

Single-homed SCTP associations and TCP[8] connections are using parameter similar to what has been described in Section 4.4.3 and Table 4.1. The CMT association uses DAC and RTX-CWND as RTX policy. Both sender and receiver buffers at the transport connections are unlimited.

5.2.4 The Framework

We first started n TCP flows from nodes S_i to D_i randomly between the 0^{th} and the 5^{th} seconds of the simulation. Then, at the 10^{th} second, we introduced either (i) the CMT flow, (ii) TCP1 and TCP2 flows, or (iii) SCTP1 and SCTP2 flows into the network. For each case, we measured the performance (mainly the sending rate) of the TCP flows from nodes S_i to D_i, and the performance of the newly introduced flow(s). Our goal in this framework is to see if CMT behaves more or less aggressively than the two independent TCP connections or SCTP associations. We explore answers to the following questions.

- TCP's congestion control algorithms aim to achieve an "equal" share of the tight link bandwidth. How much of the bandwidth sharing an CMT flow could achieve compared to two independent TCP or SCTP flows?
- What is the cost of introducing one CMT flow into the other network flows compared to introducing two independent TCP or SCTP flows?

[8] All the TCP flows in this chapter are TCP-SACK connections.

5.2.5 Metrics

We measured the following metrics in our simulations.

- **Per flow throughput** – similar to the definition given in Section 4.4.5, we defined throughput (sending rate) of a transport flow as the amount of application-layer data (including the original, the fast-retransmitted, and the re-transmitted data, that may potentially be lost) put on the wire by the transport protocol sender per unit time.

- **Average (avg.) flow throughput** – is calculated using Equation 5.1, where λ_i is the throughput (sending rate) of transport flow $i \in [1..(n+2)]$. While calculating the avg. flow throughput, we counted each CMT subflow as one flow. We expected the avg. flow throughput to be close to the equal share of the available bandwidth in the network.

$$avg.\ flow\ throughput = \frac{\sum_{i=1}^{n+2} \lambda_i}{n+2} \tag{5.1}$$

- **Fairness Index** – as defined by Equation 4.2 in Section 4.4.5. We measured the fairness index of all the flows in the network (each CMT subflow is counted as one flow) to understand how equally flows in the network actually share the available bandwidth.

We run simulations for 36 minutes. The result for each configuration is averaged over 30 runs with a 95% confidence interval. We measured the metrics between the 3^{rd} and 33^{rd} minutes.

5.3 Simulation Results and Analysis

Before analyzing the simulation results, we have the following hypotheses for each case.

- *Introducing two TCP flows:* After TCP1 and TCP2 have been introduced into the network, we expect all the flows (including the newly introduced TCP flows) to get more or less an equal share of the available bandwidth.

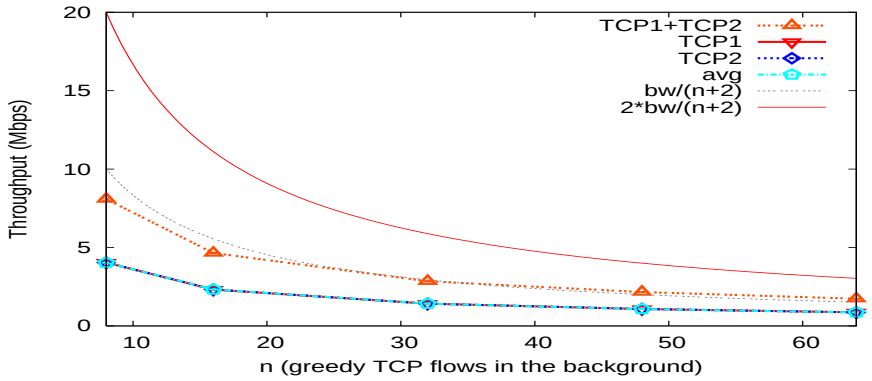
- *Introducing two SCTP flows:* After SCTP1 and SCTP2 have been introduced into the network, we expect the SCTP flows to get similar or higher throughput than the existing TCP flows in the network. As elaborated in Chapter 4, the proposed TCP improvements that have been incorporated into the SCTP's congestion control mechanism facilitate better loss recovery and hence improved throughput of SCTP compared to TCP [15, 82, 83, 84].

- *Introducing the CMT flow:* In a tight-link-independent topology (with drop-tail queues), CMT achieves higher throughput than the independent SCTP flows (especially when the receiver buffer is unconstrained and the paths have similar characteristics), as CMT shares the TSN space and hence is more resilient to losses [60]. Similarly, in a tight-link-dependent topology (with RED queues) as in Figure 5.1, we expect CMT to obtain higher throughput (i.e., higher share of the tight link bandwidth) compared to two TCP or two SCTP flows.

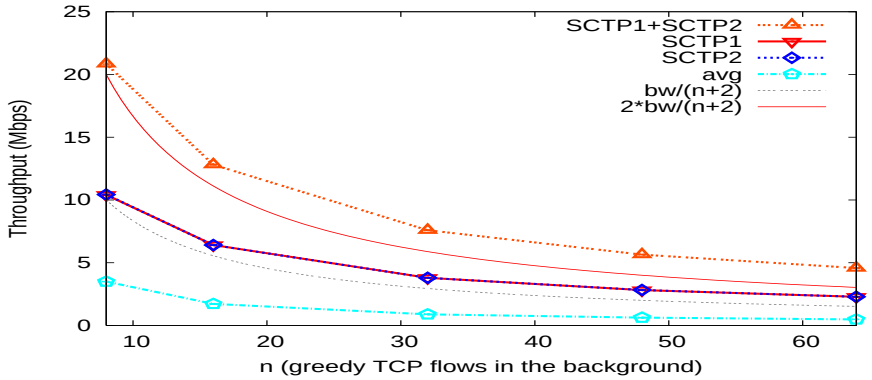
The simulated results are depicted in Figures 5.2 and 5.3. We observed the following from the figures.

- *two-TCP case:* From Figure 5.2(a), TCP shows TCP-friendly behavior, where TCP1, TCP2 and an average TCP flow in the network all get "equal" throughput, which is less than the ideal bandwidth share, $\frac{bw}{(n+2)}$. The high fairness index values (very close to 1) in Figure 5.3 for the two-TCP case also confirm the equal sharing of the bandwidth among the TCP flows. We also checked the throughput of all the individual TCP flows in the network for all the n values and again confirmed that all the TCP flows in the network obtained "equal" throughputs. Appendix A shows the individual TCP flow throughputs for $n = 8$ and $n = 64$.

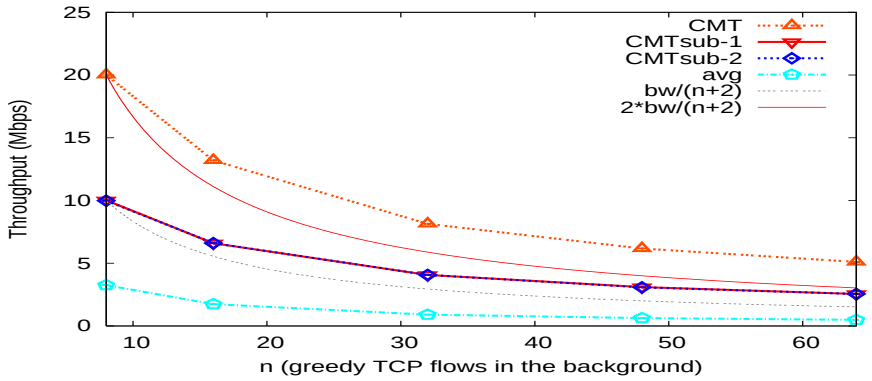
- *two-SCTP case:* From Figure 5.2(b), SCTP1 and SCTP2 get "equal" throughput, but the achieved throughput is higher than both the throughput of an average TCP flow and the ideal share of bandwidth. The low fairness index values in Figure 5.3



(a) two-TCP

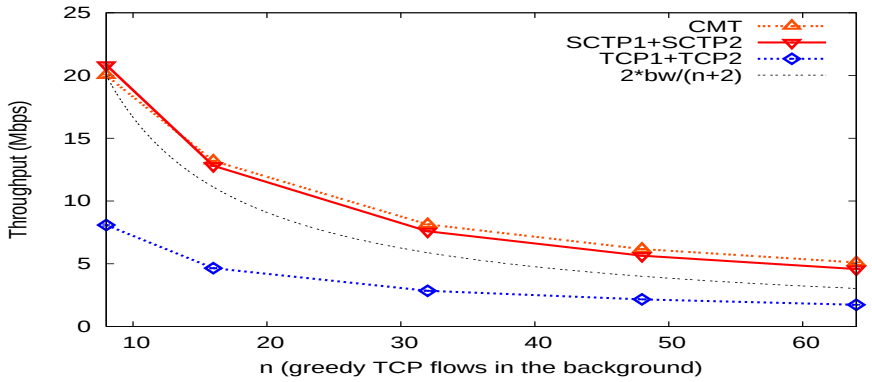


(b) two-SCTP

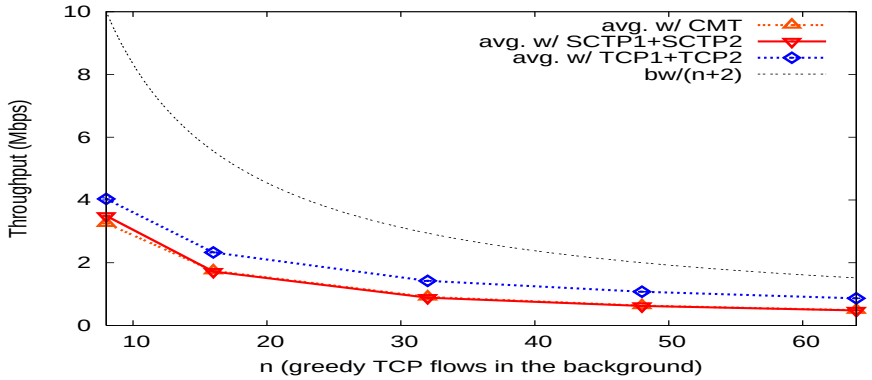


(c) CMT

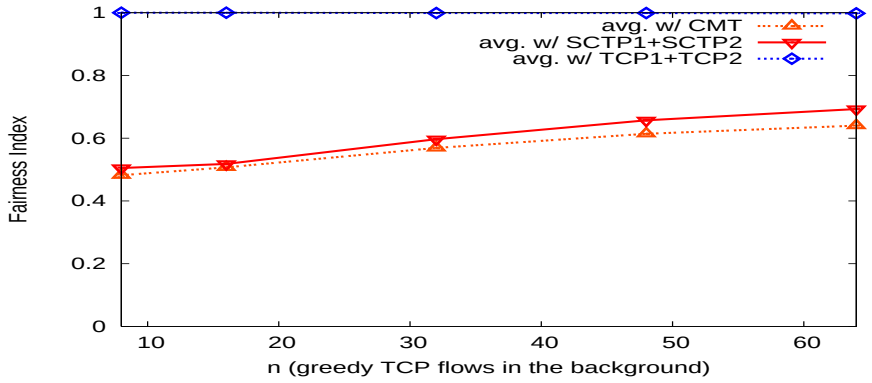
Figure 5.2: Throughputs of two-TCP, two-SCTP, and CMT flow together with the avg. flow throughputs



(a)



(b)



(c)

Figure 5.3: (a) Throughput of two-TCP vs. two-SCTP vs. CMT flows, (b)Average flow throughputs, and (c) Fairness index of all the flows in the network.

for the two-SCTP case are resulted from SCTP flows obtaining higher throughput than the TCP flows in the network. However, as we investigated further, adding the SCTP flows into the network does not "starve" any of TCP flows in the network (see Appendix B of the individual flow throughputs for $n = 8$ and $n = 64$). Such behavior is due to the fact that SCTP implements a congestion control mechanism and does not frivolously send as much as it can. As a result, we see other TCP flows co-existing (without being starved) with two SCTP flows in the network (although SCTP's thruput is higher). Referring back to the definition of TCP-friendliness in Section 4.1, we conclude that SCTP is TCP-friendly but achieves higher throughput than TCP, due to SCTP's better loss recovery mechanisms[9] just as TCP-SACK or TCP-Reno performs better than TCP-Tahoe.

- *the CMT case:* From Figure 5.2(c), each CMT subflow obtains "equal" throughput, but the achieved throughout is higher than the throughput of an average TCP flow in the network. We also checked the individual subflow throughput and confirmed that CMT subflows perform better than the TCP flows in the network (see Appendix C of the individual flow throughputs). As depicted in Figure 5.3(a), CMT performance is better than the total performance of TCP1 and TCP2. We had also expected CMT to perform better than two SCTP flows. However, CMT is actually showing similar or worse (for $n = 8$) performance than two SCTP flows, which invalidates our earlier hypothesis. To further investigate this issue, we run another set of experiments with n values set to 4, 6, 8, 10, and 12. We observed that the performance of CMT gets worse than two SCTP flows as n gets smaller, as depicted in Figure 5.4(a). To investigate the worse performance of CMT compared to two SCTP flows as n gets smaller, we have the following hypothesis.

  hypothesis:* CMT subflows share the same TSN space and ACK information, unlike independent SCTP flows. Therefore, one ACK can simultaneously trigger all

[9] Reports in [15, 82, 83, 84] also support our conclusion.

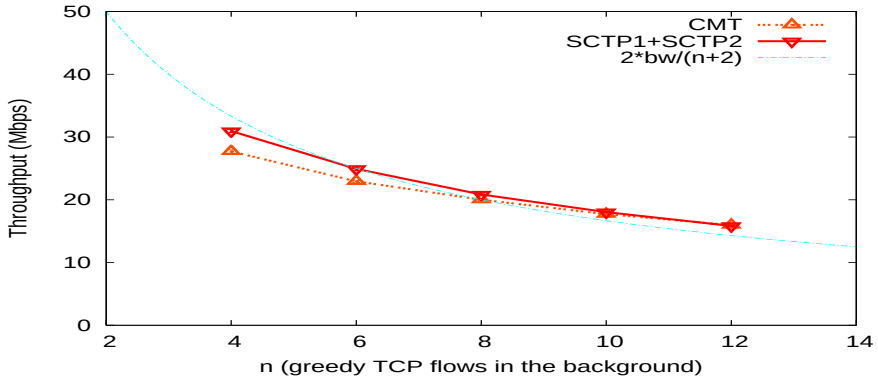
the CMT subflows to send data to the network. Consequently, one CMT flow (containing two CMT subflows) can create *burstier* data traffic compared to two SCTP flows. The burstiness causes more packets of the CMT subflows to be marked by the RED queues. Therefore, the CMT flow does not perform as good as we expected (i.e., better than two SCTP flows).

To validate hypothesis*, we examined the RED parameter w_q that can be adjusted to alter RED's responses to burstiness. The rationale is that if we can make RED to react burstiness less aggressively, then we should observe CMT not performing as bad compared to two SCTP flows.

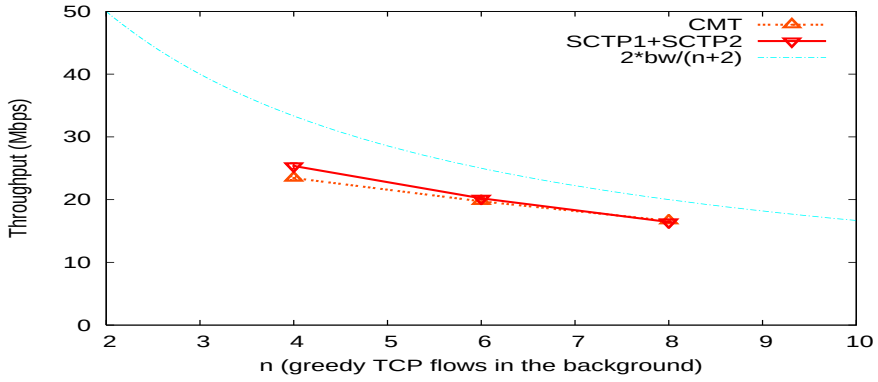
As suggested by [53, 57], we changed w_q to be 0.001 (instead of 0.002 used in other simulations in this chapter), in order for making RED queue to react less aggressively to bursty traffic. The simulation results for $w_q = 0.001$ is depicted in Figure 5.4(b). Comparing Figures[10] 5.4(a) and 5.4(b), in the latter figure, CMT still performs similar or worse than SCTP for small *n*'s, but not as bad as in the former figure. Therefore, we have a reason to believe that hypothesis* holds. One last question on the comparative performance of CMT and two SCTP flows is why CMT performs worse *for smaller n values*? Intuitively, as *n* gets smaller the marking probability at the bottleneck RED queue for each flow in the network increases, and hence burstiness affects each flow more.

After this detour to explain the performance difference between CMT and two SCTP flows, let's get back to the discussion of TCP-friendliness. As stated earlier, one CMT flow (with two subflows) has better throughput than two TCP flows and each CMT subflow has better throughput than TCP1 and TCP2, because of the better loss recovery mechanisms implemented in CMT (note that, since CMT is based on SCTP, CMT inherits all the built-in TCP enhancements in SCTP such as

[10] Each data point in both figures is an average of six runs, where the error bars are almost invisible.



(a)



(b)

Figure 5.4: Throughputs of two-SCTP and CMT flows for smaller n values (a) w_q = 0.002 (b) w_q = 0.001

appropriate byte counting – ABC) and CMT being more resilient to losses due to sharing of the TSN space and ACK information. We can perceive this situation to be similar to two TCP-Reno flows outperforms two TCP-Tahoe flows. CMT also incorporates a TCP-like (AIMD-based) congestion control mechanism and TCP flows can co-exist with CMT in the network (though CMT throughput is higher). Therefore, we conclude a two-homed CMT to be TCP-friendly.

5.4 Related Work

As explained in Section 4.1, the notion of TCP-friendliness emerged in the late 1990s as a reaction to the increase of non-TCP traffic in the Internet. Given the enormous economic, social, and political importance of the Internet, it is easy to justify a conservative approach, such as TCP-friendliness, to address the increasing non-TCP traffic in order not to upset the health and stability of the Internet. However, a lot has changed since the late 1990s. Newer developments (one of them being CMT) demand that we re-consider the notion of TCP-friendliness and the Internet that TCP-friendliness sets the norms. In the following sub-sections, we discuss both proposals similar to CMT and criticisms against TCP-friendliness.

5.4.1 Other CMT-like Schemes

Seminal documents from IETF and S. Floyd et. al. related to the notion of TCP-friendliness, such as [28][11], [47][12], and [55][13], discuss the appropriate level of *granularity* for a "flow" (i.e., end-to-end connection subject to congestion control). Although the IETF allows an HTTP application to open *up to two* TCP connections[14], applications opening multiple TCP connections or splitting one TCP connection into multiple TCP

[11] in Section 4.

[12] in Section 3.2.

[13] in Appendix A.

[14] Refer to Section 8.1.4 of [46].

connections (i.e., a flow granularity other than one source IP address-destination IP address pair) have been frowned upon as they get more aggressive share of the bandwidth compared to a single TCP connection. Running between a *set* of source and a *set* of destination IP addresses, and over *multiple* paths, clearly, a CMT flow does not conform to the suggested granularity of a flow in the context of TCP-friendliness.

However, other proposals, similar to CMT, such as CP [85], MulTFRC [40], mulTCP [39], MPAT [97], and PA-MulTCP [72], also aim to achieve aggregated flow rates (i.e., rates similar to aggregated rate of a group of TCP connections). Some of these proposals are based on window-based AIMD[15] mechanism, while some are based on the TCP-Friendly Rate Control (TFRC) [50, 56] protocol. TFRC uses Padhye's TCP-friendly equation [86] to adjust its sending rate. AIMD responds to every congestion indication (packet drop) by multiplicatively decreasing its sending rate. On the other hand, TFRC does not respond to a single packet loss but instead responds to the (measured) average loss rate – or *loss events* that can include one or multiple packets losses. Therefore, TFRC aims to achieve a smoother sending rate compared to window-based TCP, making TRFC more suitable for streaming or multimedia applications.

- CP (Coordination Protocol) defines "flowshare" as the rate of a *single* TCP flow and aims to obtain *multiple* flowshares. CP uses TFRC and estimates a single flowshare (i.e., the available bandwidth for a single TCP flow). Then, CP multiplies the estimated flowshare bandwidth with N to emulate an aggregated flow rate similar to N flowshares.

- Similar to CP, MulTFRC aims to emulate the behavior of N TFRC protocols for providing smoother aggregated sending rate. Unlike CP, instead of naively multiplying the TFRC rate by N, MulTFRC implicitly estimates the loss event per flow in the aggregate flow. MulTFRC extends the TCP-friendly equation[16] to support

[15] We explained AIMD earlier in Section 4.1.

[16] Details of the mulTFRC equation is given at [41].

multiple TCP flows and uses estimated *per aggregate-flow loss rate* in the equation. It is shown that MulTFRC produces smoother sending rates than CP [40].

- MPAT is based on mulTCP [39] which in turn is based on AIMD. MulTCP takes N as an input parameter and aims to behave like N TCP flows. Standard TCP uses the AIMD(a=1, b=1/2) algorithm. That is, if there is a sign of congestion, the congestion window (cwnd) is decreased by b=1/2 of the current congestion window value, while cwnd is increased by a=1 in every RTT if there is no congestion (during steady-state). MulTCP assigns AIMD(a=N, b=1/2N) to the aggregate flow to emulate N TCP flows. However, it is shown in [97] that the loss rate experienced by mulTCP ends up being smaller than the loss rate experienced by N TCP flows. This makes mulTCP to behave more aggressively than N TCP flows, especially as N grows. MPAT is proposed to provided better fairness than mulTCP. MPAT maintains congestion control states as many as the number of flows it manages. Therefore, as N grows, the overhead of MPAT increases.

- Like MPAT, PA-mulTCP is also based on mulTCP. However, unlike MPAT, PA-mulTCP maintains a single congestion window state for the entire aggregate flow (which reduces the overhead) and yet achieves fairer aggregated flow than mulTCP. PA-mulTCP adds an additional probe window to detect the sending rate of a real TCP connection and uses this rate to adjust the rate of the aggregated flow.

In addition to the proposals above, in the *tsv working group*, IETF has started the Multipath-TCP (MPTCP) activity [8]. Similar to SCTP and SCTP-based CMT, MPTCP aims to achieve TCP connections over multiple paths (IP addresses), an ability to move data traffic to a less congested path when needed, or to use multiple paths simultaneously to utilize available capacity in the network.

5.4.2 Criticism against TCP-Friendliness

The most explicit and blunt criticism on TCP-friendliness came from B. Briscoe starting in his controversial and seminal paper "Flow-Rate Fairness: Dismantling a Religion" [31] in 2007. Briscoe, referring TCP-friendliness as *flow-rate fairness*, instead proposed what he called *cost fairness*. Considering social, real-world, and economic examples, cost fairness (which takes its roots from Kelly's work on weighted proportional fairness [71]) is a more realistic measure of fairness than flow-rate fairness. Briscoe refuses the dogma that *equal flow rates are fair*. Instead, in a system where cost fairness is established, each "economic entity" would be accountable for the costs they caused to others. Cost fairness allocates cost to *bits* instead of flows; hence, cost fairness is immune to the problems such as splitting flow identifiers or opening multiple connections as flow-rate fairness is. Representing the viewpoint of flow-rate fairness, S. Floyd and M. Allmann replied with a "rebuttal" [48] not only stating the usefulness of flow-rate fairness but also accepting the limitations of flow-rate fairness. Following Briscoe, M. Mathis published an Internet draft [79] arguing that we have to *rethink* the notion TCP-friendliness to keep up with an evolving Internet.

The views from both sides, one clinging on to the flow-rate fairness and the other asking flow-rate fairness to be dismantled, are now being heavily discussed in the IETF mailing lists such as end2end-interest, iccrg, tsvwg, and tmrg. In addition, workshops such as [10] discuss the compelling reasons to replace or renew TCP and its congestion control algorithms. Moreover, bigger research activities and agendas that will change and redesign the entire Internet architecture are underway, such as [6, 7, 9, 11]. We are going towards a world where TCP and TCP-friendliness might not set the standards any longer. However, this author believes that it will be a while before any other view becomes an alternative or displaces TCP-friendliness.

5.5 Summary and Conclusions

In this chapter, we designed an experimental framework to investigate TCP-friendliness of CMT. Via QualNet simulations, we measured the sending rate of a two-homed CMT and two SCTP flows and the impact of CMT and two SCTP flows on the other TCP flows in a network while sharing a tight link. We found out that a two-homed CMT association has similar or worse performance (for smaller number of competing TCP flows) than the aggregated performance of two independent, single-homed SCTP associations while sharing the tight link with other TCP connections, due to the fact that a CMT flow creates burstier data traffic than two independent SCTP flows. When compared to the aggregated performance of two-independent TCP connections, two-homed CMT obtains higher share of the tight link bandwidth because of better loss recovery mechanisms in CMT (as CMT inherits all the built-in TCP enhancements in SCTP). In addition, sharing ACK information makes CMT more resilient to losses [60]. Although CMT obtains higher throughput than two independent TCP flows, CMT's AIMD-based congestion control mechanism allows other TCP flows to co-exist in the network. Therefore, we conclude that CMT to be TCP-friendly. We also surveyed and presented other (multipath and alike) research efforts similar to CMT and the recent activities that questioned the very foundation, assumptions, and goals of the TCP-friendliness doctrine. Our survey shows that CMT is not the only transport layer protocol that does not directly fit into the current TCP-friendliness doctrine and hence TCP-friendliness and its assumptions should be re-considered to include newer developments such as multihoming.

Chapter 6

SCTP QUALNET SIMULATION MODULE

In this chapter we describe the SCTP module (which is used throughout the simulation studies in this dissertation) developed for the QualNet network simulator [21].

6.1 Introduction

Simulation is a widely used methodology for performance evaluation in networking research. In particular, it is crucial to use the high-fidelity wireless physical layer models in simulation studies to correctly evaluate the performance/behavior of higher layer protocols of wireless networks [102]. Since the QualNet network simulator [21] supports better wireless physical layer models than the other network simulators such as ns-2 [4], we designed and developed an SCTP simulation module for QualNet the simulator in the NSF WHYNET project [5]. The first official release of the SCTP module is designed for QualNet 3.6.1 (October 2004) and the latest official release (release 2.0) is for QualNet 3.9 (April 2006)[1]. The official SCTP QualNet module can be downloaded from the WHYNET website at [5] or requested from the DEGAS Networking group SCTP web site at [21].

In the following subsections, we briefly describe the implementation of the SCTP QualNet module and its use within the networking community.

[1] As of today, the SCTP QualNet module has been ported to QualNet 4.5.1. Therefore, the next official release of the SCTP QualNet module will be released in QualNet 4.5.1 or higher.

6.2 Implementation

A comprehensive SCTP simulation model using the QualNet simulator has been developed in steps. Initially, we developed the standard SCTP protocol with both (1) the multi-homing and (2) the multi-streaming features by following RFC 2960 and the SCTP Implementer's Guide `draft-ietf-tsvwg-sctpimpguide-10.txt`[2]. Later, we added (3) the Dynamic Address Reconfiguration (DAR) extension of SCTP into the module by following `draft-ietf-tsvwg-addip-sctp-10.txt`[3]. We have also incorporated (4) the Partial Reliability Extension of SCTP (i.e., PR-SCTP) into the module by following RFC 3758. Our latest official release (release 2.0) includes all the components (1-4). Finally, we have added the CMT algorithms and retransmission policies [63] into the SCTP QualNet module[4].

For details of the user guide to install and use the SCTP QualNet module, as well as its implementation details, please visit the DEGAS SCTP QualNet module web site at [21].

6.2.1 Comparing the SCTP QualNet and the ns-2 Modules

To validate the correctness of our SCTP QualNet implementation, we cross-checked with the SCTP ns-2 simulation module [32] using a *wired* networking scenario. We have repeated a subset of the CMT over wired networking simulations conducted using the SCTP ns-2 simulation module (as presented in [63]) by using the SCTP QualNet simulation module[5]. The simulation setup and topology are shown in Figure 6.1. An 8 MB file is transferred from node A to node B using the CMT schemes by varying simulation

[2] Note that, the SCTP Implementer's Guide had become RFC 4460. RFC 2960 had also been obsoleted by RFC 4960, which includes the errata and issues in RFC 4460.

[3] Note that the DAR extension is now RFC 5061.

[4] Therefore, the next official SCTP QualNet release will include CMT as well.

[5] For the validation study, we used svn revision 1 of SCTP Module in QualNet 4.5.1. We believe that the simulations in [63] mainly use ns-2.31.

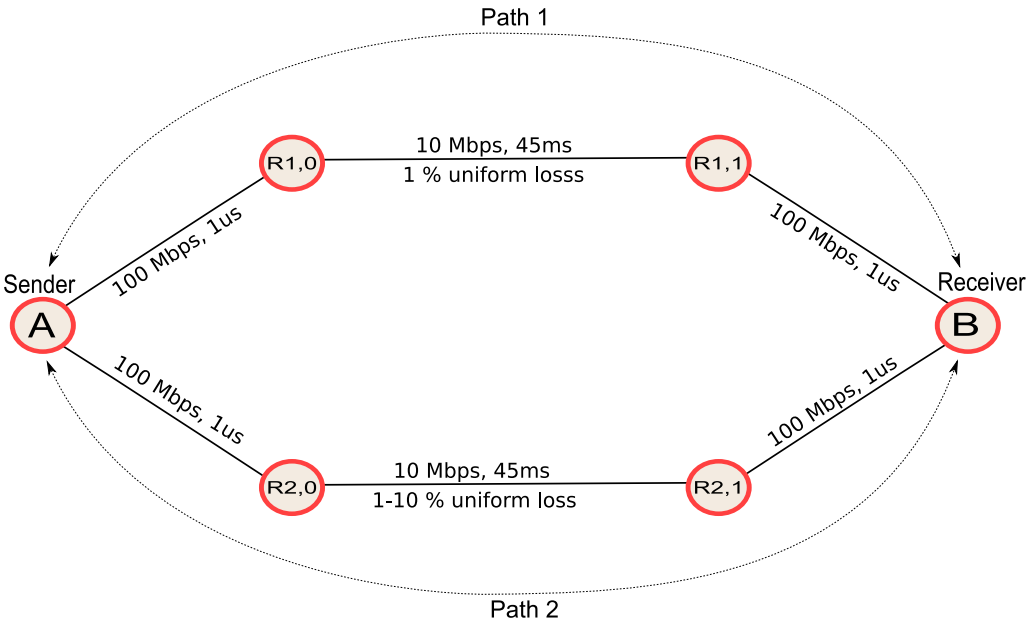


Figure 6.1: Wired simulation topology for evaluating SCTP CMT scheme

parameters (such as rBuf size, CMT RTX scheme, and the loss rate on path 2)[6]. In Figure 6.2 and 6.3, we annotated the QualNet results obtained with the ns-2 results from [63] for unlimited rBuf and rBuf = 32 KB, respectively. In the figures, the red (solid) lines show the ns-2 results extracted from [63] and the blue (dotted) lines show the QualNet simulation results we obtained. The next sub-section discusses the results.

6.2.1.1 Discussion

Figures 6.2 and 6.3 show that our SCTP CMT QualNet implementation is compatible with the SCTP CMT ns-2 implementation. We observe the following comparing the SCTP CMT QualNet and SCTP CMT ns-2 results.

- Although the absolute values of data points in the graphs between QualNet and ns-2 implementations differ, the performance trends are similar. For instance, for

[6] For further details of the simulation set up, please refer to [63] and [25].

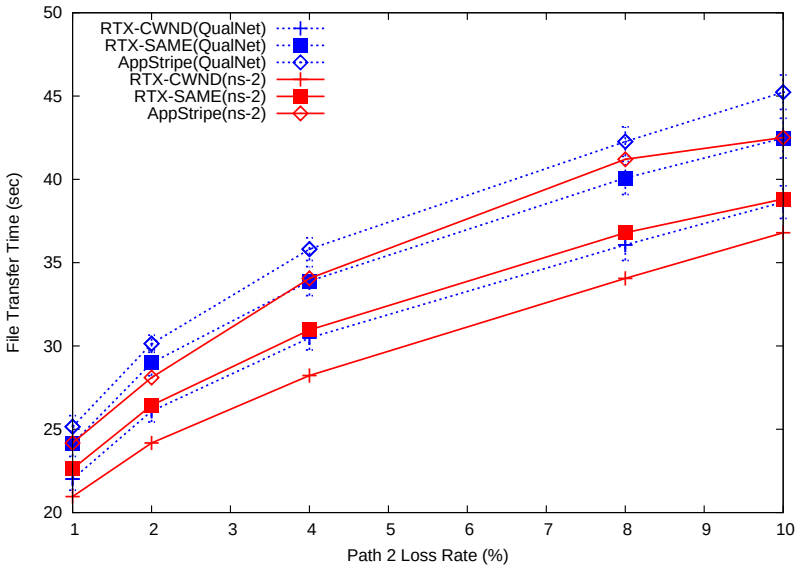


Figure 6.2: AppStripe vs. CMT with unlimited rBuf – Comparing the ns-2 results in Figure 11(a) of [63] with QualNet results

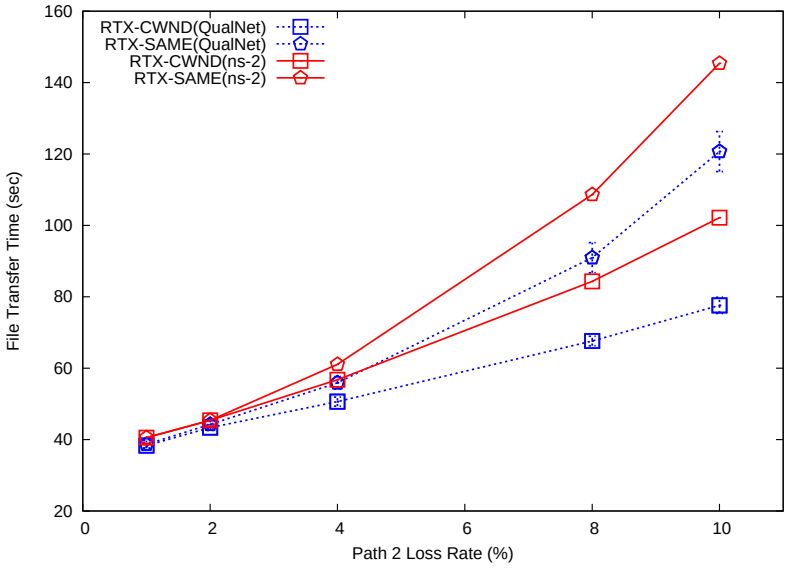


Figure 6.3: Performance of the two of the CMT RTX policies for rBuf = 32 KB – Comparing the ns-2 results in Figure 13(a) of [63] with QualNet results

the unlimited rBuf case (Figure 6.2) the file transfer time of AppStripe is larger compared to the file transfer time with CMT for both QualNet and ns-2. In Figure 6.3, both QualNet and ns-2 results show that CMT with RTX-CWND outperforms CMT with RTX-SAME especially for higher loss rates and limited rBuf (i.e., the file transfer time of CMT using RTX-CWND is smaller compared to CMT using RTX-SAME).

- We also observed that the absolute values of the data points (or the curves) of ns-2 are lower than those of QualNet for the unlimited rBuf case (Figure 6.2). However, for the limited rBuf case (Figure 6.3), QualNet values (curves) are lower than the ns-2 values. Note that, the impact of RTX-CWND on the performance of the CMT associations is better observed for the limited rBuf and higher loss rates. We suspect that the slight implementation differences between the SCTP QualNet and the ns-2 modules might have caused the absolute value differences[7]. In addition, since we did not run the ns-2 simulations in person, we do not know all the network parameter values used in the ns-2 simulations (such as the queue lengths in the routers).

6.3 Community Interest

While we are using the SCTP QualNet module for our own research [24, 23, 22, 26], the networking community's interest in the module is also on the rise. According to download records, there are together 40+ downloads from the WHYNET web site [5] (since November 2004) and the DEGAS SCTP web site [21] (since January 2006). Some of the universities and institutions downloaded the SCTP QualNet module are listed in Table 6.1.

[7] While implementing the SCTP QualNet module, we also examined the SCTP ns-2.31 code. We had observed and noted some slight implementation differences between SCTP QualNet and ns-2.31 modules, which are presented in Appendix D.

6.4 Conclusion

Overall, we believe that the SCTP QualNet module is a comprehensive implementation of the SCTP protocol and its extensions. Currently, the SCTP QualNet module is available as a patch to the official QualNet simulator package. In the near future, we plan to merge and distribute the module within the official QualNet simulator releases.

UC Berkeley; Hanyang University, South Korea; Yonsei University, Seoul, South Korea; Dept. of Computer Science, DR M.G.R Deemed University, India; College of Logistics Engineering, Wuhan University of Technology, China; University Visvesvaraya College of Engineering, India; The Department of Electronic and Information Engineering of HUST, China; Italtel, Italy; Iran University of Science and Technology; Dept. of Computer Science, UT-Dallas; Korea University, Seoul, Korea; Altsoft Inc.; Veermata Jijabai Technological Institute, Computer Technology, India; King's College London, United Kingdom; San Diego Research Center Inc., USA; Sejong University, Republic of Korea; Keio University, Japan; National Cheng Kung University (NCKU), Taiwan; Anna University, India; Dept. of Computer Science, University College Dublin; Dept. of Computer Science and Info. Technology, Tezpur University, India; Universidade Federal de Campina Grande (UFCG), Brazil; Dept. of Electrical Engineering, Nation Taiwan University; Warsaw School of Technologies, Poland; Centre for Telecommunications Research, King's College London, University of London (Feb'06); Computer Science and Informatics, University College Dublin (Feb'06); College of Automation, Northwesten Polytechnical University (March'06); Dept. of Computer Science and Information Engineering, National Taiwan University (April'06); Department of Information Science, Gyeongsang National University, Korea (May'06); Istanbul Technical University, Turkey (May'06); Tohoku University, Japan (June'06); Electrical and Computer Engineering, University of Victoria, Canada (July'06); Department of Computer Science, Karachi University, Pakistan (Jan'07); Comsats Institute of Information Technology (CIIT), Pakistan (March'07); Hiroshima City University (Aug'07); Department of Electronic Engineering, National Taiwan University of Science and Technology (Nov'07); Jawaharlal Nenru University, New Delhi, India (June'08); Nokia Siemens Networks(NSN), Du-dubai Project in United Arab Emirates (July'08); E'cole Polytechnique de Montre'al, Laboratory: LARIM (Laboratoire de Recherche en Re'sautique et Informatique) (Aug'08); Department of Engineering Science, National Cheng Kung University (Sept'08); Dept. of Information and Communication Technology, Manipal Institute of Technology, India (Jan'09); Universidad Te'cnica Federico Santa Mari'a en Chile, Departamento de Informa'tica (Feb'09); Mehran University of Engineering and Technology, Pakistan (Feb'09); Universidade Federal de Santa Catarina, Brazil (March'09); Institute for Information Industry, Taiwan (June'09); National Institute of Technology Kurukshetra, India (July'09); Dept. of Computer Science, SUNY Binghamton, US (July'09); Dept of Computer Science, B.S.Abdur Rahman University, India (March'10).

Table 6.1: Some of the SCTP QualNet module downloads since November 2004.

Chapter 7

CONCLUSIONS AND FUTURE WORK

In this dissertation, we studied SCTP-based CMT in two contexts: (1) multihop wireless networks (MWNs) and (2) TCP-friendliness. We summarize our contributions and future research directions in the following two sub-sections.

7.1 CMT over MWNs

We studied two related problems in the context of CMT over MWNs. In the first problem (Chapter 2), we examined the performance of CMT over IEEE 802.11 based MWNs using both limited and unlimited receiver buffer (rBuf) size. We found out that similar to the wired cases, when rBuf is not constrained, CMT over MWNs shows better performance than one single-homed SCTP association and even the ideal AppStripe application. We suggest that considering the bandwidth limitations of MWNs compared to their wired counterparts, applications will benefit from using CMT with a reasonably large rBuf size over MWNs.

In the second problem (Chapter 3), we studied the ACK mechanism of CMT to mitigate the effects of self-contention between T-DATA and T-ACK packets by exploiting the availability of multiple (return) paths. The original CMT sends a T-ACK packet to the latest path the DATA packet arrived from (i.e., potentially to any one of the return paths). We explored another ACK policy, where ACK packets are always sent to the same (return) path during the association lifetime. We found out that especially as the multihop wireless path gets longer and the number of simultaneous CMT sub-flows increases, the alternative

ACK policy shows better performance, even when the quality of the paths differ. We showed that the better performance of the alternative ACK policy is due to the fact that the inter-flow interference (between the DATA and the ACK packets) within a CMT flow using the alternative ACK policy is less than the inter-flow interference within a CMT flow with the original ACK policy. This situation causes the aggregated sending rate of the CMT flow using the alternative ACK policy to become higher than the aggregated sending rate of the CMT flow using the original CMT ACK policy.

Our future research is to further improve the performance of CMT over MWNs by adapting techniques of improving (single-path) TCP performance over MWNs (such as data pacing [45, 59], congestion window limiting [35, 36, 70, 74] and ACK thinning [19, 34, 42, 45, 49, 96]) to exploit the availability of multiple paths between the source and the destination hosts.

7.2 TCP-Friendliness with CMT

TCP-friendliness in the Internet has been traditionally studied in the context of single-path or single-homed transport connections. We designed an experimental framework to investigate TCP-friendliness of CMT, which, unlike standard TCP, uses multiple paths simultaneously. In our experimental framework, we first explored TCP-friendliness of single-homed SCTP (Chapter 4). We showed that although SCTP's congestion control mechanisms are intended to "be similar to" TCP's, being a newer protocol, SCTP has already incorporated some of TCP's enhancements. Therefore, SCTP obtains higher share of the bandwidth when competing with TCP that does not have similar enhancements. We conclude that SCTP is TCP-friendly but achieves higher throughput than TCP, due to SCTP's better loss recovery mechanisms [15, 82, 83, 84], just as TCP-SACK or TCP-Reno outperforms TCP-Tahoe.

In Chapter 5, we investigated the TCP-friendliness of CMT. We measured the sending rate of one two-homed CMT flow and two SCTP flows, and also the impact of CMT and two SCTP flows on the other TCP flows in the network while sharing a

tight link. We found out that while sharing a tight link with other TCP flows, CMT's performance is similar or worse than two SCTP flows mainly because of the burstier data traffic that CMT creates compared to two SCTP flows. We also discovered that one two-homed CMT flow obtains higher share of the tight link bandwidth compared to two TCP flows, because of better loss recovery mechanisms in CMT (as CMT inherits built-in TCP enhancements in SCTP). In addition, sharing of ACK information makes CMT more resilient to losses [60]. Although CMT obtains higher throughput than two independent TCP flows, CMT's AIMD-based congestion control mechanism allows other TCP flows to co-exist in the network. We conclude that CMT to be TCP-friendly, just as two TCP-Reno flows are TCP-friendly compared to two TCP-Tahoe flows.

The experimental framework designed in this dissertation can be extended to have more rigorous study of TCP-friendliness of both single-homed SCTP and CMT. We expect to obtain more insights by investigating (i) the increase in the number of SCTP and CMT flows in the network, (ii) the increase in the number of CMT subflows (hence, concurrency of one CMT flow), (iii) the impact of asymmetric RTTs and edge links, and (iv) the existence of unresponsive flow (similar to UDP) and short-lived TCP flows in the background traffic similar to the testing suite in [20]. In addition to simulations, it will be worth developing the experimental framework in a network emulator to work with SCTP and CMT kernel implementations.

Our final word on TCP-friendliness of CMT is that although this dissertation investigates the TCP-friendliness of CMT in accordance with the current TCP-friendliness doctrine, we witness hot debates in IETF that questioned the very foundation of the TCP-friendly Internet. We argue that multihoming and CMT are two of the developments that support a research agenda to pursue alternative fairness criteria for the Internet.

Appendix A

TWO-TCP CASE: INDIVIDUAL FLOW THROUGHPUTS (FOR CHAPTER 5)

Note that, the first two bars in each of the figures in this appendix are for TCP1 and TCP2, respectively. The remaining bars are for each of the other *n* TCP flows.

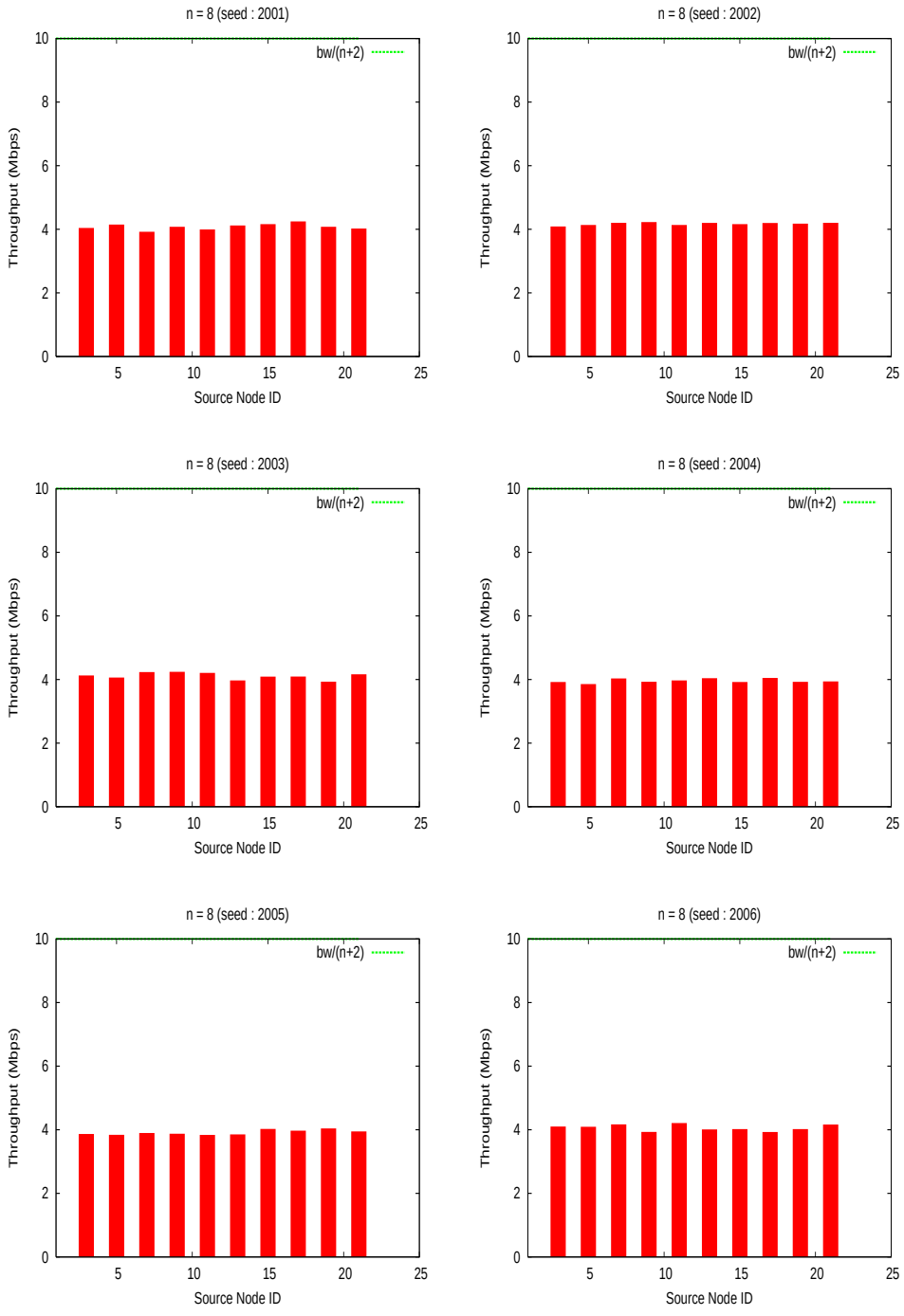


Figure A.1: Per flow throughput for two-TCP case (n:8, seeds: 2001-2006)

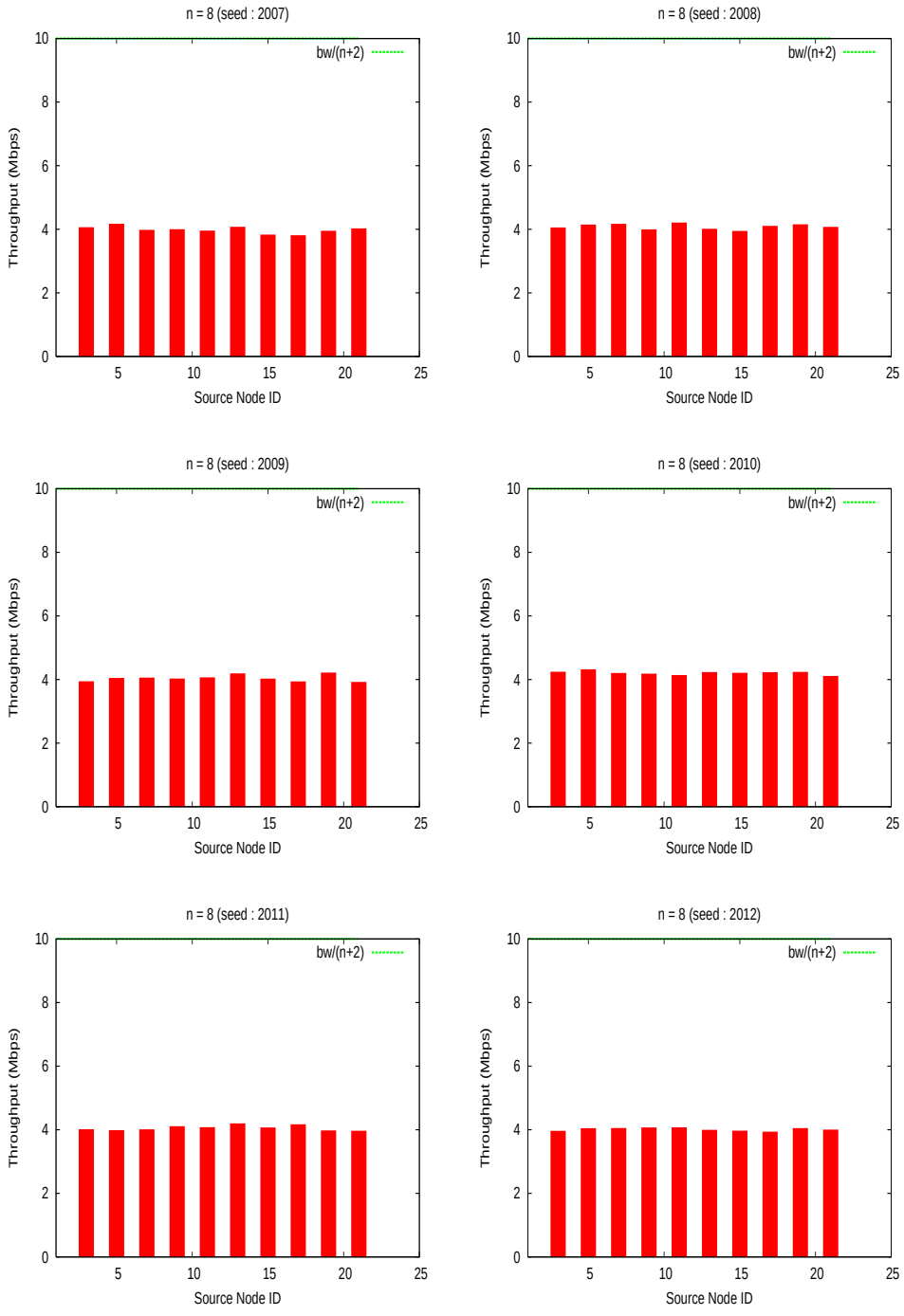


Figure A.2: Per flow throughput for two-TCP case (n:8, seeds: 2007-2012)

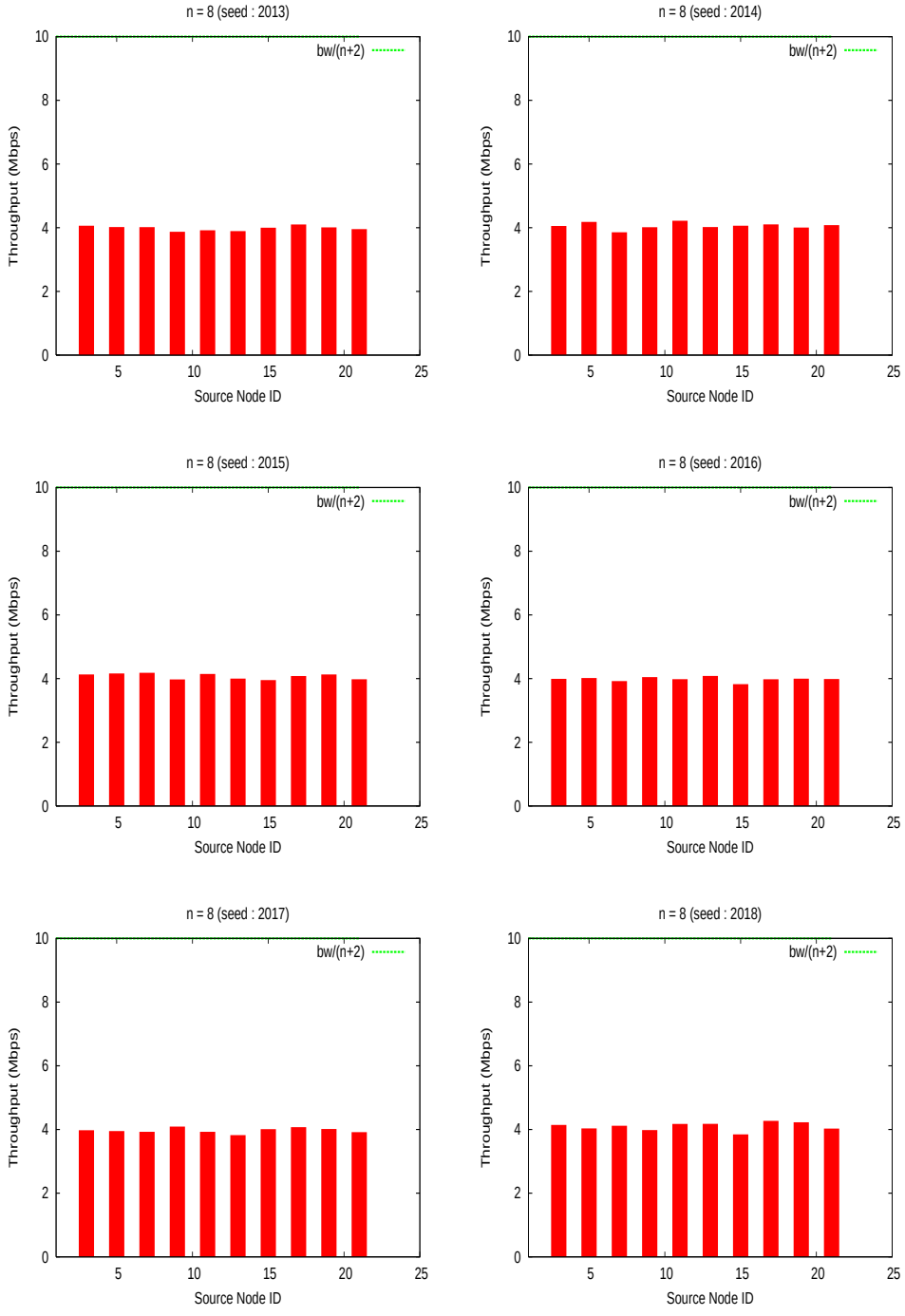


Figure A.3: Per flow throughput for two-TCP case (n:8, seeds: 2013-2018)

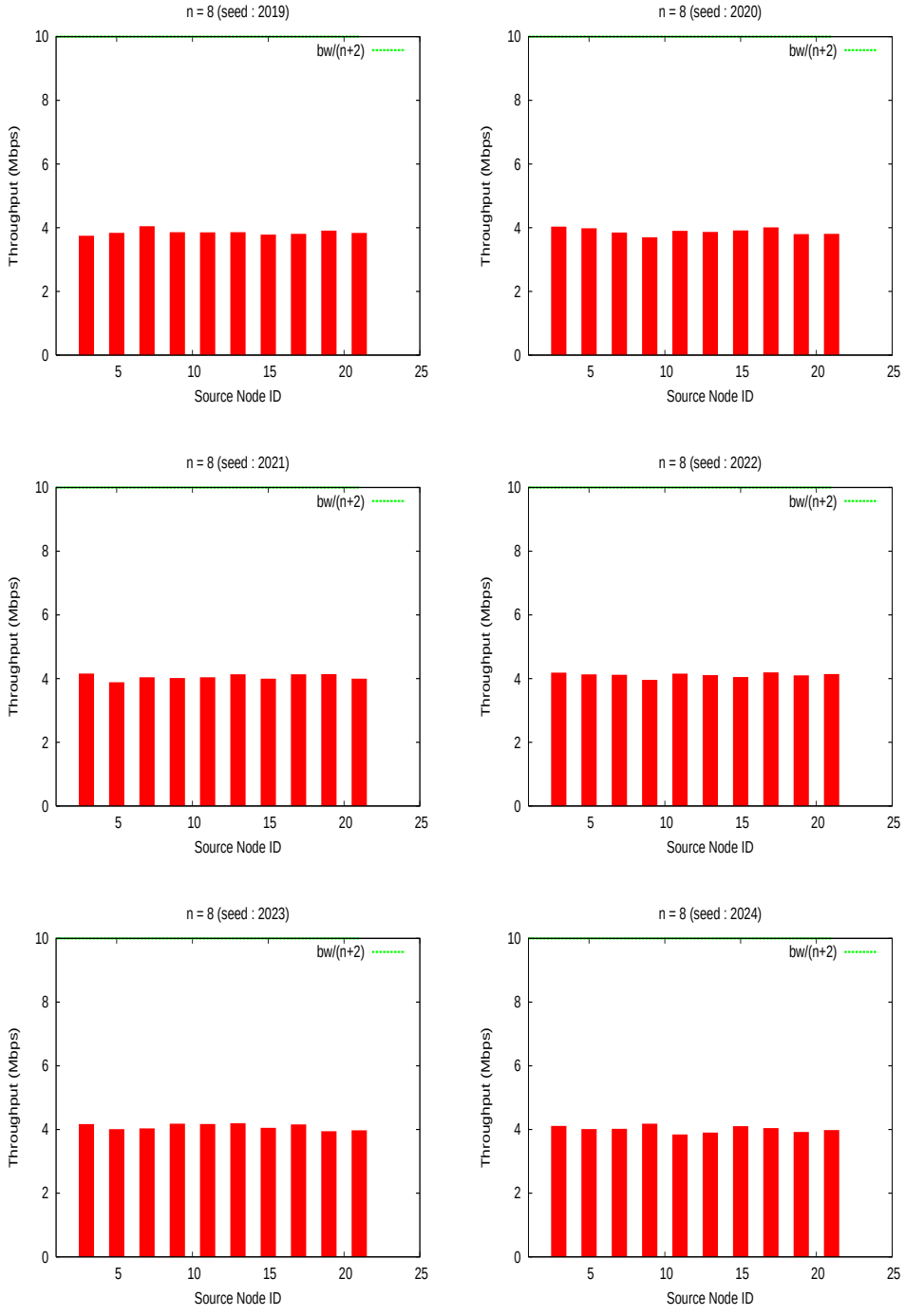


Figure A.4: Per flow throughput for two-TCP case (n:8, seeds: 2019-2024)

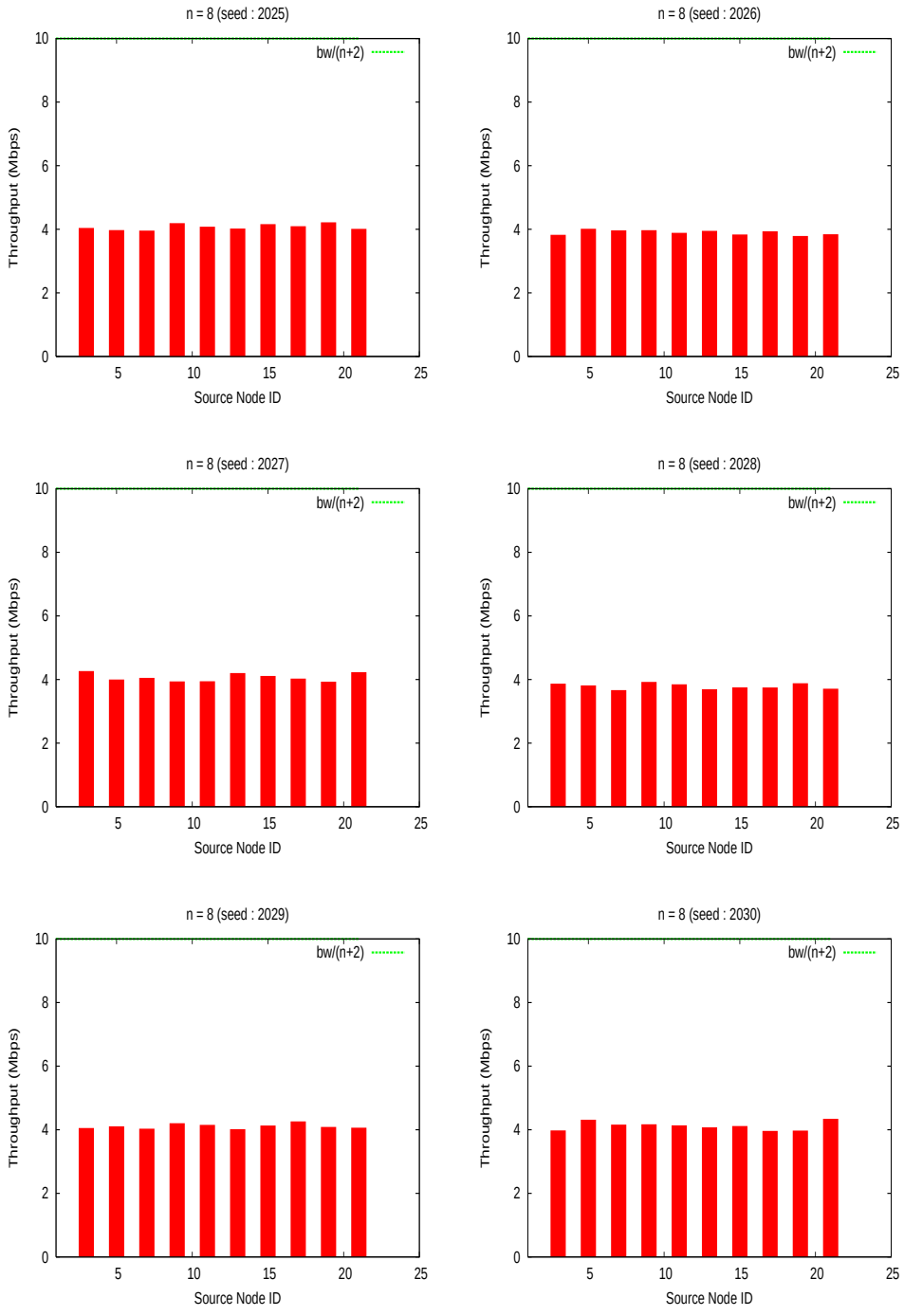


Figure A.5: Per flow throughput for two-TCP case (n:8, seeds: 2025-2030)

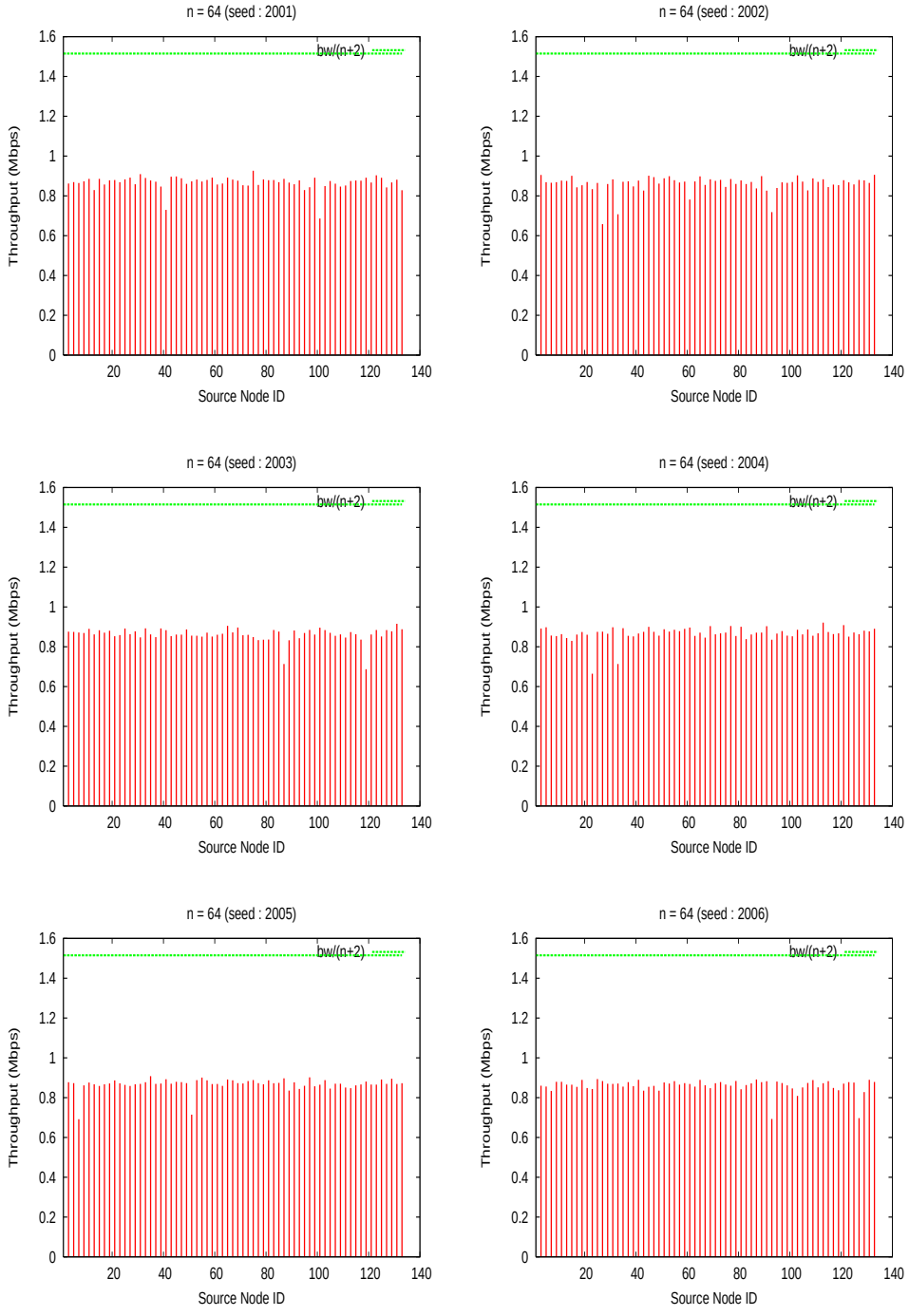


Figure A.6: Per flow throughput for two-TCP case (n:64, seeds: 2001-2006)

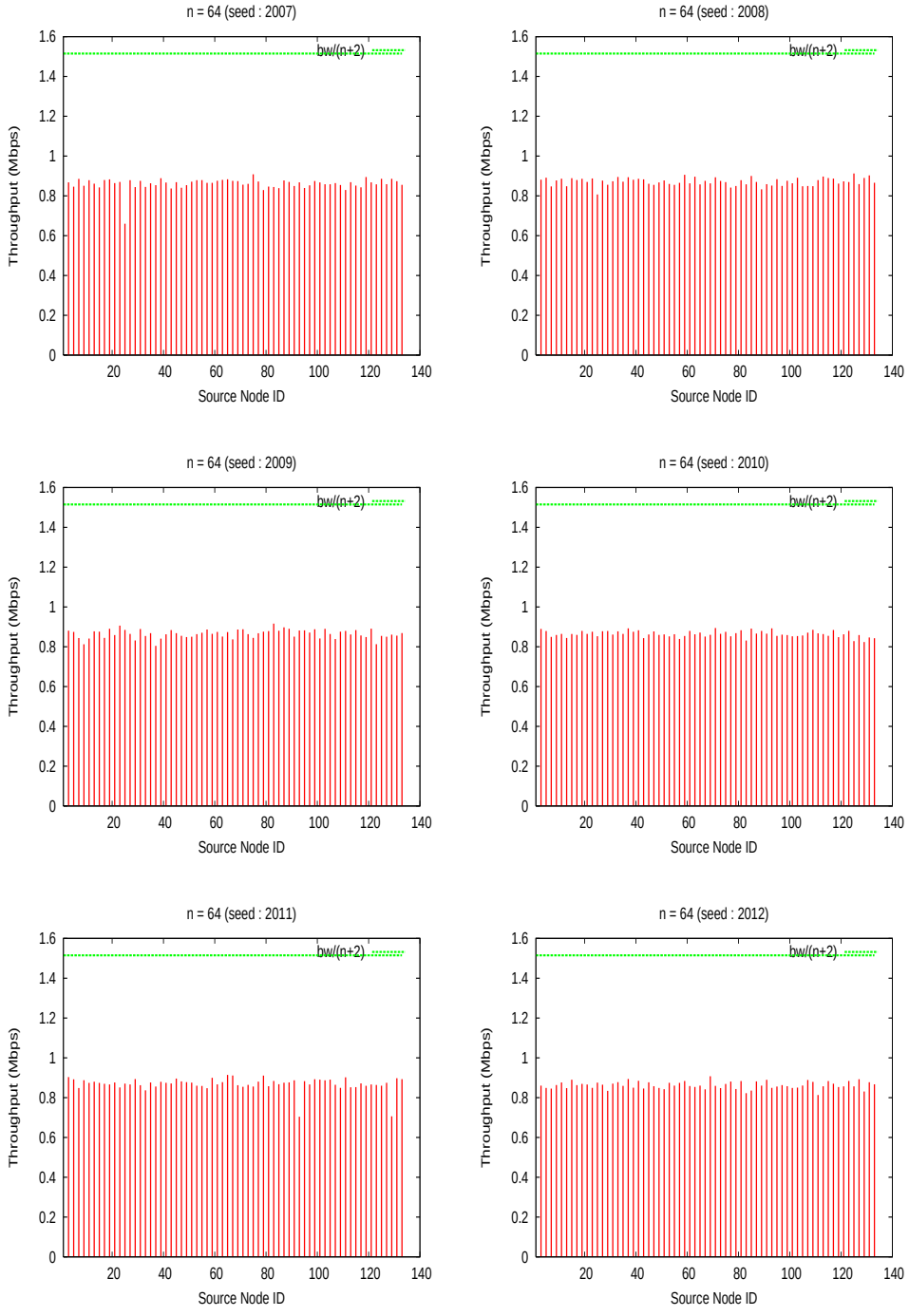


Figure A.7: Per flow throughput for two-TCP case (n:64, seeds: 2007-2012)

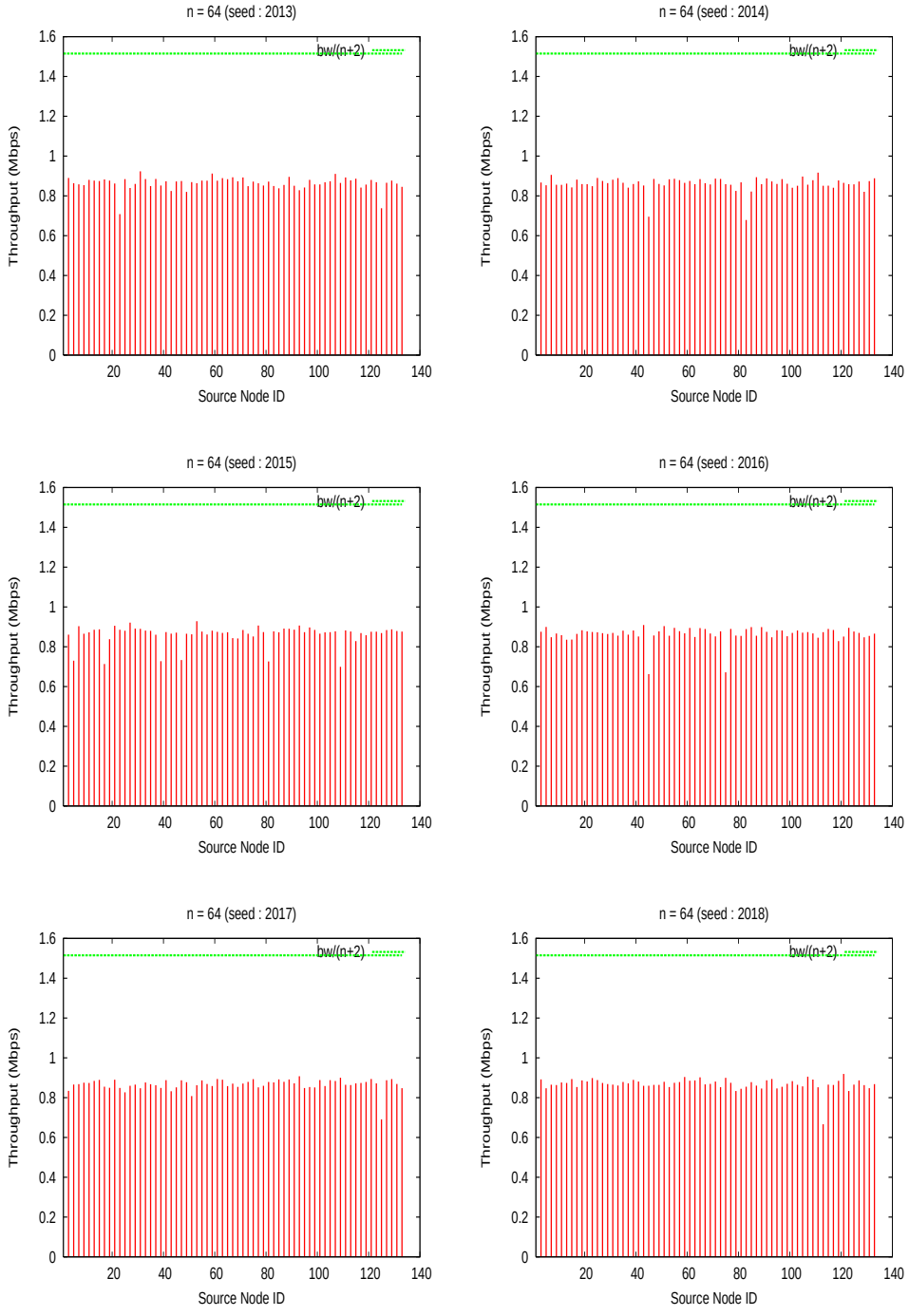


Figure A.8: Per flow throughput for two-TCP case (n:64, seeds: 2013-2018)

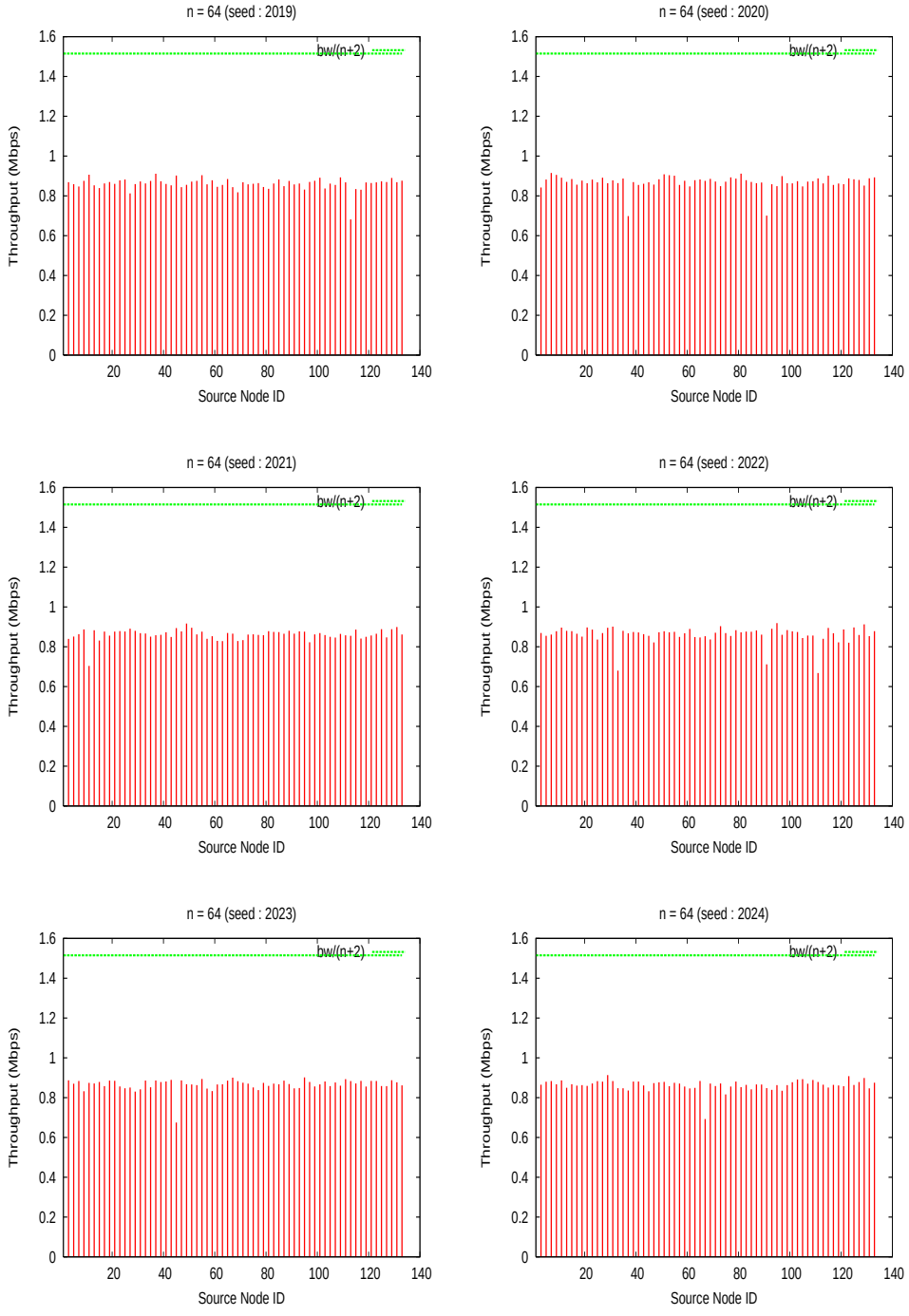


Figure A.9: Per flow throughput for two-TCP case (n:64, seeds: 2019-2024)

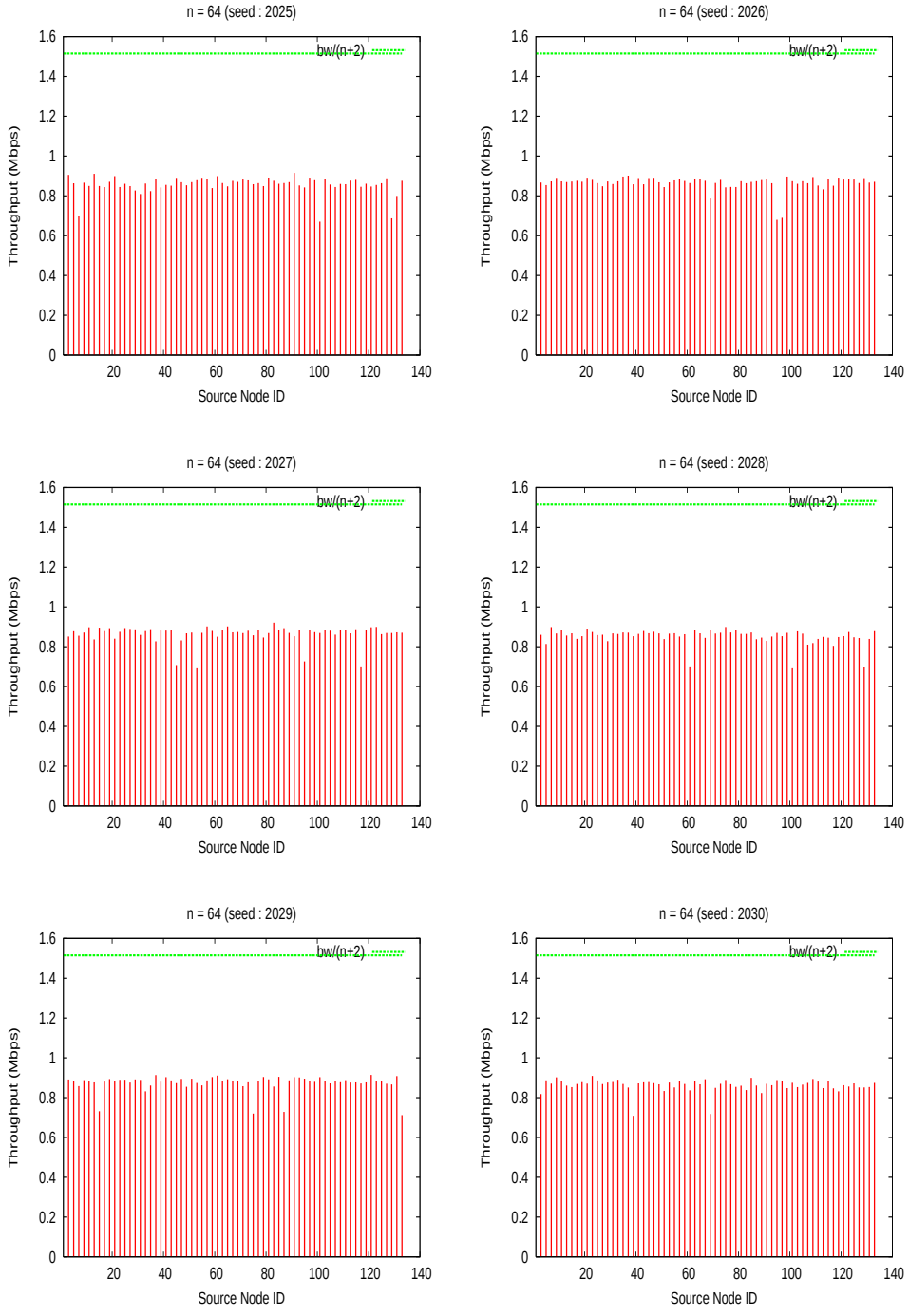


Figure A.10: Per flow throughput for two-TCP case (n:64, seeds: 2025-2030)

Appendix B

TWO-SCTP CASE: INDIVIDUAL FLOW THROUGHPUTS (FOR CHAPTER 5)

Note that, the first two bars in each of the figures in this appendix are for SCTP1 and SCTP2, respectively. The remaining bars are for each of the *n* TCP flows.

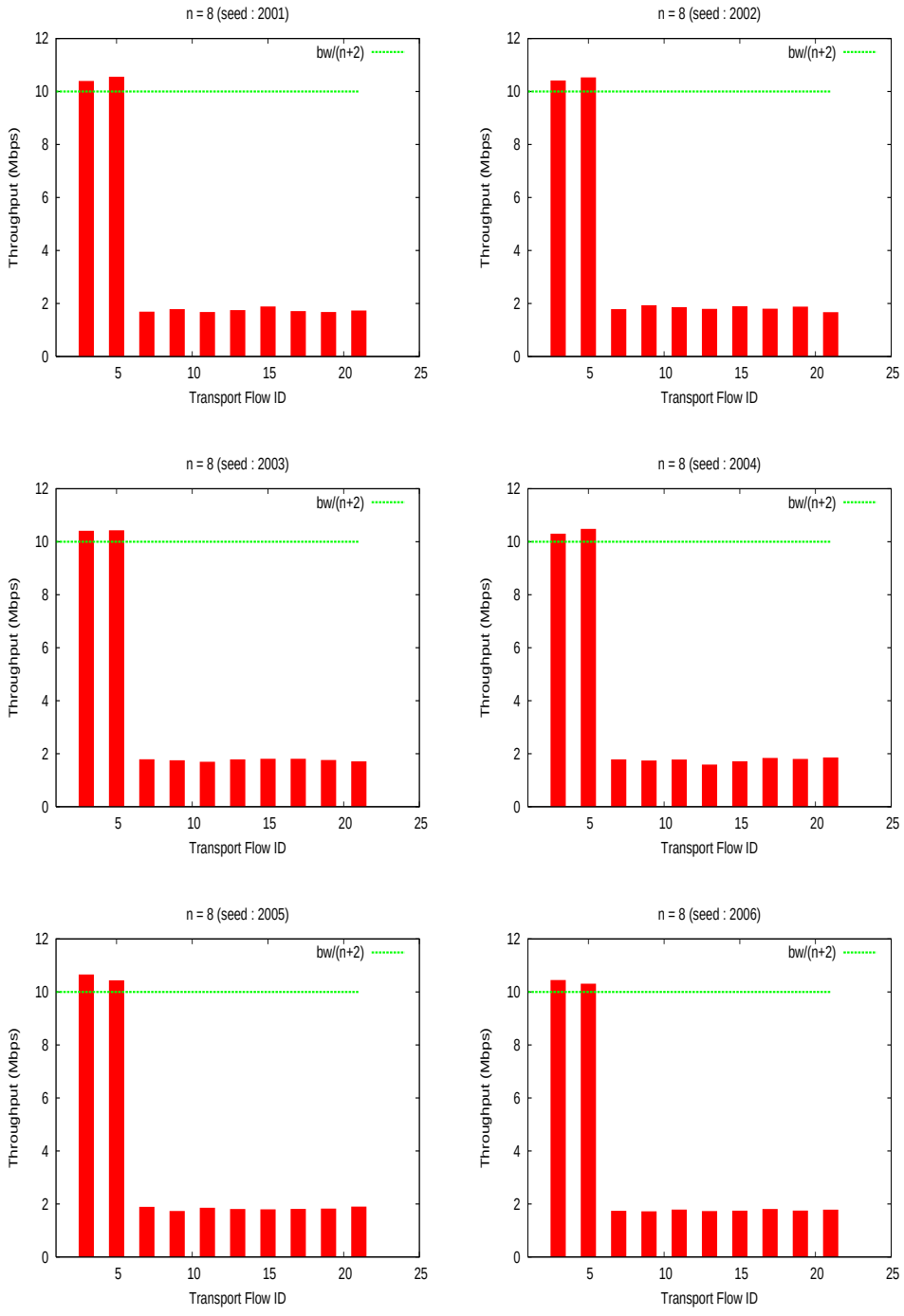


Figure B.1: Per flow throughput for two-SCTP case (n:8, seeds: 2001-2006)

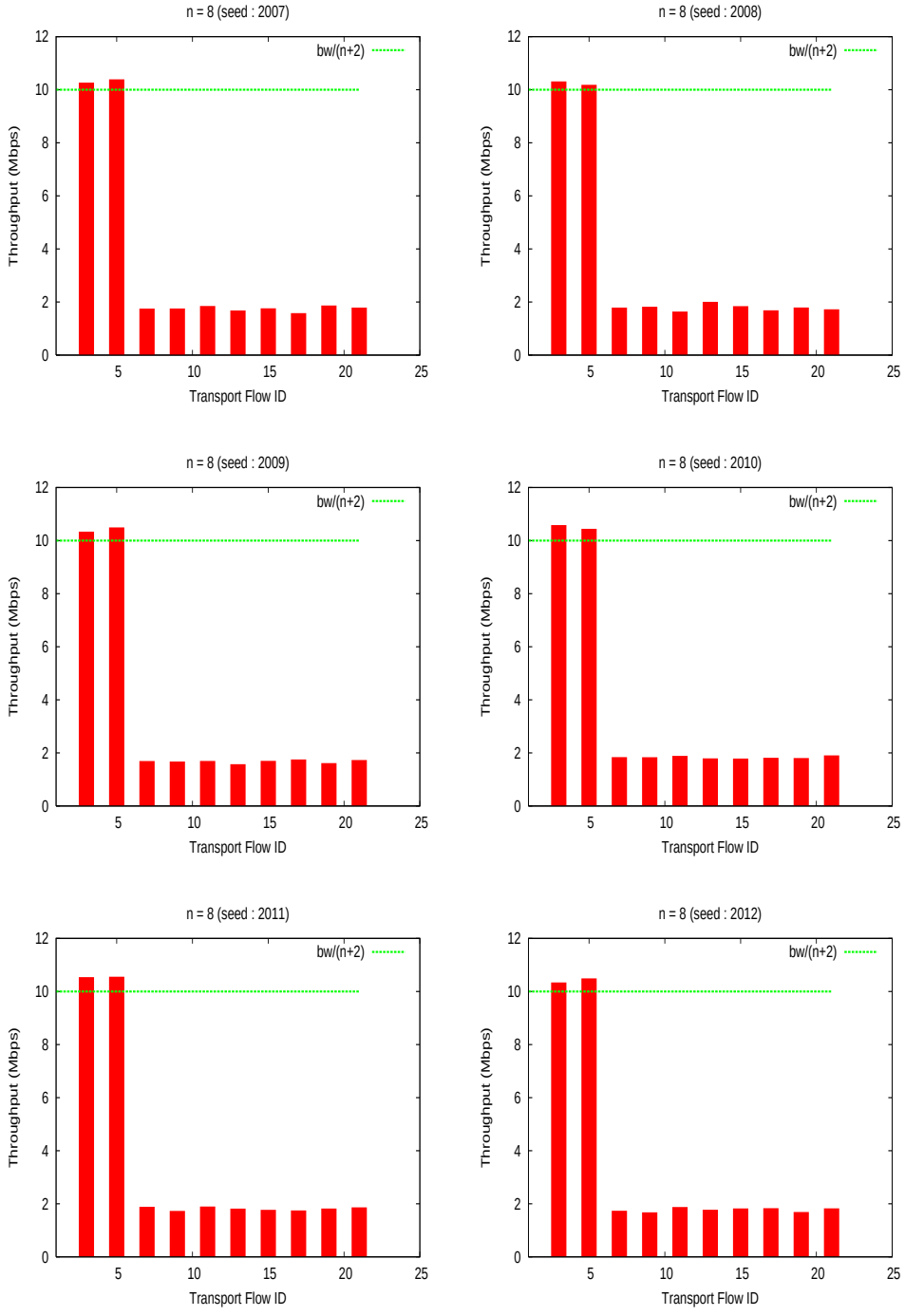


Figure B.2: Per flow throughput for two-SCTP case (n:8, seeds: 2007-2012)

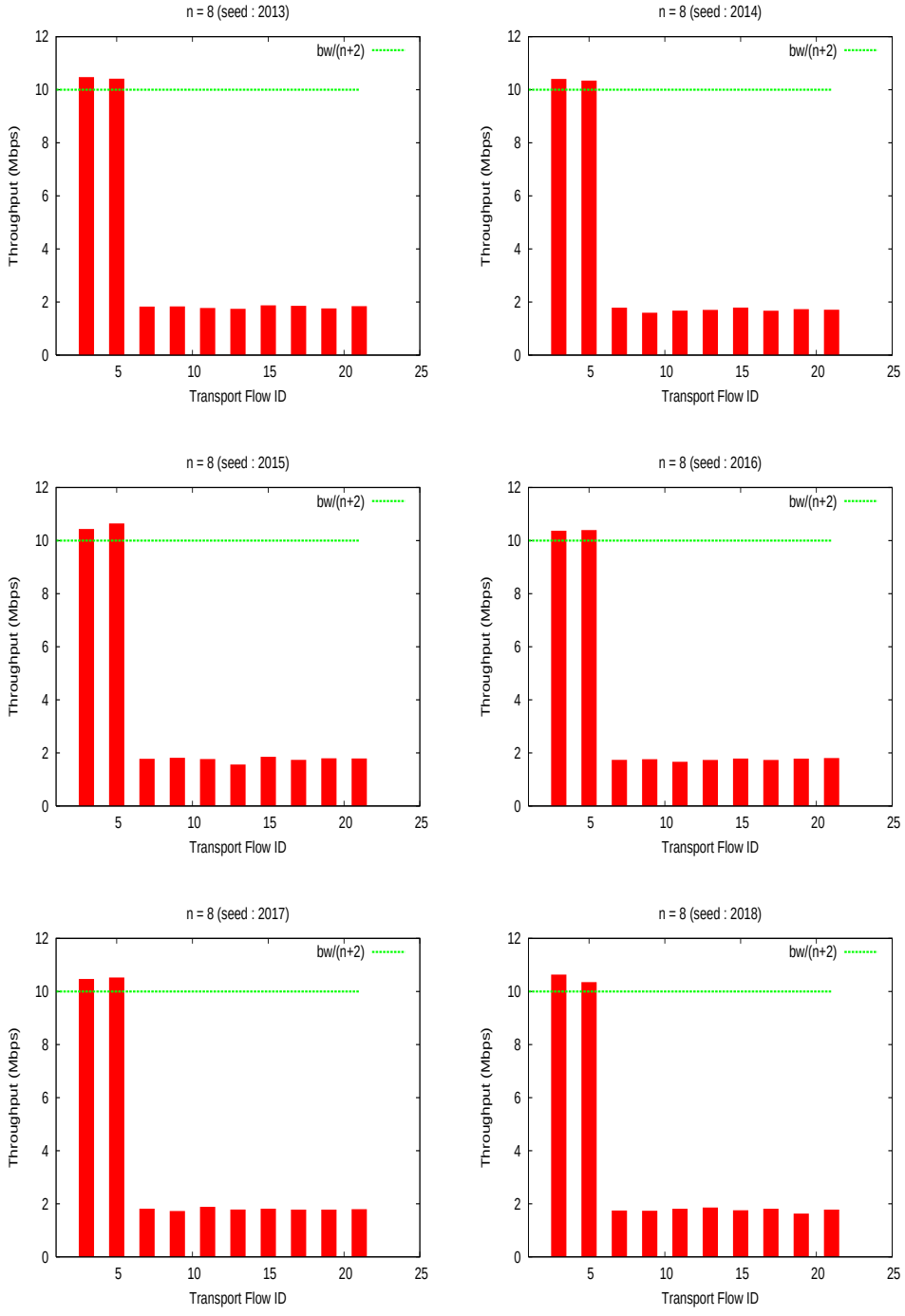


Figure B.3: Per flow throughput for two-SCTP case (n:8, seeds: 2013-2018)

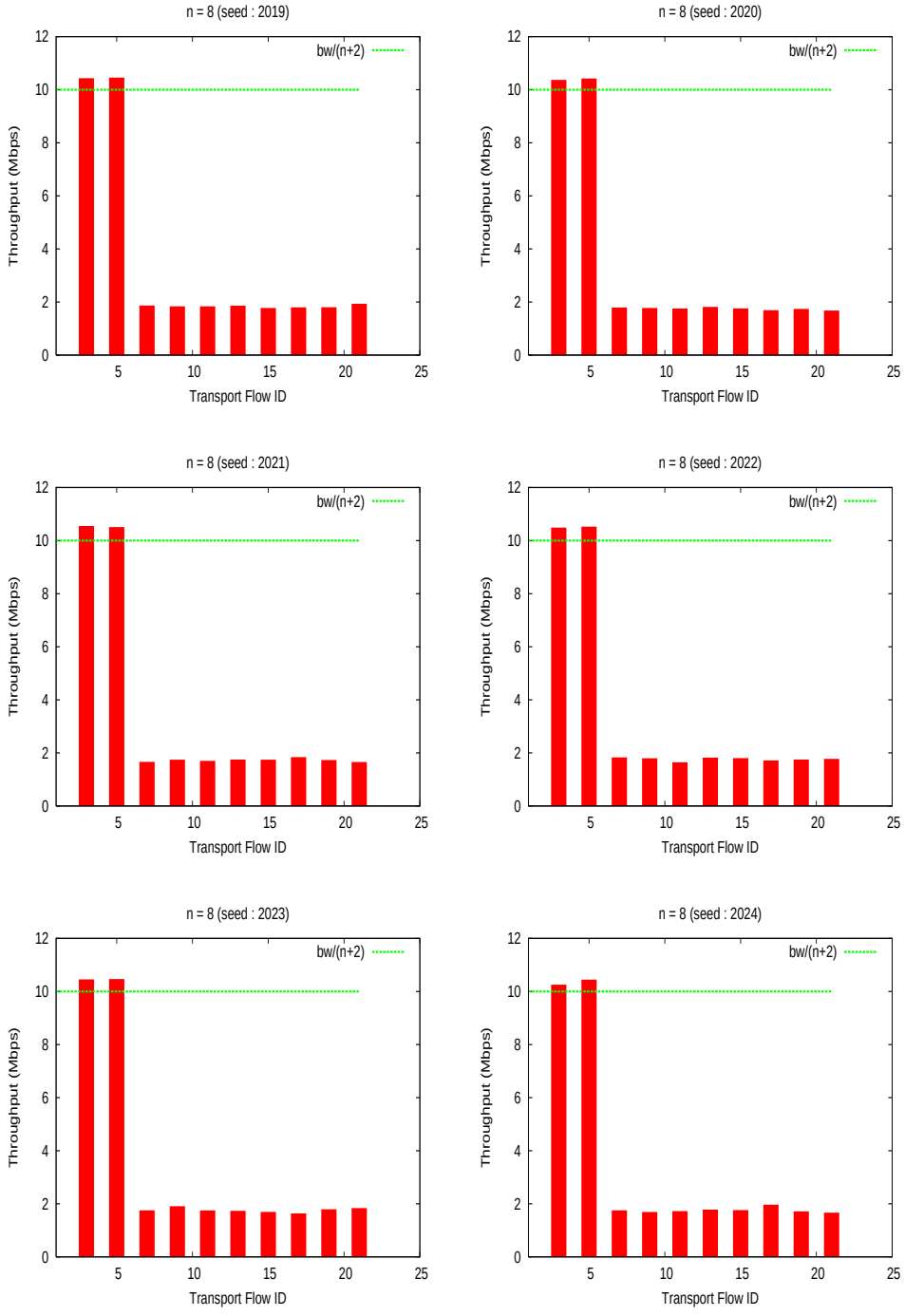


Figure B.4: Per flow throughput for two-SCTP case (n:8, seeds: 2019-2024)

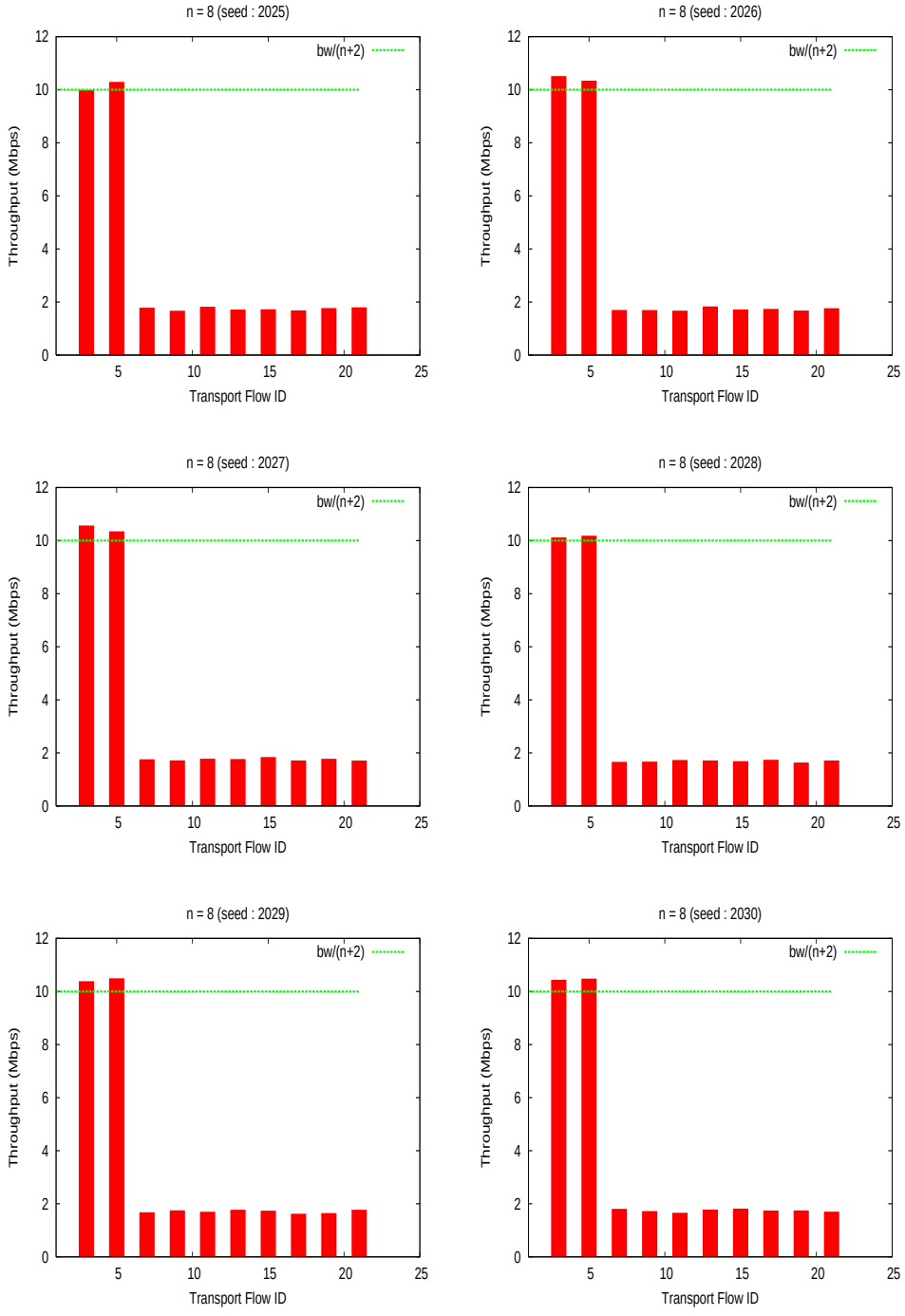


Figure B.5: Per flow throughput for two-SCTP case (n:8, seeds: 2025-2030)

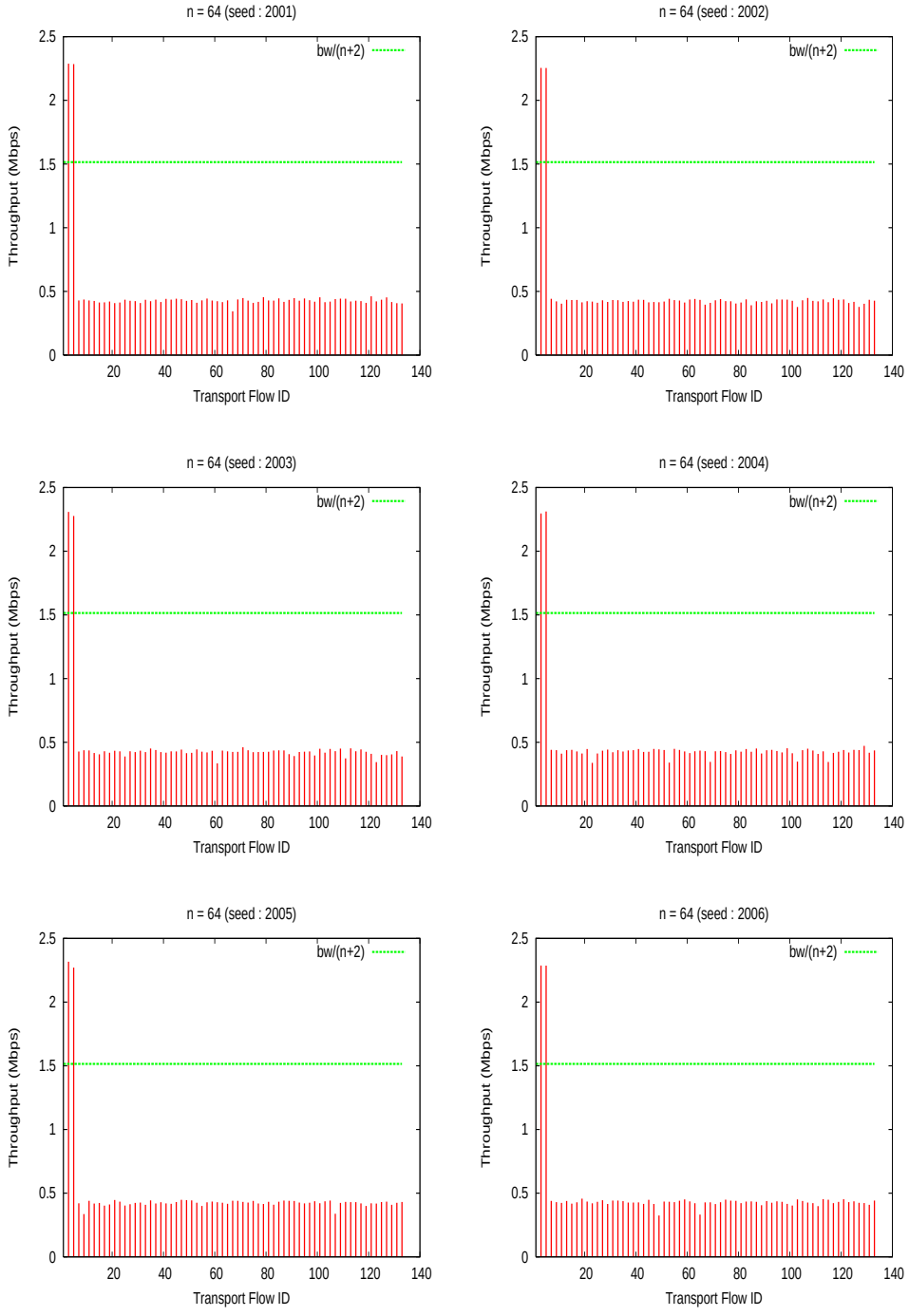


Figure B.6: Per flow throughput for two-SCTP case (n:64, seeds: 2001-2006)

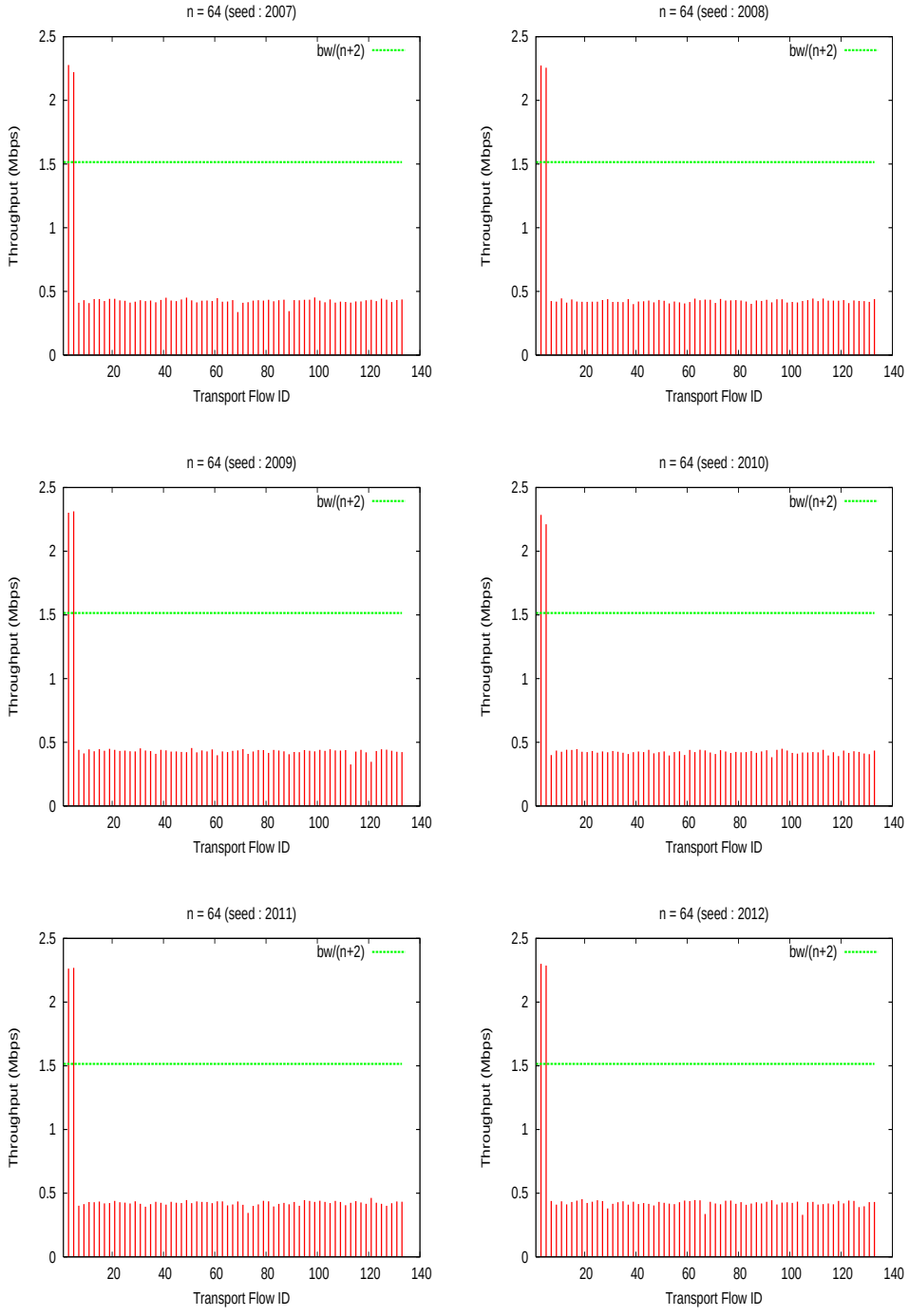


Figure B.7: Per flow throughput for two-SCTP case (n:64, seeds: 2007-2012)

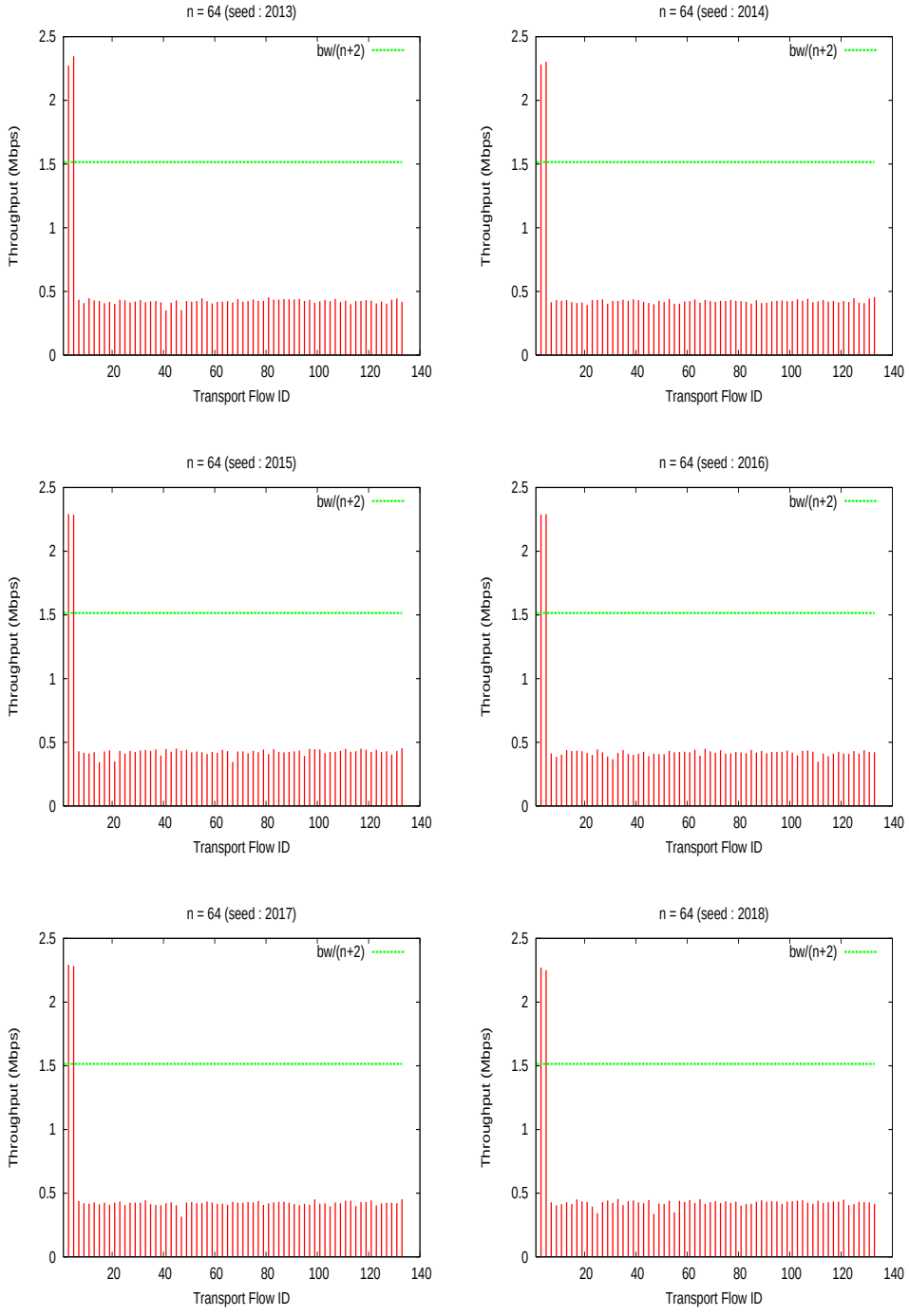


Figure B.8: Per flow throughput for two-SCTP case (n:64, seeds: 2013-2018)

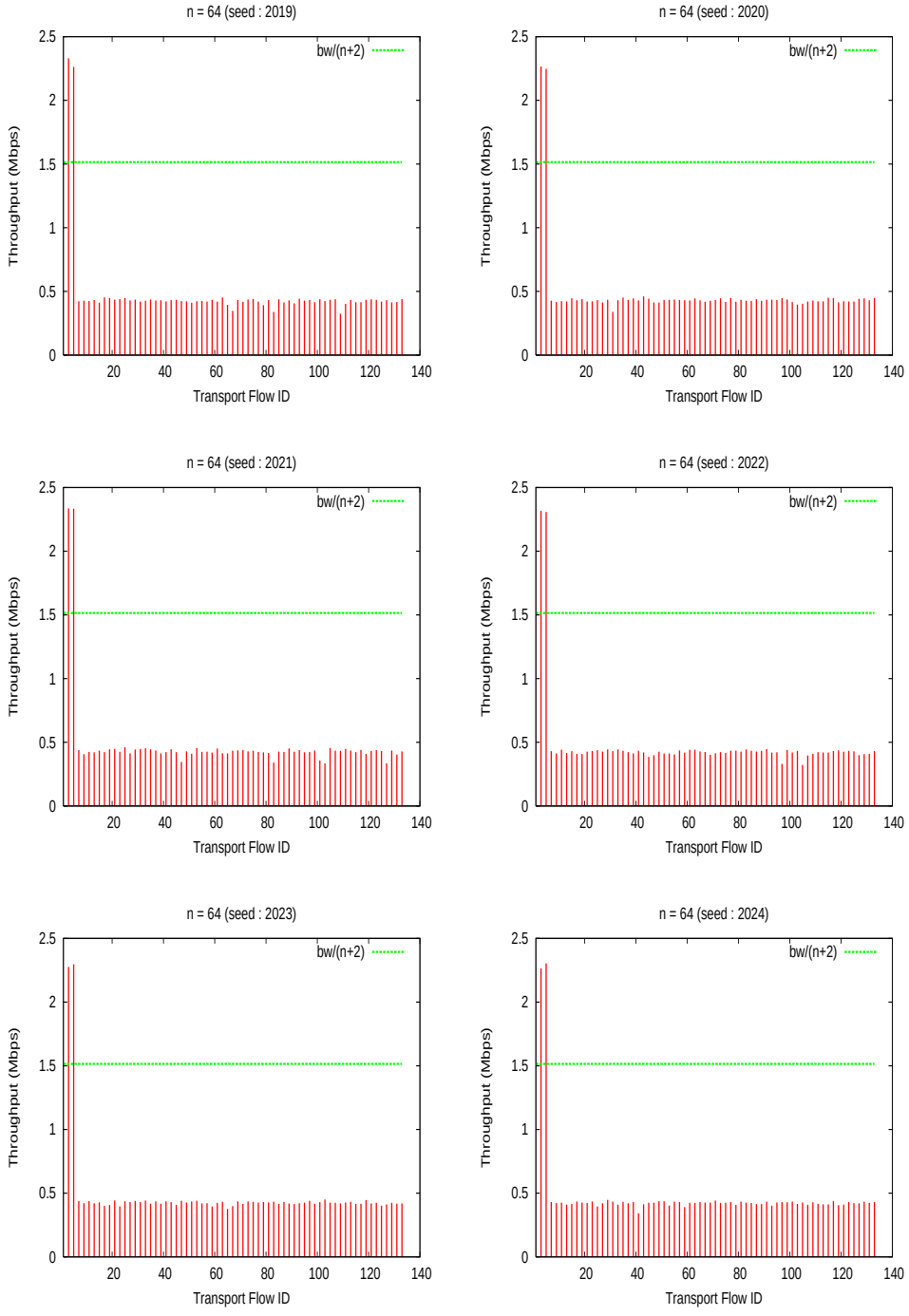


Figure B.9: Per flow throughput for two-SCTP case (n:64, seeds: 2019-2024)

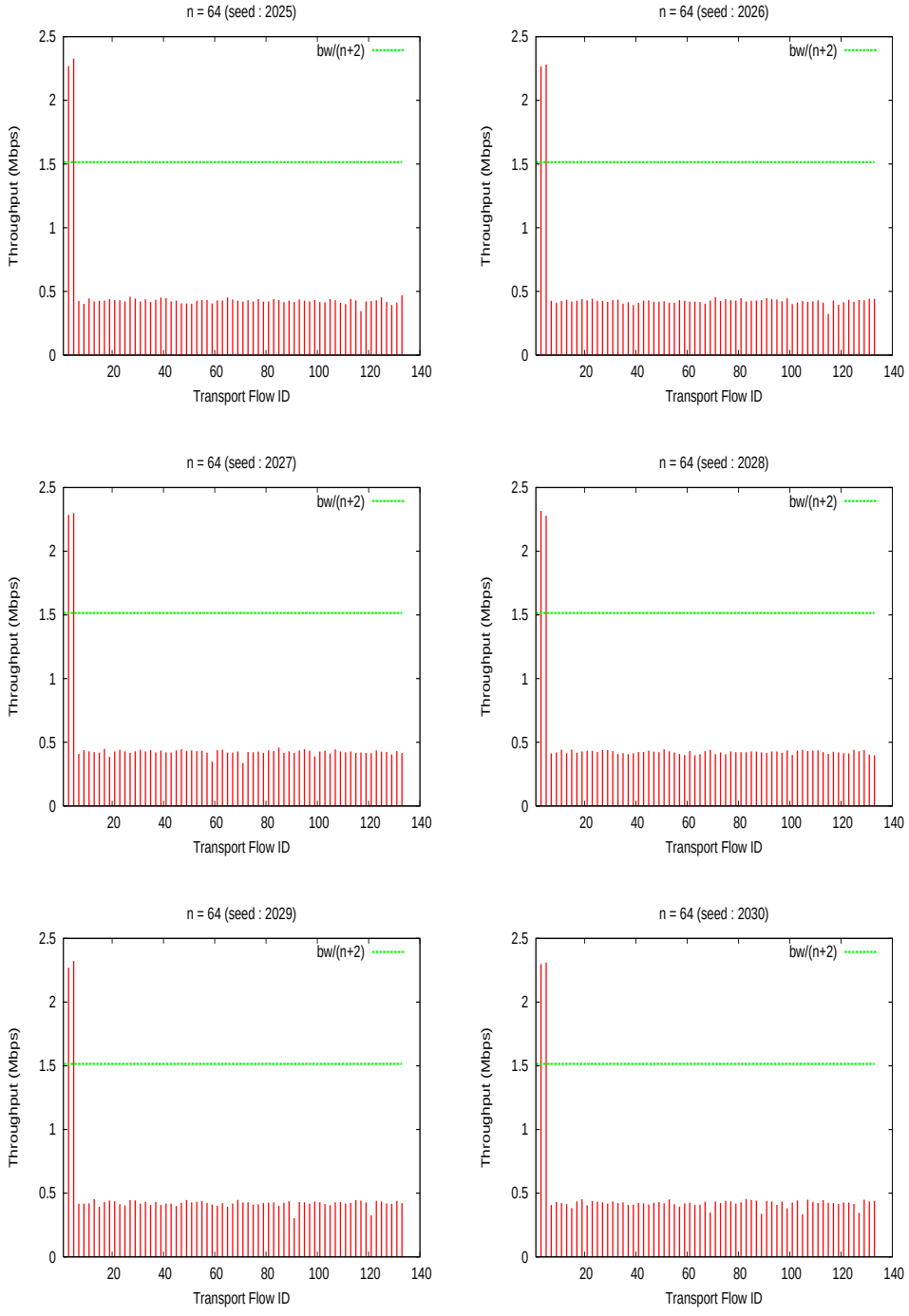


Figure B.10: Per flow throughput for two-SCTP case (n:64, seeds: 2025-2030)

Appendix C

THE CMT CASE: INDIVIDUAL FLOW THROUGHPUTS (FOR CHAPTER 5)

Note that, the first two bars in each of the figures in this appendix are for the two subflows of the CMT flow. The remaining bars are for each of the *n* TCP flows.

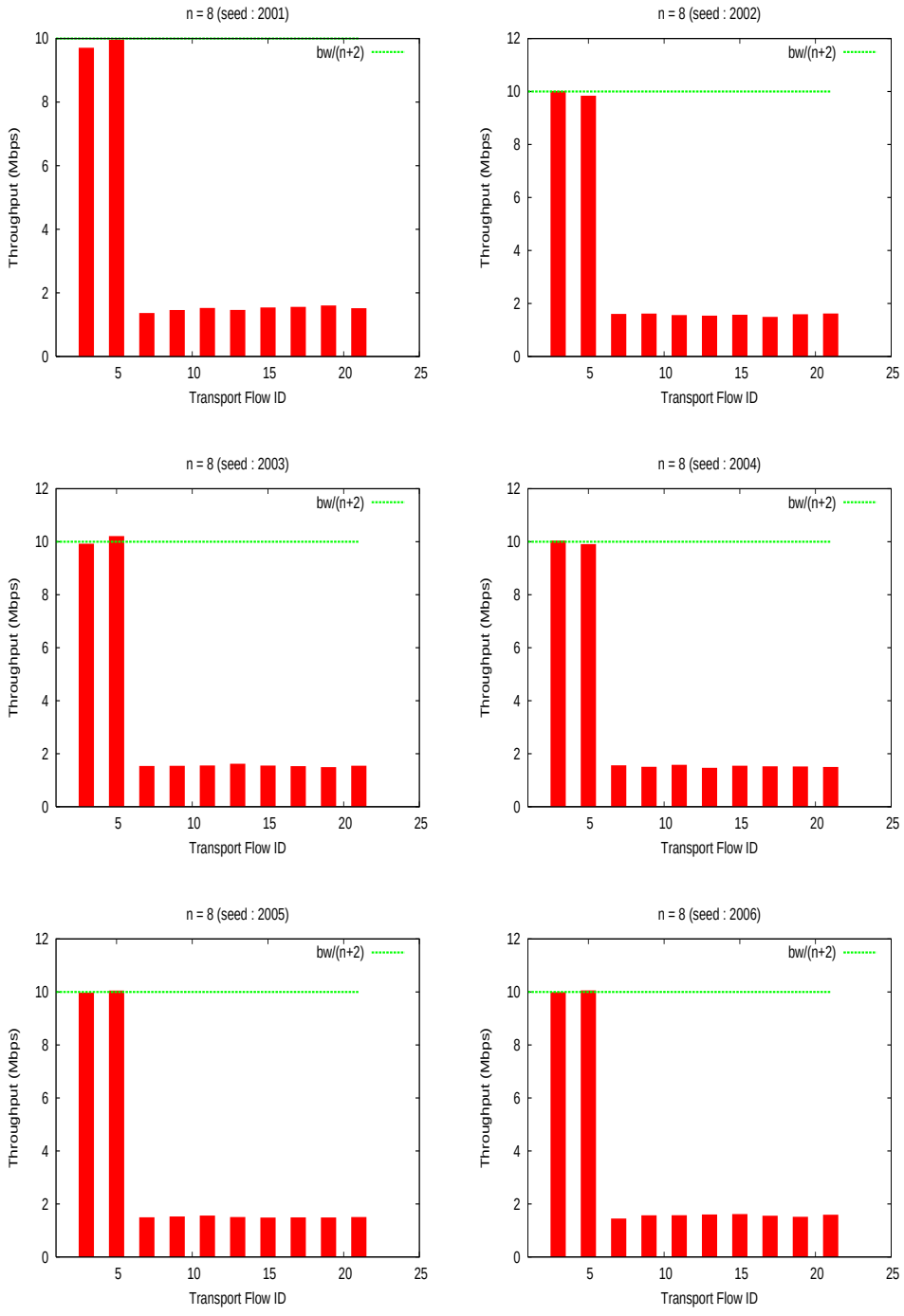


Figure C.1: Per flow throughput for the CMT case (n:8, seeds: 2001-2006)

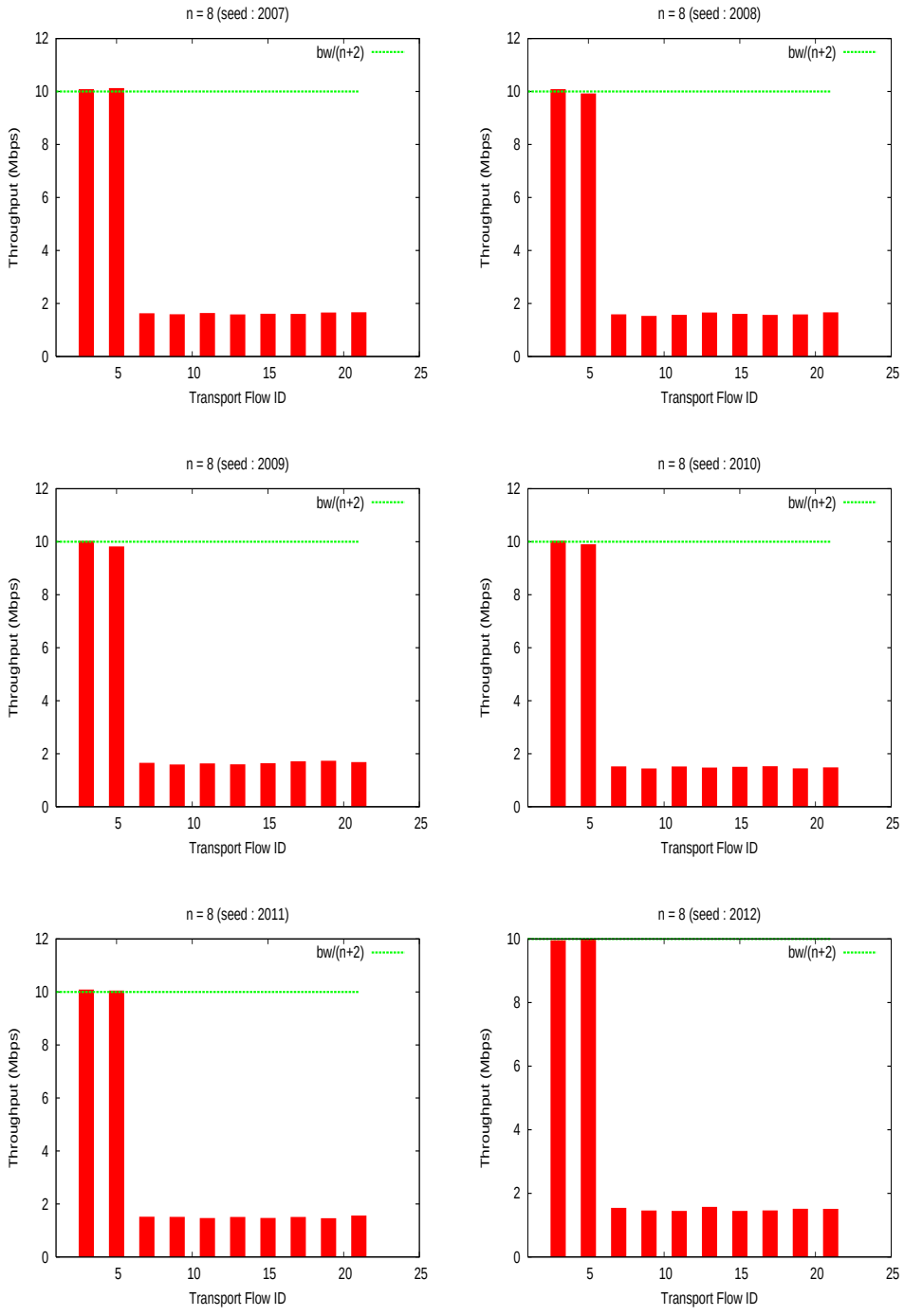


Figure C.2: Per flow throughput for the CMT case (n:8, seeds: 2007-2012)

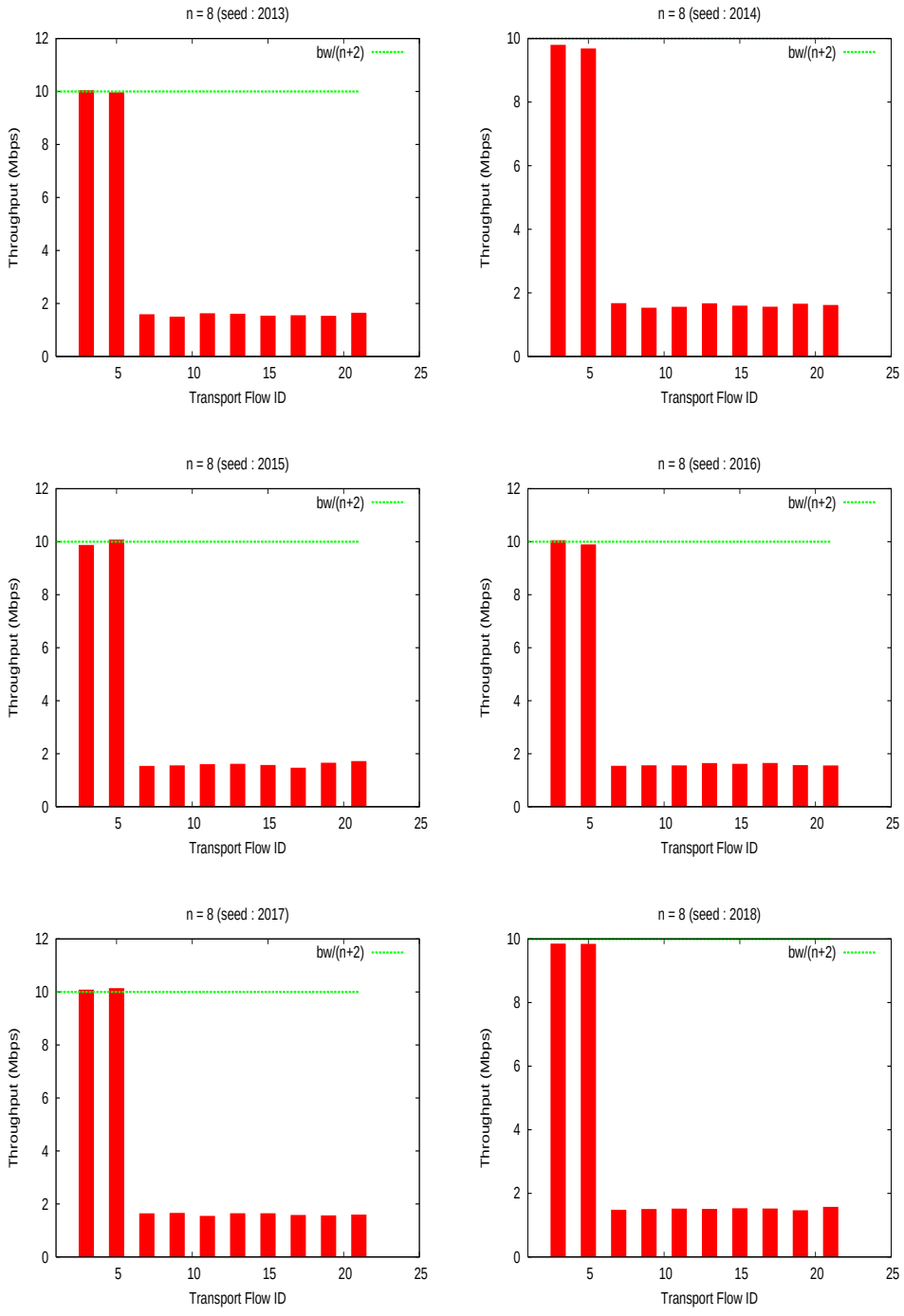


Figure C.3: Per flow throughput for the CMT case (n:8, seeds: 2013-2018)

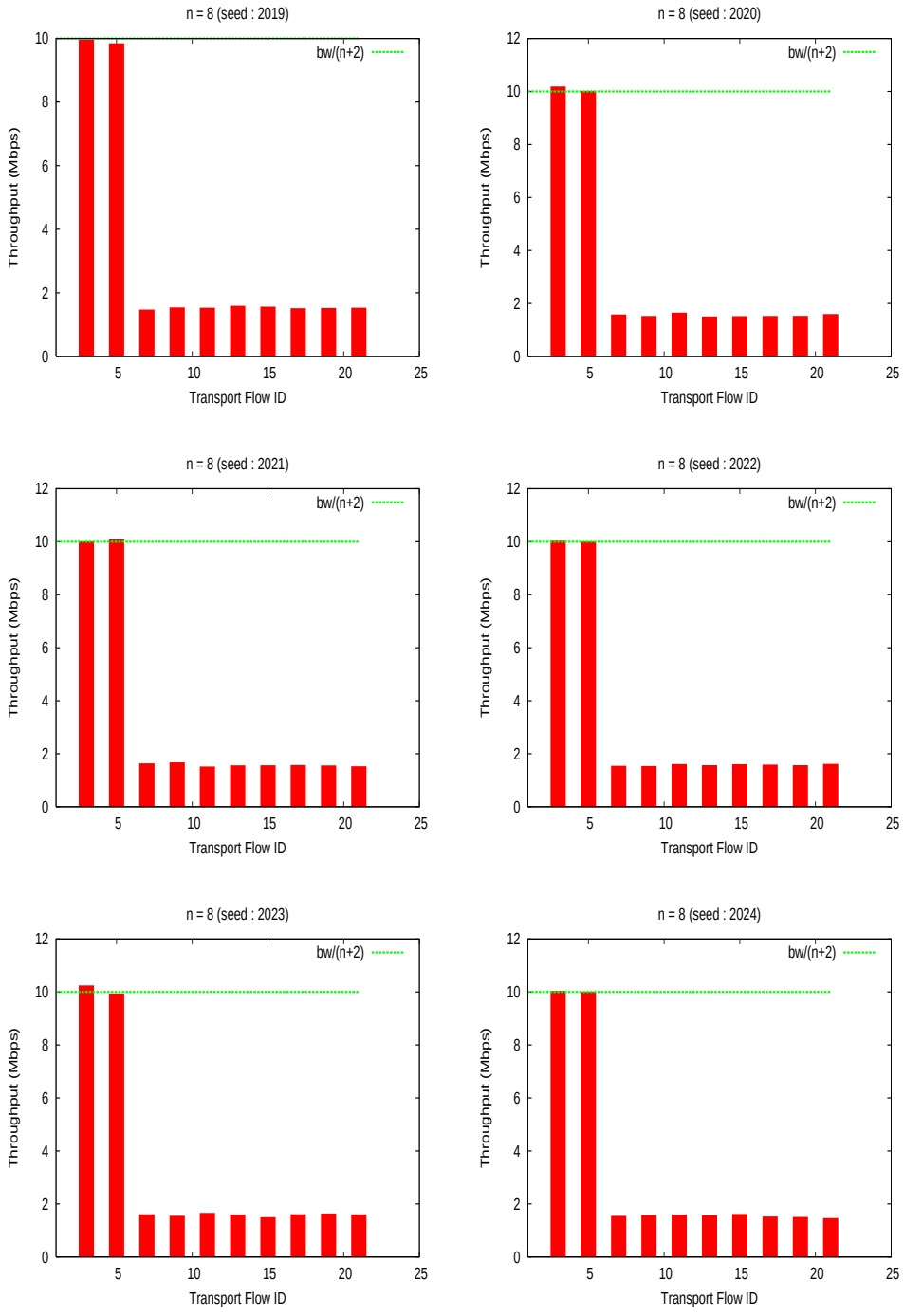


Figure C.4: Per flow throughput for the CMT case (n:8, seeds: 2019-2024)

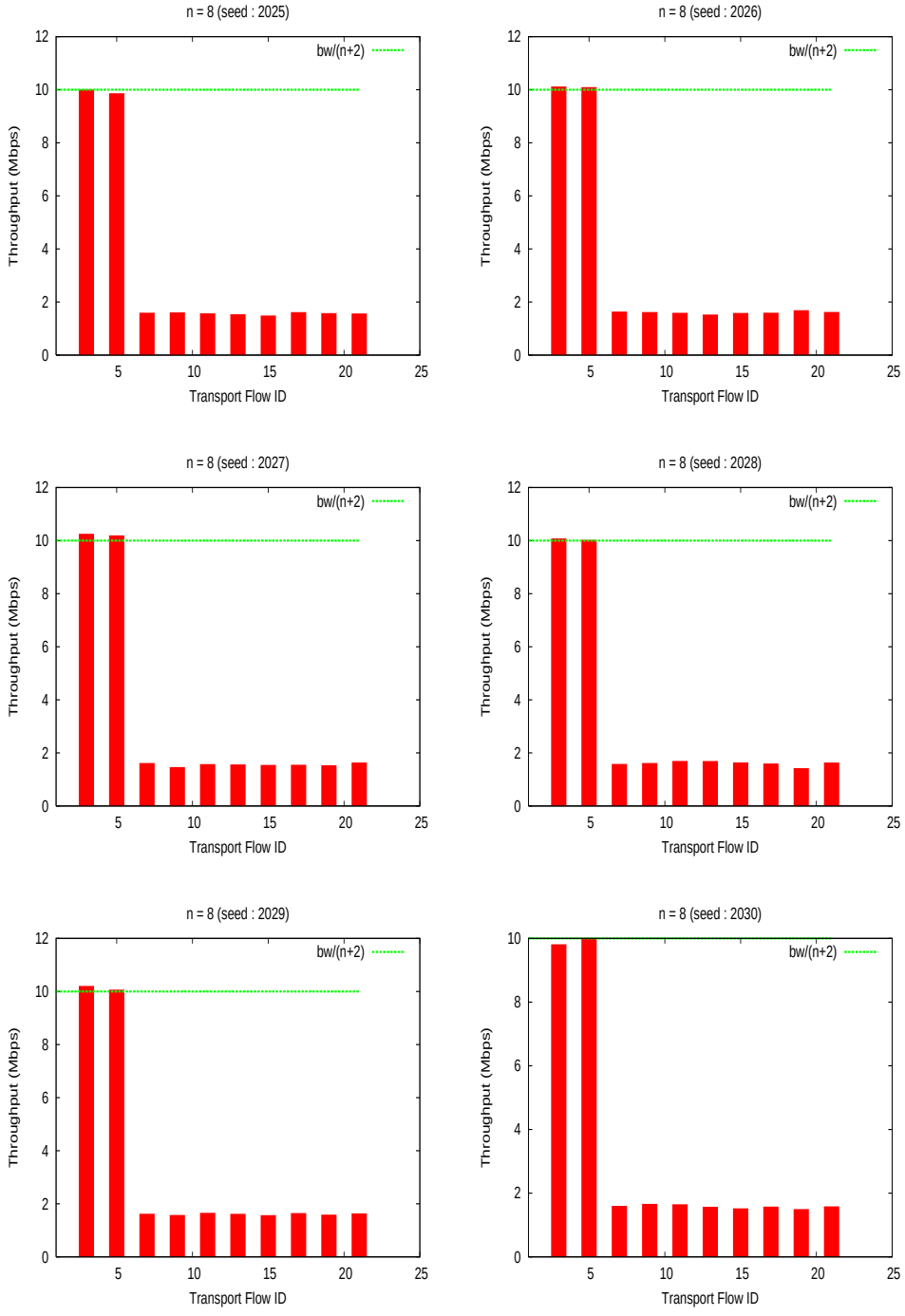


Figure C.5: Per flow throughput for the CMT case (n:8, seeds: 2025-2030)

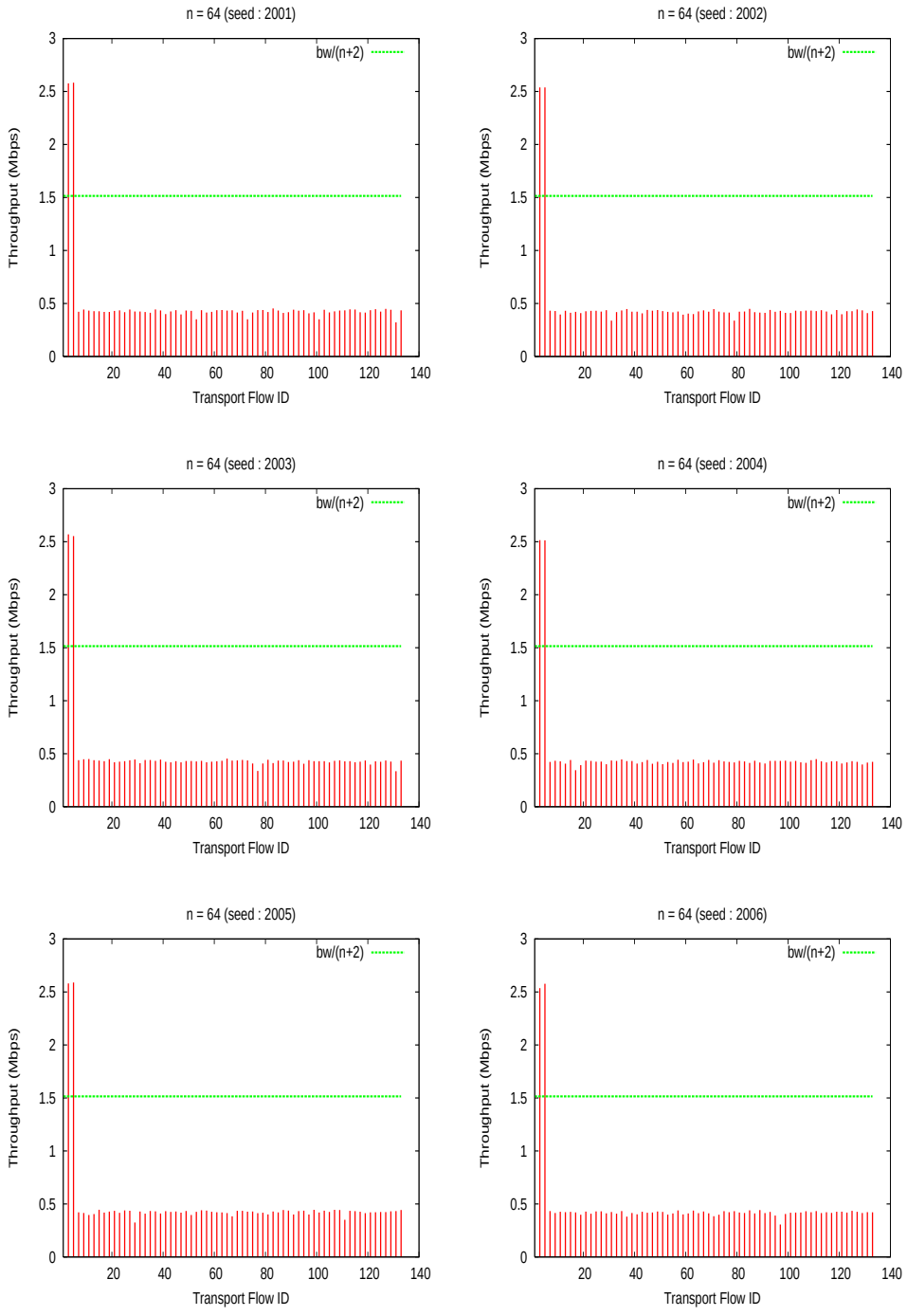


Figure C.6: Per flow throughput for the CMT case (n:64, seeds: 2001-2006)

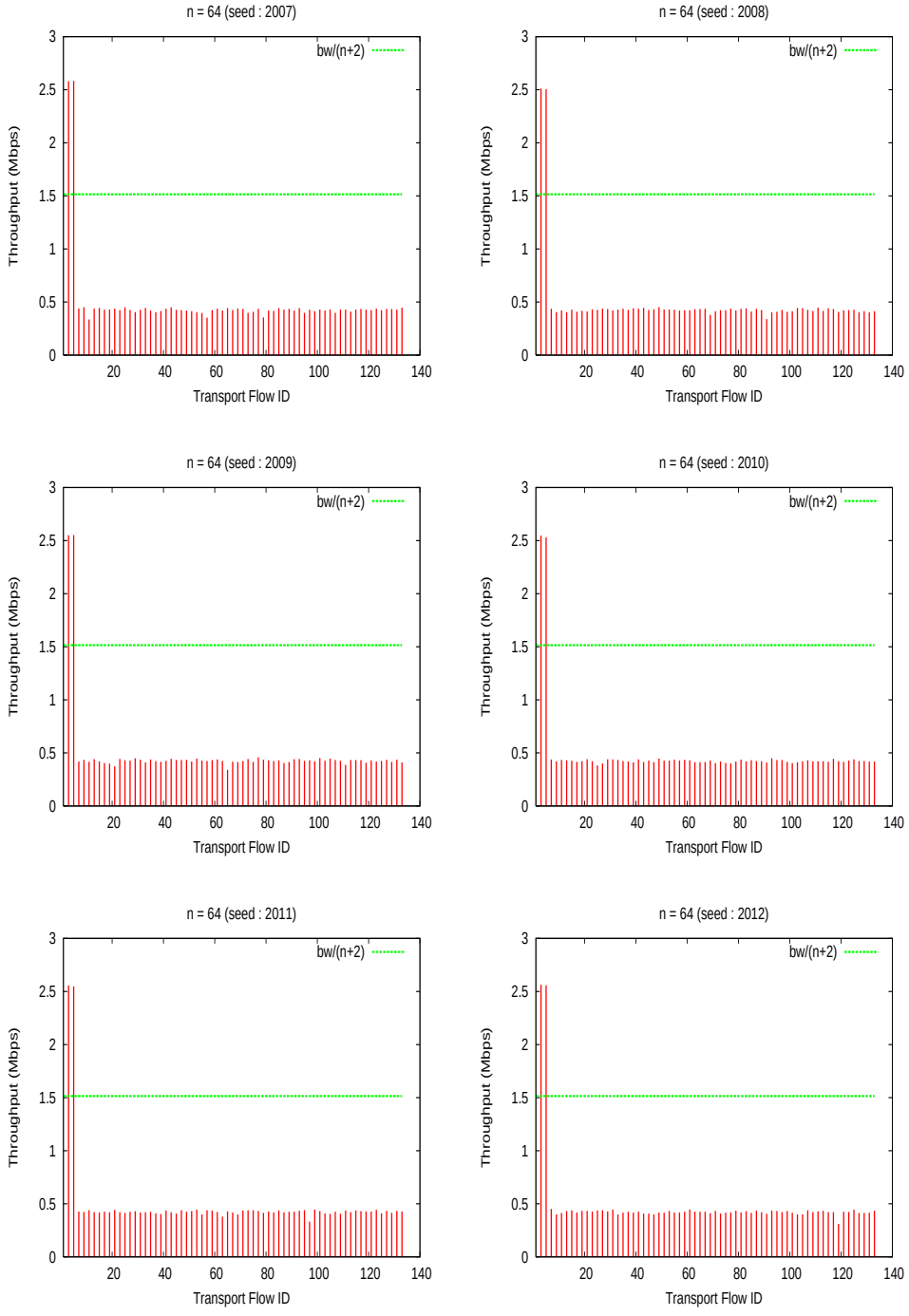


Figure C.7: Per flow throughput for the CMT case (n:64, seeds: 2007-2012)

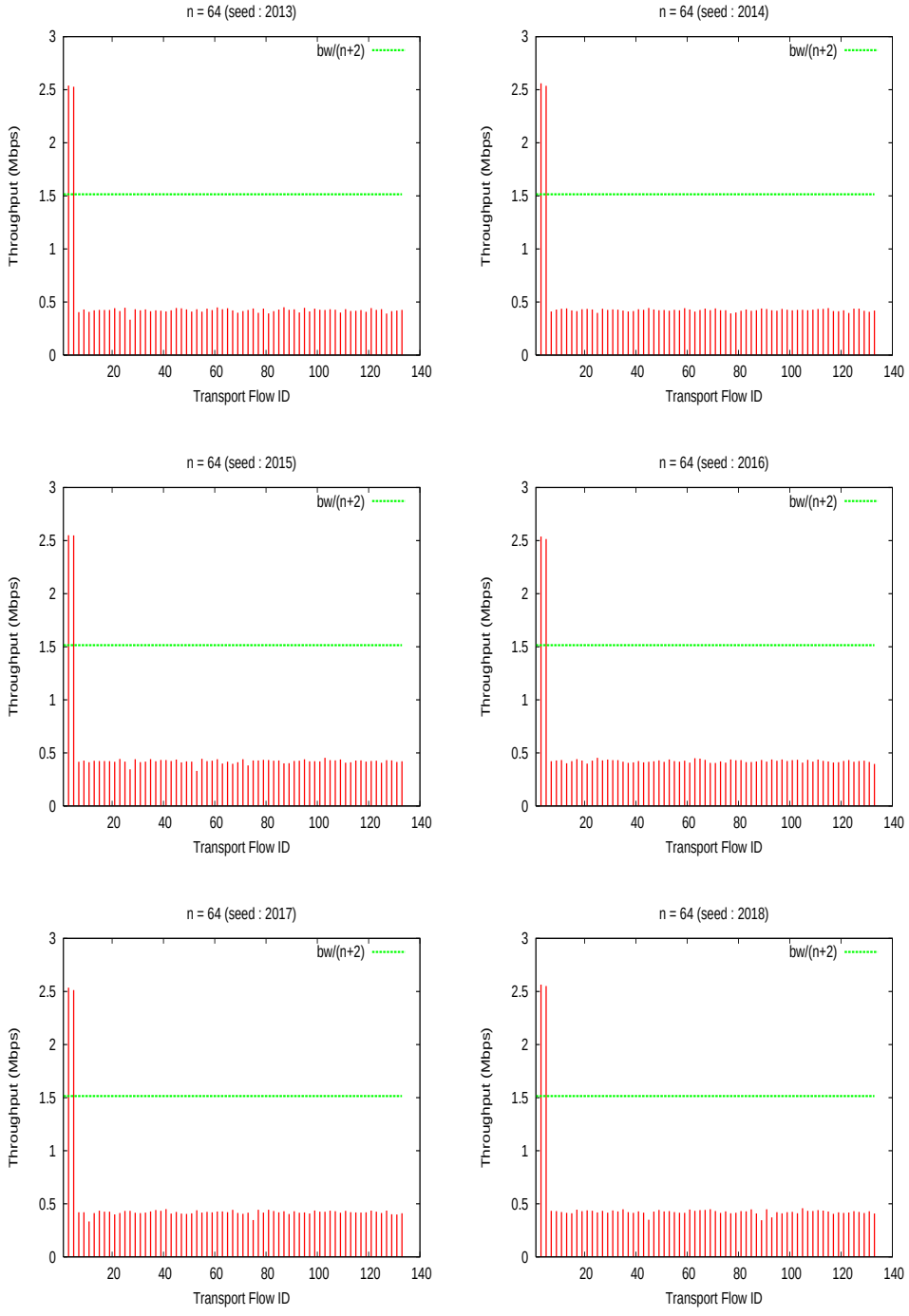


Figure C.8: Per flow throughput for the CMT case (n:64, seeds: 2013-2018)

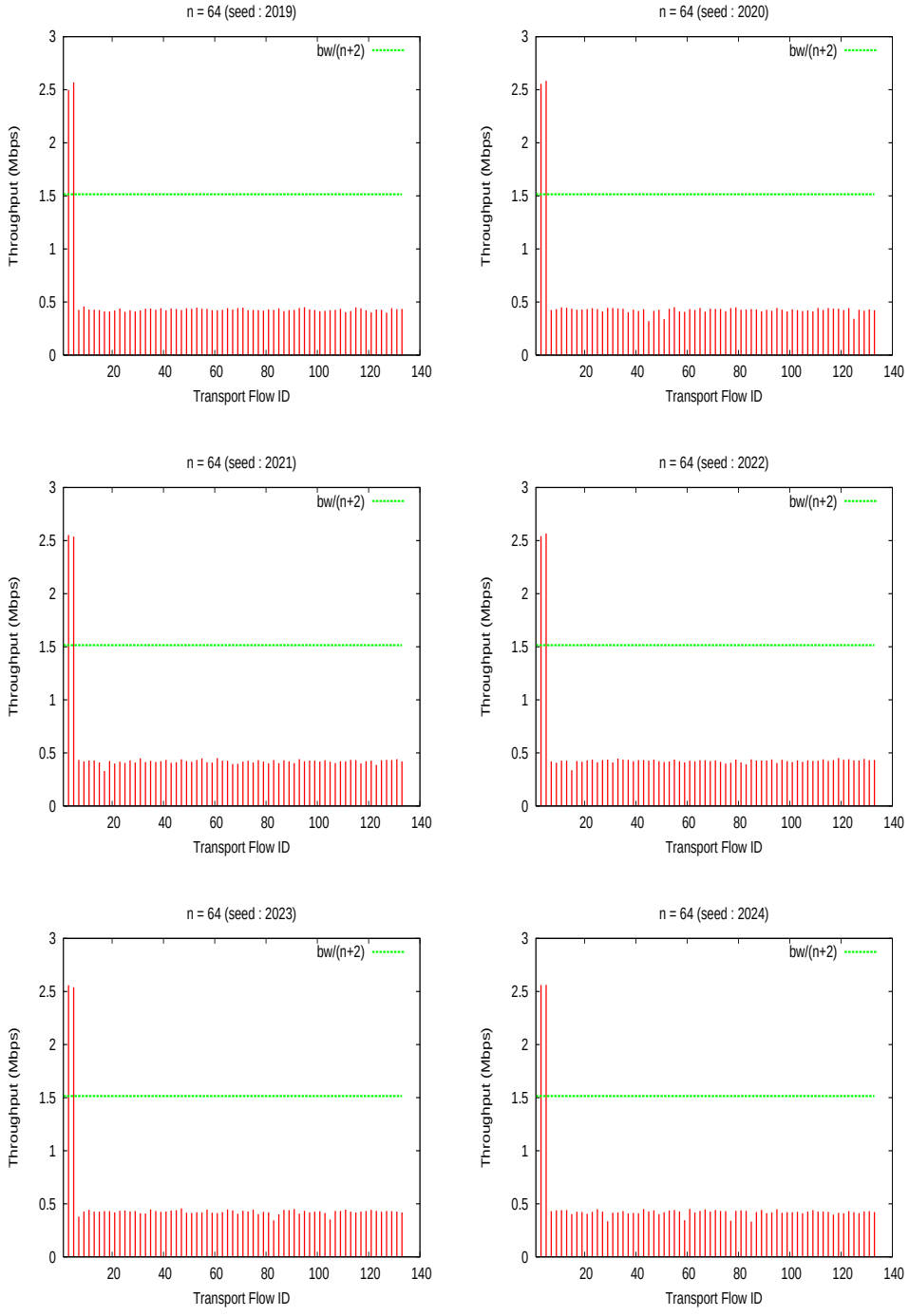


Figure C.9: Per flow throughput for the CMT case (n:64, seeds: 2019-2024)

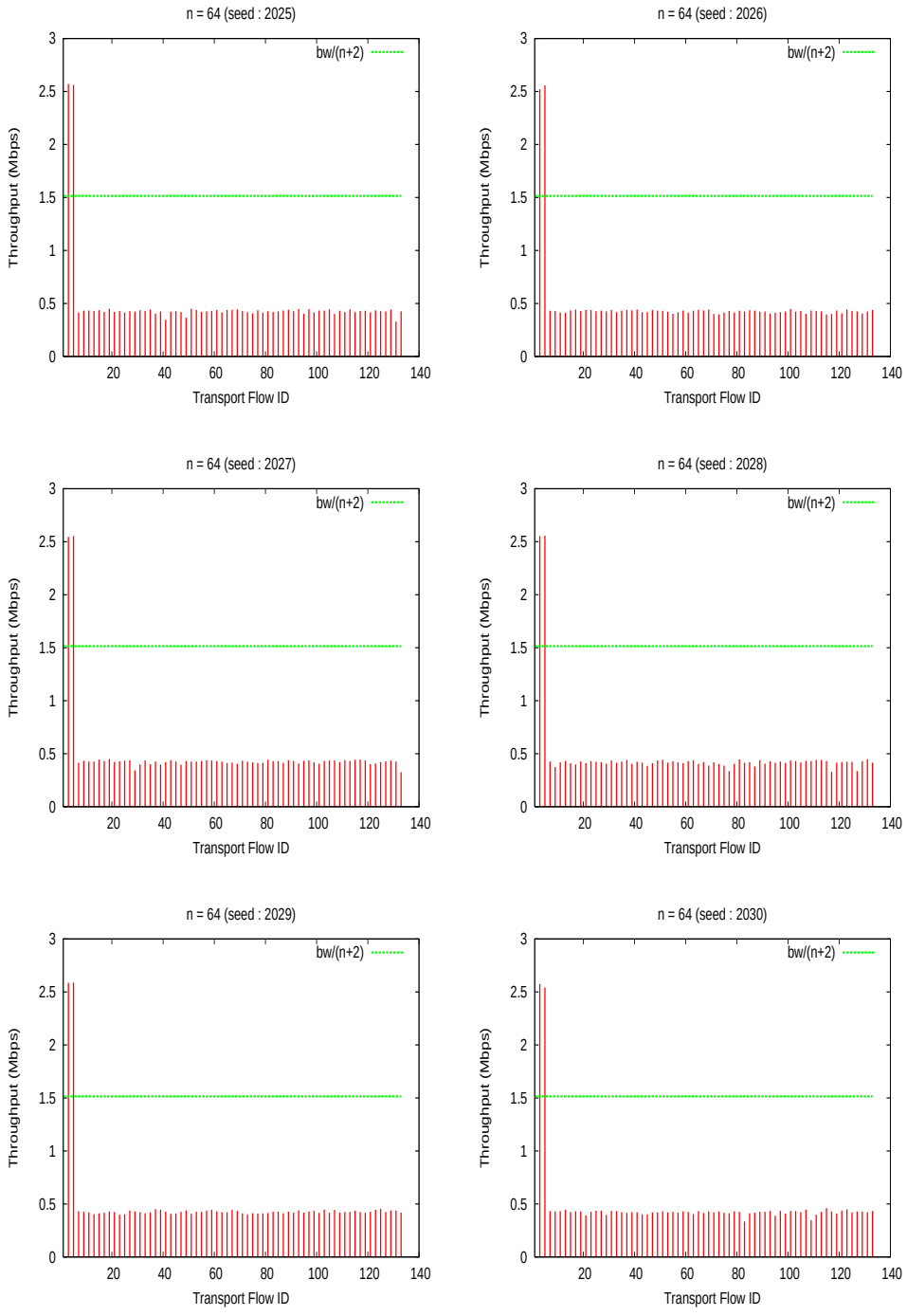


Figure C.10: Per flow throughput for the CMT case (n:64, seeds: 2025-2030)

Appendix D

SCTP QUALNET VS. NS-2 IMPLEMENTATION

In this appendix, we present some of the slight implementation differences that we noted between the SCTP QualNet 4.5.1 (svn version 1) module and the SCTP ns-2.31 module. The legend we used is as follows.

`DIFF(X, Y, ...): a feature where the QualNet module has a different implementation of the same functionality in ns-2 which might affect protocols X, Y, ....`

- **DIFF(CMT, SCTP[1]):** in detecting the stale acks[2]. Let's examine the code segment below in the `processSackChunk()` function of the ns-2 code.

```
....
if(spSackChunk->uiCumAck < uiCumAckPoint)
{
   /* this cumAck's a previously cumAck'd tsn
   * (ie, it's out of order!)
   * ...so ignore!
   */
   DBG_PL(ProcessSackChunk,
          "ignoring out of order sack!") DBG_PR;
```

[1] We refer to the plain SCTP as in [98].

[2] The conclusion section of [62] also talks about stale acks.

```
    DBG_X(ProcessSackChunk);
    return;
}
....
```

We noted that, this `if` statement is not sufficient to decide the most up-to-date SACK chunk reaching the data sender. Let's consider a CMT association between two paths P_s and P_f where P_s is slower (for instance, longer delay) than P_f. Usually SACKs from P_s will arrive at the sender later than SACKs from P_f. For instance, a SACK chunk such as

(i) `SACK(cumAck: 300, gapAckBlock: 302-303, arwnd: 5000)` via path P_s can arrive at the sender *after* the SACK chunk

(ii) `SACK(cumAck: 300, gapAckBlock: 302-305, arwnd: 3000)` via Path P_f.

Since, the `cumAck` values on both SACK chunks are the same, both SACK chunks will be processed at the sender. However, the SACK data at (i) is *older* than the SACK data at (ii). If the SACK chunk at (i) is processed after the SACK chunk at (ii), then the data sender will calculate the `peerRwnd` value incorrectly. This is because at the end of the `processSackChunk()` function, the following code segment is executed to calculate `peerRwnd` at the sender.

```
....
uiTotalOutstanding = TotalOutstanding();
if(uiTotalOutstanding <= spSackChunk->uiArwnd)
    uiPeerRwnd =
          (spSackChunk->uiArwnd  - uiTotalOutstanding);
else
    uiPeerRwnd = 0;
....
```

When `peerRwnd` is calculated incorrectly at the data sender, the sender might send more data chunks than the receiver can store in its receiver buffer. In this case, the receiver keep dropping the extra data chunks and will keep advertising `arwnd` value as zero. This situation causes the CMT association to get into a deadlock where some the data chunks are dropped at the receiver due to being out of buffer space while the sender keeps retransmitting these data chunks.

As a solution to this problem, we introduced a new variable `tcb->lastArwnd` into the QualNet code and modified the `if` statement in `processSackChunk()` function, where out-of-order SACK chunks are decided as follows.

```
....
if (
     (
      modularLt(sackChunk.cumTsnAck,
                tcb-> lastRcvdTsn, SCTP_SERIAL_BITS_32)
     )
     ||
     (
      (sackChunk.cumTsnAck == tcb-> lastRcvdTsn)
      && (sackChunk.arwnd > tcb-> lastArwnd)
     )
  )

{// out-of-order SACK
  snprintf(buf, SCTP_BUFFER_SIZE-1,
           "Node %u, SACK chunk is out-of-order.
           Drop the SACK silently.\n",
           node-> nodeId,
```

```
            tcb-> state);
    printSctpDebugMsg(buf);
  }
  else
  {// SACK is up-to-date and ready to be processed
    ....
  }
```

- **DIFF(CMT):** The ns-2 module introduces variable `recover` per destination address which is used only to prevent multiple cwnd cuts in a window. However, (*) when a SACK is being processed while a particular destination address is in fast recovery mode, the cwnd of the destination should not be modified [98]. It seems that the ns-2 module did not implement (*). However, in the QualNet module, we did implement (*), after a discussion with Dr. Janardhan Iyengar.

- **DIFF(CMT):** The ns-2 module keeps a variable named `recover` per destination address to keep track of the fast recovery mode per destination. Whenever a marked `TSN` is $>$ `recover`, cwnd cut is allowed. Instead, in the QualNet module, we keep two variables named `recover` and `inFastRecovery` per destination to keep track of the fast recovery mode per destination. Whenever there is a fast retransmission, the fast recovery mode is entered (if not already in fast recovery) and the highest outstanding TSN for the destination is recorded. Whenever the `pseudocumAck` for the destination $>$ `recover`, fast recovery for the destination terminates.

  This method of keeping track of the fast recovery mode per destination in the QualNet implementation is definitely different from ns-2 implementation. However, based on our discussion with Dr. Janardhan Iyengar, both methods make sense and should not make a difference in term of the functionality.

Appendix E

SCTP AND TCP PACKET FORMATS

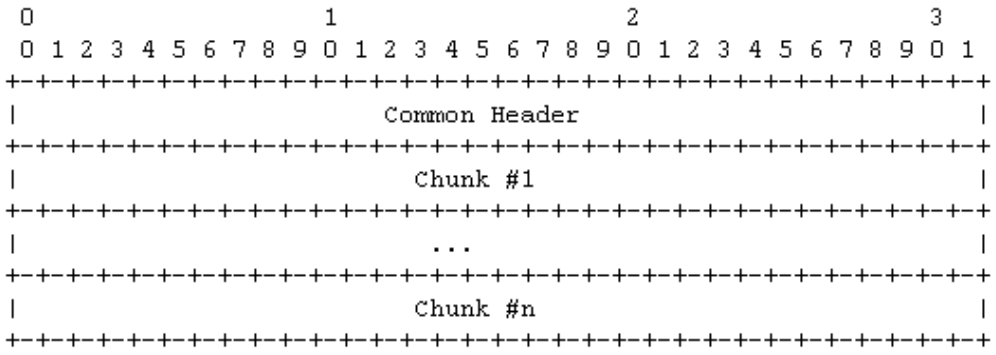


Figure E.1: SCTP Protocol Data Unit (PDU) format

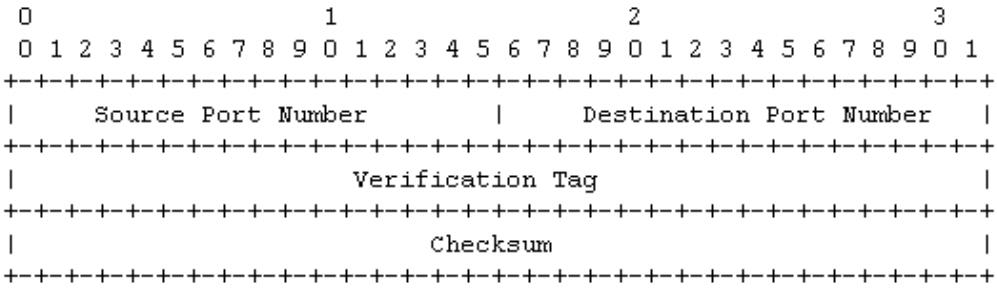


Figure E.2: SCTP Common Header format

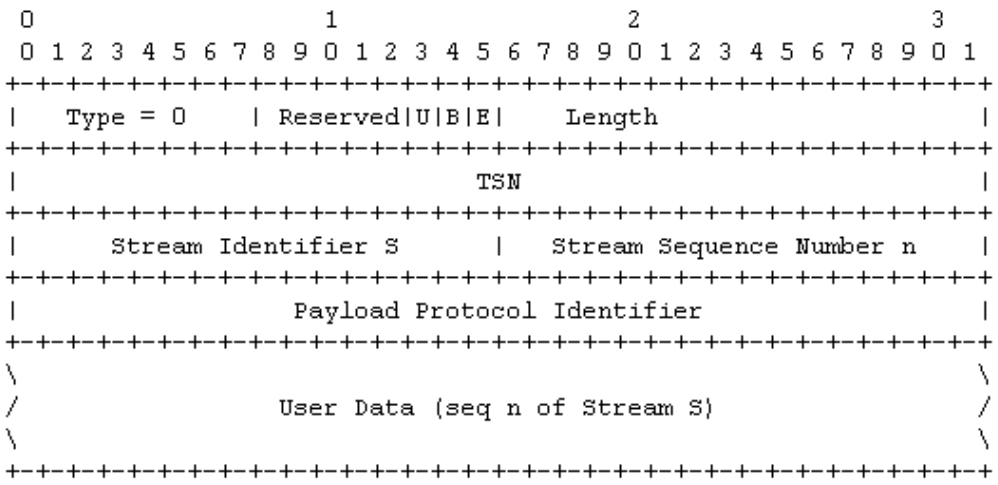


Figure E.3: SCTP Data chunk format

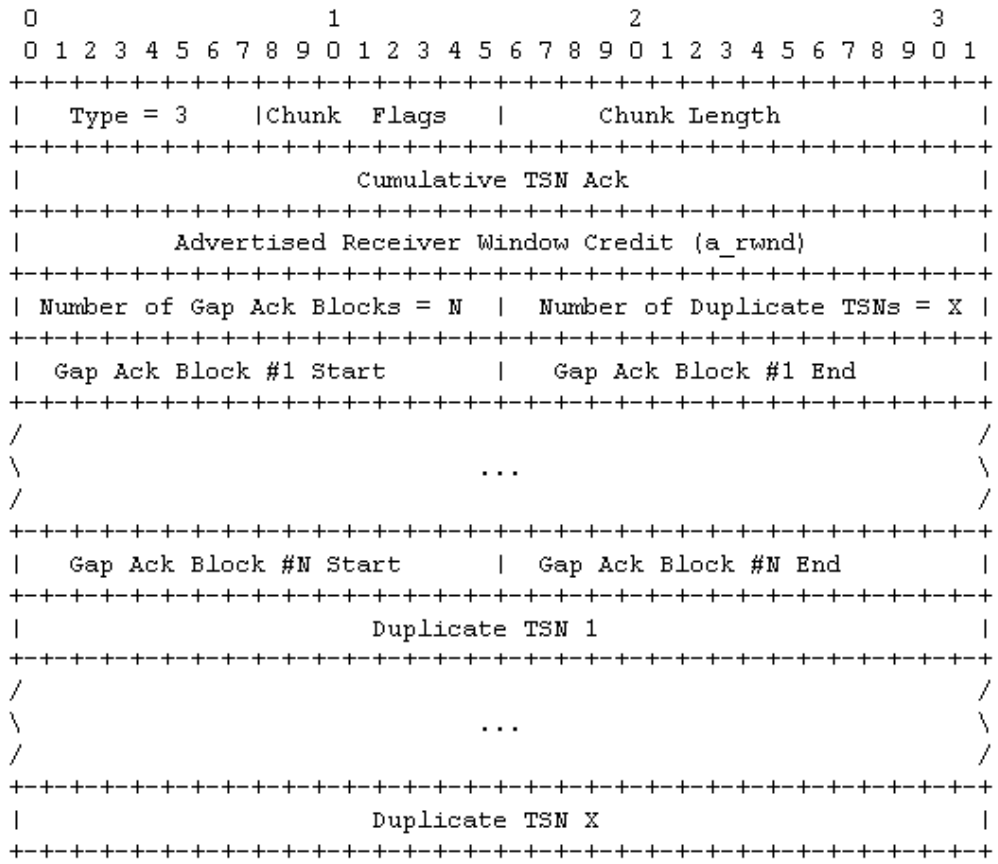


Figure E.4: SCTP SACK chunk format

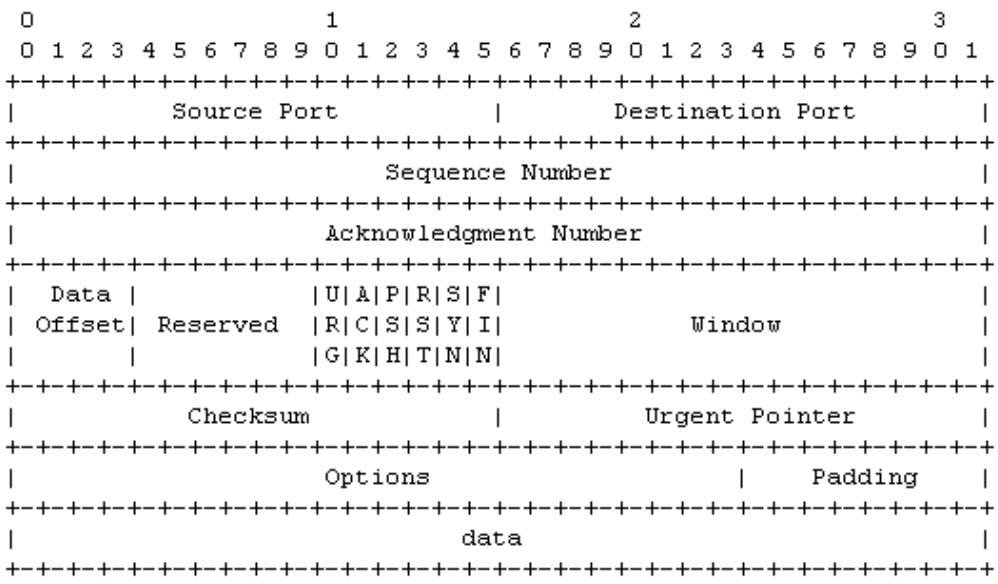


Figure E.5: TCP Protocol Data Unit (PDU) format

BIBLIOGRAPHY

[1] DARPA's Wireless Network After Next (WNAN) Program. http://www.darpa.mil/sto/solicitations/WNaN.

[2] BBN's PIRANA project. http://www.ir.bbn.com/~ramanath.

[3] Scalable Network Technologies, Inc. http://www.scalable-networks.com.

[4] The Network Simulator – ns-2. http://www.isi.edu/nsnam/ns.

[5] NSF WHYNET (Scalable Testbed for the Next Generation Wireless Networking Technologies) project. http://chenyen.cs.ucla.edu/projects/whynet.

[6] Clean Slate Design for the Internet. http://cleanslate.stanford.edu.

[7] International Conference Re-Architecting the Internet (ReArch). http://conferences.sigcomm.org/co-next/2009/workshops/rearch.

[8] Multipath TCP (MPTCP). http://trac.tools.ietf.org/area/tsv/trac/wiki/MultipathTcp.

[9] NSF's FIND (Future Internet Design) Program. http://www.nets-find.net.

[10] The Future of TCP: Train-wreck or Evolution? http://yuba.stanford.edu/trainwreck.

[11] Trilogy Project – Architecturing the Future Internet. http://www.trilogy-project.org.

[12] IEEE Std. 802.11, Wireless LAN Media Access Control (MAC) and Physical Layer (PHY) Specification, 1999.

[13] Ian F. Akyildiz, Xudong Wang, and Weilin Wang. Wireless mesh networks: a survey. *Elsevier Science, Computer Networks*, 47:445–487, 2005.

[14] R. Alamgir, M. Atiquzzaman, and W. Ivancic. Impact of retransmission mechanisms on the performance of SCTP and TCP. In *AMCOS'05: Proceedings of the 4th WSEAS International Conference on Applied Mathematics and Computer Science*, Rio de Janeiro, Brazil, 2005. World Scientific and Engineering Academy and Society (WSEAS).

[15] Rumana Alamgir, Mohammed Atiquzzaman, and William Ivancic. Effect of Congestion Control on the Performance of TCP and SCTP over Satellite Networks. In *In NASA Earth Science Technology Conference*, 2002.

[16] M. Allman. TCP Congestion Control with Appropriate Byte Counting (ABC). RFC 3465, Internet Engineering Task Force, February 2003.

[17] M. Allman, S. Floyd, and C. Partridge. Increasing TCP's Initial Window. RFC 3390, Internet Engineering Task Force, October 2002.

[18] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. RFC 2581, Internet Engineering Task Force, April 1999.

[19] E. Altman and T. Jimenez. Novel delayed ACK techniques for improving TCP performance in multihop wireless networks. In *Personal Wireless Communications (PWC)*, Venice, Italy, September 2003.

[20] L. Andrew, C. Marcondes, S. Floyd, L. Dunn, R. Guillier, W. Gang, L. Eggert, S. Ha, and I. Rhee. Towards a Common TCP Evaluation Suite. In *Proceedings of the International Workshop on Protocols for Fast Long-Distance Networks (PFLDnet)*, Manchester, United Kingdom, March 2008.

[21] Ilknur Aydin. SCTP QualNet Simulation Module. `http://degas.cis.udel.edu/SCTP/`.

[22] Ilknur Aydin, Renwei Ge, Preethi Natarajan, and Chien-Chung Shen. Performance Evaluation of SCTP in Mobile Ad Hoc Network. Technical Report 2005-18, University of Delaware, Newark, DE, May 2005.

[23] Ilknur Aydin, Woojin Seok, and Chien-Chung Shen. Cellular SCTP: A Transport-Layer Approach to Internet Mobility. In *IEEE 12th International Conference on Computer Communications and Networks (IC3N'03)*, Texas, October 2003.

[24] Ilknur Aydin and Chien-Chung Shen. Evaluating Cellular SCTP over One-Hop Wireless Networks. In *IEEE 62nd Semiannual Vehicular Technology Conference (VTC2005-Fall)*, Dallas, Texas, September 2005.

[25] Ilknur Aydin and Chien-Chung Shen. NSF WHYNET Project Progress Report, October 2007.

[26] Ilknur Aydin and Chien-Chung Shen. Performance Evaluation of Concurrent Multipath Transfer using SCTP Multihoming in Multihop Wireless Networks. In *IEEE 8th International Symposium on Network Computing and Applications (NCA)*, Cambridge, MA, USA, July 2009.

[27] 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 *IEEE MASS*, 2004.

[28] B. Braden, D. Clark, J. Crowcroft, S. Deering, D. Estrin, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang. RFC 2309, Internet Engineering Task Force, April 1998.

[29] R. Braden. Requirements for Internet Hosts – Communication Layers. RFC 1122, Internet Engineering Task Force, October 1989.

[30] R. Brennan and T. Curran. SCTP congestion control: Initial simulation studies. In *International Teletraffic Congress (ITC 17)*, Brazil, 2001.

[31] Bob Briscoe. Flow Rate Fairness: Dismantling a Religion. *SIGCOMM Computer Communication Review*, 37(2):63–74, April 2007.

[32] Armando Caro and Janardhan Iyengar. SCTP ns-2 Simulation Module. `http://pel.cis.udel.edu`.

[33] C. Casetti and W. Gaiotto. Westwood SCTP: load balancing over multipaths using bandwidth-aware source scheduling. In *IEEE VTC2004-Fall*, volume 4, pages 3025–3029, September 2004.

[34] J. Chen, M. Gerla, Y. Z. Lee, and M. Y. Sanadidi. TCP with delayed ack for wireless networks. *Elsevier Ad hoc Networks*, 2007.

[35] K. Chen, Y. Xue, and K. Nahrstedt. On Setting TCP's Congestion Window Limit in Mobile Ad hoc Networks. In *IEEE ICC*, 2003.

[36] K. Chen, Y. Xue, S.H. Shah, and K. Nahrstedt. Understanding Bandwidth-Delay Product in Mobile Ad Hoc Networks. *Elsevier Computer Communications Journal*, 27(10):923–934, 2004.

[37] Dah-Ming Chiu and Raj Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, 17(1):1–14, 1989.

[38] Carlos De M. Cordeiro, Samir R. Das, and Dharma P. Agrawal. COPAS: Dynamic Contention Balancing to Enhance the Performance of TCP over Multi-hop Wireless Networks. In *In Proc. 10th Intl. Conf. on Computer Comm. and Networks (IC3N)*, pages 382–387, 2002.

[39] Jon Crowcroft and Phillipe Oechslin. Differentiated end-to-end Internet Services using a Weighted Proportional Fair Sharing TCP. *SIGCOMM Comput. Commun. Rev.*, 28(3):53–69, 1998.

[40] Dragana Damjanovic and Michael Welzl. MulTFRC: Providing Weighted Fairness for Multimedia Applications (and others too!). *SIGCOMM Comput. Commun. Rev.*, 2009.

[41] Dragana Damjanovic, Michael Welzl, Miklos Telek, and Werner Heiss. Extending the TCP Steady-State Throughput Equation for Parallel TCP Flows. Technical Report DPS NSG Technical Report 2, University of Innsbruck, Institute of Computer Science, August 2008.

[42] R. de Oliveira and T. Braun. A Smart TCP Acknowledgment Approach for Multihop Wireless Networks. *EEE Transactions on Mobile Computing*, 6(2):192–205, 2007.

[43] Richard Draves, Jitendra Padhye, and Brian Zill. Routing in Multi-radio, Multi-hop Wireless Mesh Networks. In *ACM MobiCom*, pages 114–128, 2004.

[44] Ahmed Abd El, Tarek Saadawi, and Myung Lee. LS-SCTP: a Bandwidth Aggregation Technique for Stream Control Transmission Protocol. *Computer Communications*, 27:1012–1024, 2004.

[45] S. Elrakabawy, A. Klemm, and C. Lindemann. TCP with Adaptive Pacing for Multihop Wireless Networks. In *IEEE Mobihoc*, Urbana-Champaign, Illinois, USA, 25 – 27 May 2005.

[46] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, Internet Engineering Task Force, June 1999.

[47] S. Floyd. Congestion Control Principles. RFC 2914, Internet Engineering Task Force, September 2000.

[48] S. Floyd and M. Allman. Comments on the Usefulness of Simple Best-Effort Traffic. RFC 5290, Internet Engineering Task Force, July 2008.

[49] S. Floyd, A. Arcia, D. Ros, and J. Iyengar. Adding Acknowledgement Congestion Control to TCP, November 2007. IETF Internet Draft, draft-floyd-tcpm-ackcc-02.txt (work in progress).

[50] S. Floyd, M. Handley, J. Padhye, and J. Widmer. TCP Friendly Rate Control (TFRC): Protocol Specification. RFC 5348, Internet Engineering Task Force, September 2008.

[51] S. Floyd and T. Henderson. The NewReno Modification to TCP's Fast Recovery Algorithm. RFC 2582, Internet Engineering Task Force, April 1999.

[52] S. Floyd, T. Henderson, and A. Gurtov. The NewReno Modification to TCP's Fast Recovery Algorithm. RFC 3782, Internet Engineering Task Force, April 2004.

[53] S. Floyd and V. Jacobson. On Traffic Phase Effects in Packet-Switched Gateways. *Internetworking: Research and Experience*, 3:115–156, 1992.

[54] Sally Floyd. RED: Discussions of Setting Parameters, November 2007. `http://www.icir.org/floyd/REDparameters.txt`.

[55] Sally Floyd and Kevin Fall. Promoting the Use of End-to-End Congestion Control in the Internet. *IEEE/ACM Transactions on Networking*, 7:458–472, 1999.

[56] Sally Floyd, Mark Handley, Jitendra Padhye, and Jrg Widmer. Equation-Based Congestion Control for Unicast Applications. In *ACM SIGCOMM*, 2000.

[57] Sally Floyd and Van Jacobson. Random Early Detection gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[58] Roberta Fracchia, Claudio Casetti, Carla-Fabiana Chiasserini, and Michela Meo. WiSE Extension of SCTP for Wireless Networks. In *IEEE ICC*, Seoul, South Korea, May 2005.

[59] Z. Fu, P. Zerfoz, H. Luo, S. Lu, L. Zhang, and M. Gerla. The Impact of Multihop Wireless Channel on TCP Throughput and Loss. In *IEEE INFOCOM*, 2003.

[60] J. Iyengar. *End-to-end Concurrent Multipath Transfer Using Transport Layer Multihoming*. Ph.D. dissertation, University of Delaware, Newark, DE, USA, April 2006.

[61] J. Iyengar, P. Amer, and R. Stewart. Performance Implications of a Bounded Receive Buffer In Concurrent Multipath Transfer. *Computer Communications*, 30(4):818–829, February 2007.

[62] Janardhan Iyengar, Paul Amer, and Randall Stewart. Receive Buffer Blocking In Concurrent Multipath Transfer. In *GLOBECOM*, November 2005.

[63] Janardhan Iyengar, Paul Amer, and Randall Stewart. Concurrent Multipath Transfer Using SCTP Multihoming Over Independent End-to-End Paths. *IEEE/ACM Transactions on Networking*, 14(5):951–964, October 2006.

[64] V. Jacobson, R. Braden, and D. Borman. TCP Extensions for High Performance. RFC 1323, Internet Engineering Task Force, May 1992.

[65] Van Jacobson. Congestion Avoidance and Control. In *ACM SIGCOMM*, pages 314–329, Stanford, CA, August 1988.

[66] Manish Jain and Constantinos Dovrolis. End-to-End Available Bandwidth: Measurement methodology, Dynamics, and Relation with TCP Throughput. *IEEE/ACM Transactions on Networking*, 11(4), August 2003.

[67] R. Jain, W. Hawe, and D. Chiu. A Quantitative measure of fairness and discrimination for resource allocation in Shared Computer Systems. Technical Report 301, DEC, September 1984.

[68] D. Johnson, Y. Hu, and D. Maltz. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. Technical Report 4728, Internet Engineering Task Force, February 2007.

[69] A. Jungmaier, M. Schopp, and M. Tuxen. Performance Evaluation of the Stream Control Transmission Protocol. In *IEEE Conference on High Performance Switching and Routing (HPSR)*, pages 141–148, Germany, June 2000.

[70] V. Kawadia and P.R. Kumar. Experimantal Investigations into TCP Performance Over Wireless Multihop Networks. In *Sigcomm Workshops*, Philadelphia, USA, August 2005.

[71] Frank Kelly. Charging and Rate Control for Elastic Traffic. *European Transactions on Telecommunications*, 8:33–37, 1997.

[72] Fang-Chun Kuo and Xiaoming Fu. Probe-Aided MulTCP: an Aggregate Congestion Control Mechanism. *SIGCOMM Comput. Commun. Rev.*, 38(1):17–28, 2008.

[73] Sung-Ju Lee and Mario Gerla. Split Multipath Routing with Maximally Disjoint Paths in Ad hoc Networks. In *IEEE International Conference on Communications (ICC)*, Helsinki, 2001.

[74] Jinyang Li, Charles Blake, Douglas S. J. De Couto, Hu Imm Lee, and Robert Morris. Capacity of Ad Hoc wireless Networks. In *Mobicom*, pages 61–69, Rome, Italy, 2001.

[75] Jianxin Liao, Jingyu Wang, and Xiaomin Zhu. cmpSCTP: An Extension of SCTP to Support Concurrent Multi-Path Transfer. In *IEEE ICC*, pages 5762–5766, May 2008.

[76] Yunxin Liu, Yongqiang Xiong, Yang Yang, Pengzhi Xu, and Qian Zhang. An Experimental Study on Multi-channel Multi-radio Multi-hop Wireless Networks. In *IEEE Globecom*, 2005.

[77] J. Mahdavi and S. Floyd. TCP-friendly unicast rate-based flow control, January 8 1997. Techical note sent to end2end-interest mailing list.

[78] Mahesh K. Marina and Samir R. Das. On-demand Multipath Distance Vector Routing in Ad Hoc Networks. In *IEEE International Conference on Network Protocols (ICNP)*, pages 14–23, 2001.

[79] M. Mathis. Rethinking the TCP-friendly Paradigm. IETF Internet-Draft: draft-mathis-iccrg-unfriendly-pre00 (work in progress), December 2008.

[80] M. Mathis, J.Mahdavi, S.Floyd, and A.Romano. TCP Selective Acknowledgment Options (SACK). RFC 2018, Internet Engineering Task Force, October 1996.

[81] J. Nagle. Congestion Control in TCP/IP internetworks. RFC 896, Internet Engineering Task Force, January 1984.

[82] P. Natarajan. *Leveraging Innovative Transport Layer Services for Improved Application Performance*. Ph.D. dissertation, University of Delaware, Newark, DE, USA, February 2009.

[83] P. Natarajan, P. D. Amer, and R. Stewart. Corrections on: Improving file transfers using SCTP multistreaming. Originally published in the proceedings of IEEE IPCCC 2004 but the corrected version is available at `http://www.eecis.udel.edu/~amer/PEL/poc/`.

[84] P. Natarajan, P. D. Amer, and R. Stewart. Multistreamed Web Transport for Developing Regions. In *ACM SIGCOMM Workshop on Networked Systems for Developing Regions (NSDR)*, Seattle, 2008.

[85] D. Ott, T. Sparks, and K. Mayer-Patel. Aggregate Congestion Control for Distributed Multimedia Applications. In *IEEE INFOCOM*, March 2004.

[86] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. *ACM SIGCOMM*, pages 303–314, 1998.

[87] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. Technical Report 3561, Internet Engineering Task Force, July 2003.

[88] Federico Perotto, Claudio Casetti, and Giulio Galante. SCTP-based Transport Protocols for Concurrent Multipath Transfer. In *IEEE WCNC 2007*, pages 2969–2974, Hong Kong, March 2007.

[89] J. Postel. Transmission Control Protocol. RFC 793, Internet Engineering Task Force, September 1981.

[90] Ioannis Psaras and Vassilis Tsaoussidis. Why TCP Timers (still) Don't Work Well. *Computer Networks Journal (COMNET), Elsevier Science*, 51:2033–2048, 2007.

[91] Lili Qiu, Yin Zhang, and Srinivasan Keshav. Understanding the performance of many tcp flows. *Computer Networks*, 37:277–306, 2001.

[92] Joshua Robinson, Konstantina Papagiannaki, Christophe Diot, Xingang Guo, and Lakshman Krishnamurthy. Experimenting with a Multi-Radio Mesh Networking Testbed. In *Testbed Workshop on Wireless Network Measurements (Winmee)*, 2005.

[93] Srinivas Shakkottai, R. Srikant, Nevil Brownlee, Andre Broido, and K.C. Claffy. The RTT distribution of TCP flows in the Internet and its impact on TCP-based flow control. Technical Report 2004-02, CADIA.

[94] Scott Shenker, Lixia Zhang, and David D. Clark. Some observations on the dynamics of a congestion control algorithm. *ACM CCR*, pages 30–39, 1990.

[95] Irfan Sheriff and Elizabeth M. Belding-Royer. Multipath Selection in Multi-radio Mesh Networks. In *BROADNETS*, 2006.

[96] A. K. Singh and K. Kankipati. TCP-ADA: TCP with adaptive delayed acknowledgement for mobile ad hoc networks. In *IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1685 – 1690, Atlanta, March 2004.

[97] M. Singh, P. Pradhan, and P. Francis. MPAT: Aggregate TCP Congestion Management as a Building Block for Internet QoS. In *IEEE International Conference on Network Protocols*, pages 129–138, October 2004.

[98] R. Stewart. Stream Control Transmission Protocol. RFC 4960, Internet Engineering Task Force, September 2007.

[99] R. Stewart, I. Arias-Rodriguez, K. Poon, A. Caro, and M. Tuexen. Stream Control Transmission Protocol (SCTP) Specification Errata and Issues . RFC 4460, Internet Engineering Task Force, April 2006.

[100] Randall Stewart and Paul D. Amer. Why is SCTP needed given TCP and UDP are widely available? ISOC MEMBER BRIEFING 17, June 2004. `http://www.isoc.org/briefings/017/index.shtml`.

[101] Anand Prabhu Subramanian, Milind M. Buddhikot, and Scott C. Miller. Interference Aware Routing in Multi-Radio Wireless Mesh Networks. In *Second International Workshop on Wireless Mesh Networks (WiMesh)*, Reston, VA, September 2006.

[102] M. Takai, J. Martin, and R. Bagrodia. Effects of Wireless Physical Layer Modeling in Mobile Ad Hoc Networks. In *ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, Rome, Italy, July 2001.

[103] Wai-Hong Tam and Yu-Chee Tseng. Joint Multi-Channel Link Layer and Multi-Path Routing Design for Wireless Mesh Networks. In *IEEE INFOCOM*, 2007.

[104] Xiaofei Wang, Wei Cai, Ye Yan, Taekyoung Kwon, Yanhee Choi, and Wenhua Zeng. A Framework of Distributed Dynamic Multi-radio Multi-channel Multi-path Routing Protocol in Wireless Mesh Networks. In *ICOIN*, Thailand, January 2009.

[105] Xiaofei Wang, Taekyoung Kwon, and Yanghee Choi. TCP improvement in Multi-radio Multi-channel Multi-hop Networks. In *ACM CoNEXT Student Workshop*, Madrid, Spain, 2008.

[106] S. Xu and T. Saadawi. Does the 802.11 MAC Protocol Work Well in Multihop Wireless Ad hoc Networks. *IEEE Communications Magazine*, June 2001.

[107] Guanhua Ye, Tarek Saadawi, and Myung Lee. SCTP Congestion Control Performance In Wireless Multi-Hop Networks. In *MILCOM*, 2002.

[108] Guanhua Ye, Tarek Saadawi, and Myung Lee. IPCC-SCTP: an enhancement to the standard SCTP to support multi-homing efficiently. In *IEEE International Conference on Performance, Computing, and Communications*, pages 523–530, 2004.