CONGESTION CONTROL FOR SCTP IN WIRELESS AD-HOC NETWORKS

by

GUANHUA YE

A dissertation submitted to the Graduate Faculty in Engineering in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

2004

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

This manuscript has been read and accepted for the Graduate Faculty in Engineering in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

7/19/04
Date

Chair of Examining Committee

July 19, 2004
Date

Mumtaz K. Kassir
Executive Officer

Myung Lee

Ümit Uyar

Yi Sun

Mehmet Ulema

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

CONGESTION CONTROL FOR SCTP
IN WIRELESS AD-HOC NETWORKS

by

Guanhua Ye

Advisor: Professor Tarek Saadawi

This thesis focuses on improving Stream Control Transmission Protocol (SCTP) congestion control performance in wireless ad hoc networks. As our experiments show, schemes proposed in this work are effective for SCTP to handle wireless errors in ad hoc network, and the proposed Independent per Path Congestion Control framework also extends SCTP's capability in supporting multi-homing applications, such as load balancing and load sharing. In this work, we first conduct extensive simulation studies of SCTP performance in wireless ad hoc networks. The goal is to identify its limitations in such environment. The performance evaluation work gives us better understanding of the operation of the protocol, and it also represents essential input for further improvement of SCTP performance in wireless ad hoc networks.

To handle packet losses caused by wireless errors in ad hoc networks, we first introduce ECN mechanism into SCTP, which makes SCTP ECN-capable. Then we use the concept of congestion coherence to differentiate wireless losses from congestion losses by using ECN as congestion signal. Wireless-aware SCTP is then developed, which decouples loss recovery mechanism from congestion control mechanism. The design of Wireless-aware SCTP considers simultaneously the nature of wireless losses

(random or bursty losses), the energy conservation for battery powered ad hoc nodes, and the goodput performance.

In this work, we also study how to overcome limitations of congestion control of standard SCTP when multi-homing feature is used. The per association congestion control structure makes it unfeasible to extend standard SCTP for load sharing, load balancing and other multi-homing applications. In this work, we identified and analyzed limitations of the congestion control mechanism of standard SCTP in the context of using multiple paths simultaneously. We propose an Independent per Path Congestion Control framework for standard SCTP to overcome these limitations. SCTP with independent per path congestion control not only overcomes existing flaws of standard SCTP in supporting the multi-homing feature but also demonstrates its capability in bandwidth aggregation under various network conditions.

Dedication

To my wife and son

Acknowledgments

I am sincerely grateful to my advisor Tarek Saadawi for giving me the privilege and honor to work with him over the last 3 years. Without his constant support, insightful advice, infinite patience and continuous encouragement, this thesis would not be possible. I always feel that I am the luckiest student in the world. Professor Saadawi not only teaches me how to conduct excellent research, but also sets a perfect model of decent person for me to follow.

I would also like to thank Professor Myung Lee for giving valuable inputs to my work, spending countless hours in inspiring discussions, sharing his visions of the fields, and providing his continuous encouragement throughout my PhD studies.

Furthermore, I would like to thank my friends and fellow students at the City University of New York for participating in numerous discussions and providing valuable comments on various aspects of this work. I am especially indebt to Zhengliang Yi, Chunhui Zhu, Yong Liu, Xuhui Hu and Dongmei Yu for being absolutely awesome friends and their countless help.

This work would not be possible without the financial aid from the Graduate Center of The City University of New York through the Robert E. Gilleece Fellowship, and the support from U.S. Army Research Laboratory under the Collaborative Technology Alliance (CTA) program in Communications and Networking.

Last, but not least, I would like to thank my parents for their love and support of my studying abroad, my wife - Chunyue Liu - for her love, sacrifice and full-hearted support of my work, and my son - Brian T. Ye - for the happiness he brings to the family.

Contents

List of Tables

List of Figures

Chapter One

1 Introduction

The ad hoc network is a desirable network form in Emergency Telecommunications, Battlefield and Intelligence Communication Systems etc. Unlike traditional wireless cell network, the ad hoc network does not require fixed infrastructure like base stations; a node in ad hoc networks can either be a host or a router that forwards packets for other nodes. Infrastructureless, easy to deploy are major advantages of using ad hoc networks. However, frequent change of network topology due to node movement, limited power supply due to the use of battery, and network competition due to the use of radio technology represent challenges in network design and protocol development. A Mobile Ad-hoc Networks (MANET) working group [1] was formed in IETF to standardize routing protocols in ad hoc networks. Extensive work has been conducted in literature on ad hoc routing and media access technologies that fit such environment. New algorithms on ad hoc routing and new mechanisms for media access control emerge almost every day. However, their impact upon transport protocols is far from well studied.

Much work has been done on TCP optimization in wireless cell environment [2-7], and they usually assume a base station exists between wired and wireless network (one-hop wireless link). Thus the base station is either used to hide the wireless link from transport protocols (e.g. TCP) or as a splitting point where one TCP connection (for wired part) stops here and another TCP connection starts here over the wireless link. The unavailability of such kind of fixed base station in wireless ad hoc network makes the

proposed transport layer solutions for wireless cell networks almost unfeasible in ad hoc network. In addition, as discussed in [8], no appropriate loss-recovery mechanism, which is able to handle wireless losses efficiently based on their nature of loss (wireless random loss and wireless bursty loss), has been developed for TCP. TCP is a dominating transport protocol of Internet. TCP is byte-oriented transport protocol and provides strict in-order delivery service to upper layer. However, TCP has its limitations. Application layer has to mark the boundary of each message because of the byte-oriented feature of TCP. The strict in-order delivery feature of TCP sometimes makes TCP a good head-of-line blocking transport protocol – segments sent later cannot be delivered to application layer without the successful delivery of previously sent segments although there is no sequence requirements for these segments from application layer.

Stream Control Transmission Protocol (SCTP) [9], which was originally designed to transport IP Telephony signaling in IP network, is becoming a good candidate of next generation transport layer protocol. As one of the "10 Hottest Technologies of Year 2001" by Telecommunications Magazine [10], SCTP is superior to TCP or UDP in many aspects, especially in the network level reliability, which is promising in situations of emergency operation or in battlefield. SCTP allows a multi-homed host to bind multiple IP addresses to a single SCTP association. If one destination interface is unreachable or fails, SCTP can smoothly switch traffic to another destination interface without the assistance from application layer. The multi-homing feature increases the availability of a remote endpoint. Applications will benefit from this feature if a transport protocol supports this kind of network level reliability. In addition, four-way handshake and

cookie mechanism make SCTP more robust to in the middle attacks, such as the SYN-attack to TCP.

Although SCTP is superior to TCP or UDP in many aspects, its congestion control mechanism is far from perfect in ad hoc networks. Congestion control forms the core of a transport protocol in providing efficiency and fairness. SCTP bases its congestion control algorithms on TCP's congestion control algorithms and extensions. As mentioned above, TCP even has difficulties to perform well in one-hop wireless cell network, not to mention the multi-hop wireless ad hoc network where frequent network topology change may occur due to node movement. The ad hoc environment introduces higher link bit errors than in wired networks, and because of mobility, link failures may even result in aborted connections under extreme cases such as network partition. A transport protocol will see more non-congestion packet losses and larger range of round trip time in ad hoc environment than in wired network. There is no doubt that SCTP will suffer similar problems as TCP in ad hoc networks since they use similar congestion control mechanism. Moreover, multi-home feature of SCTP represents another challenge in the design of congestion control mechanism. The standard SCTP does not support multi-homing applications such as load sharing and load balancing. If these kinds of multi-homing applications are to be supported, corresponding modifications to standard SCTP's congestion control mechanism are needed.

In this work, we focus on how to improve SCTP congestion control performance in ad hoc networks. Our work introduces a new form of congestion signal into SCTP, investigates how to overcome wireless errors in ad hoc network and innovative congestion control framework for multi-homed hosts. Although solutions are developed

under the context of SCTP, they should be easily extended for other transport protocols.

1.1 Research Problems

The goal of this thesis is to investigate innovative congestion control mechanism for SCTP to make it a suitable transport protocol for reliable data transmission in wireless ad hoc networks. In order to make this goal manageable, we identified a number of subproblems that naturally lead to the solution of the main problems.

- Conduct extensive performance study of SCTP in wireless ad hoc networks and identify its limitations in wireless ad hoc networks;
- Investigate mechanism to differentiate wireless losses from congestion losses for standard SCTP;
- Study appropriate loss-recovery mechanism for standard SCTP which is able to handle wireless losses based on their nature, random loss or bursty loss, and it should be energy efficient;
- Investigate novel congestion control framework for reliable data transmission between multi-homed hosts.

1.2 Solutions

First, we study the performance of SCTP congestion control in wireless multi-hop networks through extensive simulations. The goal is to see how it interacts with ad hoc routing protocols and IEEE 802.11 based MAC layer and identify its limitations in such environment.

Second, we introduce an alternate congestion signal, Explicit Congestion Notification (ECN) into standard SCTP to make SCTP ECN-capable. Then we study how

to optimize the congestion window reduction mechanism of ECN-capable SCTP to make it throughput efficient.

Third, we decouple loss-recovery mechanism from congestion control mechanism in standard SCTP and develop novel loss-recovery algorithms that are design specifically for wireless losses based on their nature.

Finally, we develop and design a novel Independent per Path Congestion Control framework for SCTP association between multi-homed hosts.

1.3 Contributions

This work makes the following contributions:

- Better understanding of the interactions of SCTP, Dynamic Source Routing protocol and IEEE 802.11 based MAC layer. Through thorough studies, we identify limitations of SCTP in such environment.
- Introduces Explicit Congestion Notification mechanism into standard SCTP. Considering the lossy wireless ad hoc environment and to be bandwidth efficient, the ECN-capable SCTP does not use the Congestion Window Reduction chunk that corresponds to the Congestion Window Reduction (CWR) flag in ECN-capable TCP.
- Studies different congestion window reduction policies upon the throughput performance of SCTP associations in a network with single bottleneck link. We identify there exists an optimal value of lower boundary of the congestion window in order to maximize throughput performance of SCTP, and develop a simple and practical method to achieve this optimal value.

- Reveals a flaw in the design of Random Early Detection method. The low pass filter used in RED method is able to accommodate quick fluctuations of network load, however, it is rather slow in reflecting real network load after persist network congestion which results in multiple window reductions of SCTP sources, thus network underutilization is resulted in. We propose a remedy that considers both actual queue size and average queue size in determining whether there is network congestion or not.
- Develops a variant of SCTP, Wireless-aware SCTP. Wireless-aware SCTP is able to differentiate wireless losses from congestion losses, and handle wireless losses based on their nature of loss. Wireless-aware SCTP is based on ECN-capable SCTP, and uses ECN as congestion signal.
- Demonstrates limitations of standard SCTP in supporting multi-homing applications such as load sharing and load balancing and design a novel Independent per Path congestion control framework for standard SCTP.

1.4 Dissertation Overview

The rest of the dissertation is structured as follows.

Chapter 2 presents an overview of background material and some of the related work. In Chapter 3, we conduct performance evaluation of standard SCTP in multi-hop wireless ad hoc network, and identify its limitations in such environment. The impact of various IEEE 802.11 parameters on SCTP congestion control parameters is simulated and their effect on throughput as well as its performance is studied. Further more, a new algorithm is proposed to overcome the "small window syndrome" resulting from the MAC layer when the SCTP receiver side window is small.

In Chapter 4, we study how to introduce another kind of congestion signal, Explicit Congestion Notification (ECN), into standard SCTP, and study how to optimize the performance of ECN-capable SCTP. An ECN-capable SCTP source needs to adjust its congestion window when receiving ECN messages. We find the optimal value of the congestion window for an ECN-capable SCTP source in response to ECN messages, and develop a simple and practical method to achieve this optimal congestion window. Both simulation results and analysis are provided to support the effectiveness of the proposed ECN mechanism for SCTP. The simplified method in achieving the optimal congestion window is attractive because the total goodput performance of SCTP associations or the bottleneck link utilization is not sensitive to the window reduction policies when the network load is heavy. We find that using complicated methods to fine-tune SCTP or TCP's congestion window in response to congestion indications may not be worth the increase in complexity of the protocol.

Chapter 5 presents a new variant of standard SCTP – Wireless-aware SCTP. Based on the ECN-capable SCTP discussed in Chapter 4, Wireless-aware SCTP is able to differentiate wireless losses from congestion losses. Moreover, Wireless-aware SCTP is able to handle wireless losses based on their nature of loss, random wireless loss or bursty wireless loss. In this chapter, we also study different retransmission schemes as well as different retransmission timer back off schemes upon the performance of Wireless-aware SCTP.

Chapter 6 studies a novel congestion control framework for standard SCTP to enhance its performance when multi-homing feature is used. We find that simultaneously using multiple paths is inherent in multi-homing applications. The per association congestion

control design in standard SCTP makes it unfeasible for SCTP to support multi-homing applications efficiently. We propose an independent per path congestion control framework for SCTP to overcome these limitations. The proposed Independent per Path Congestion Control (IPCC) SCTP can be used for various applications of the multi-homing feature when appropriate data-striping algorithms are used. The proposed IPCC SCTP modifies only the behavior of standard SCTP source endpoint, and introduces no overhead to networks.

Chapter 7 concludes the dissertation, discusses some of the future work and lists publications we have on this work.

Chapter Two

2 Related Work

In this chapter, we review related work in network congestion control as well as SCTP congestion control. Section 2.1 gives a brief description of the basic concepts in network resource management and demonstrates the importance of congestion control mechanism in a network. Section 2.2 reviews TCP congestion control. Since SCTP bases its congestion control algorithms on that of TCP's, it is necessary to understand congestion control in TCP. Section 2.3 first gives a brief review of SCTP protocol design, and then it focuses on its congestion control mechanism.

2.1 Congestion Control Basics

In general, network resource management includes three different aspects: *flow control*, *congestion control* and *congestion avoidance*. These three aspects are related to each other; however, they play distinct roles in the network resource management.

Flow control is a concept about how to use the buffer space at a destination. If there is no *flow control*, a fast source may easily overflow a slow destination's buffer space. So, a *flow control* scheme is designed to solve the incompatibility between a sender and a receiver, and to prevent the destination from being flooded. Window-based and rate-based flow control schemes are two common schemes for flow control. In window-based scheme, a sender can only transmit a certain number of packets without further permission according to the value of the receiver side window. In rate-based scheme, the

receiver specifies a maximum rate (packets per second or bits per second) at which the source may send information.

Congestion control is used to help the network to recover from the congestion state. Congestion occurs when a network is overloaded or the resource demands exceed the network capacity. Congestion usually occurs in a network with incompatible links, i.e. fast incoming links may flood slow outgoing links. Thus a *congestion control* scheme is designed to solve the incompatibility between links. The side effects of network congestion are: packet losses, decrease of throughput and increase of network delay. Severe network congestion may even cause congestion collapse - the network is busy but almost does zero useful work.

The concept of *congestion avoidance* is credited to Jain et al. [11] in 1988. They pointed out that a *congestion avoidance* scheme allows a network to work in the region of low delay and high throughput, i.e. the "knee" region in Figure 1. The Congestion avoidance mechanism prevents the network from entering into congestion state, while the congestion control mechanism recovers the network from congestion state. This means the congestion control is used to bring the network back to the left side of "cliff" shown in Figure 1 if the "cliff" is crossed. So, the congestion avoidance is a preventive mechanism while the congestion control is a recovery mechanism. In this work, we focus only on congestion control and congestion avoidance.

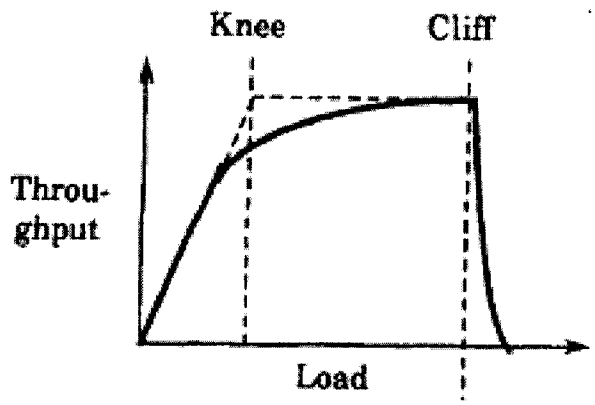


Figure 1. Throughput as a function of network load

Original TCP [12], which was based on the concepts first described by Cerf and Kahn [13], used flow control to control the use of buffer space at a receiver, and Go-Back-N retransmission model after a packet drop for reliable delivery. However, only flow control is not sufficient for TCP to be robust, efficient and fair.

In 1984, Nagle [14] first reported "congestion collapse", which refers to a state of network in which the network is increasingly busy but little useful work is getting done. Nagle explained, in heavily loaded pure datagram networks with end-to-end retransmission, as switching nodes becomes congested, the round trip time increases and exceeds the retransmission interval, thus every packet is being transmitted several times and throughput is reduced to a small fraction of normal. Nagle [14] proposed to use the ICMP Source Quench messages to solve the congestion problem. Nagle suggested that a router send a source quench message when about half the buffering space is exhausted instead after more than one packet has been dropped, and a TCP sender behave as if the window size is zero until 10 ACKs have been received. He argued that it was undesirable and a waste of bandwidth for any system to base its congestion control on discarding of packets, and it was susceptible to congestion collapse under heavy load. This becomes

the design principle for router based congestion avoidance mechanism, such as Explicit Congestion Notification (ECN).

In October 1986, the first Internet congestion collapse occurred. From then on, congestion control for TCP has been being extensively studied. A congestion control scheme usually includes two components: the congestion feedback and the reaction at the source. From the evolution of TCP congestion control, we can see clearly that all modifications are related to these two components.

2.2 TCP Congestion Control

In 1986, Jain [15] introduced a timeout-based congestion control scheme for window flow-controlled network, which was also known as *CUTE* scheme (*c*ongestion control *u*sing *t*imeouts at the *e*nd-to-end layer). Traditionally, retransmission timeouts are used to provide the reliability for end-to-end transmission. However, it is also an indication of severe state of network congestion if proper flow control mechanism has already been used. Simply retransmit the lost packet and without any other adjustment of the transmission rate of a data source may cause congestion collapse as we discussed in the previous section. Jain proposed to reduce the window to one packet when timeout occurs, and to increase the window by one every round trip delay interval.

Later in 1980s, Jain et al. published a series of papers [16, 17] in congestion avoidance. The famous Additive Increase, Multiplicative Decrease (*AIMD*) policy was developed during this period, and it was incorporated into TCP later. Jain et al. found that simple additive increase and multiplicative decrease algorithm satisfies the sufficient conditions for convergence to an efficient and fair state regardless of the starting state of the network.

It was in 1988 that TCP first started to have its own congestion control algorithms. Jacobson [18] proposed several modifications to TCP that increased its stability during periods of heavy loss and reduced the number of spurious retransmissions. The proposed modifications included round-trip-time variance estimation, exponential retransmit timer back off, slow start, dynamic window sizing on congestion, and fast retransmit. Jacobson suggested that TCP flows should obey a 'conservation of packets' principle: A new packet isn't put into the network until an old packet leaves.

Original TCP, RFC 793, took two times of the estimated round-trip-time as the retransmission timeouts (RTO). Jacobson estimated that this RTO value could adapt to loads of at most 30%, which was too low and might cause the congestion collapse when network was heavily loaded. If RTO is too small, TCP will retransmit segments unnecessarily. Both average round-trip-time and the variance estimation are suggested in calculating the RTO. Following equations are used to estimate the RTO [19].

$$Err = M - A$$

$$A \leftarrow A + gErr$$

$$D \leftarrow D + h(|Err| - D)$$

$$RTO = A + 4D$$

Where M is the currently measured RTT, A is the smoothed RTT (an estimator of the average) and D is the smoothed mean deviation. Err is the difference between the measured value just obtained and the current RTT estimator. The gain g is for the average and is set to 1/8 (0.125). The gain for the deviation is h and is set to 0.25. The larger gain for the deviation makes the RTO go up faster when the RTT changes.

Exponential retransmit timer back off algorithm is used to instruct a TCP sender to double its retransmission timer each time the same packet is retransmitted. Exponential back off slows down the retransmission rate, thus the network has more chances to handle other traffic. This increases the stability of the network.

Before the proposed usage of slow start, original TCP [12] injected entire window of packets into the network when a TCP connection starts. Slow start algorithm is used to probe slowly the availability of network resources. The initial slow start window is one *MSS* (maximum segment size). The congestion window (*cwnd*) is increased by one for every new incoming acknowledgment during the slow start phase. A TCP source sends the minimum of the receiver's advertised window and *cwnd*. Here, we can see that TCP uses both flow control and congestion control. Receiver's advertised window is used for flow control. *Cwnd* is used for congestion control. Slow start mode stops when the *cwnd* crosses the slow start threshold (*ssthresh*), then congestion avoidance mode is entered.

Dynamic window sizing on congestion (congestion avoidance) allows the network to stay at the knee point, high throughput and low delay. *AIMD* algorithm is the core of the congestion avoidance algorithm for TCP. TCP uses packet losses as indications of congestion. Congestion window is basically cut to half the previous value at the time of a packet loss. However, Tahoe TCP differs later TCP versions in the implementation of this algorithm, which is illustrated in the fast retransmit algorithm. Tahoe TCP cuts *cwnd* to one when a packet loss is detected by three duplicate acknowledgments, and reduces *ssthresh* to half the previous *cwnd* value. Later TCP versions set both *cwnd* and *ssthresh* to half the previous *cwnd* value when a packet loss is detected by three duplicate acknowledgments.

A duplicate acknowledgment in TCP represents an out of order segment received by the receiver. Since TCP does not know whether a duplicate ACK is caused by a lost segment or just a reordering of segments, it waits for a small number of duplicate ACKs to be received. It is assumed that if there is just a reordering of the segments, there will be only one or two duplicate ACKs before the reordered segment is processed, which will then generate a new ACK. If a threshold (3 in current TCP) of duplicate ACKs are received in a row, it is a strong indication that a segment has been lost. TCP then performs a retransmission of what appears to be the missing segment, without waiting for a retransmission timer to expire. This is so-called fast retransmit in TCP.

With Jacobson's modifications in 1988, TCP was implemented in 4.3 BSD Unix in 1988 and became known as Tahoe TCP. Tahoe TCP was the first TCP implementation having congestion control. It included slow start, congestion avoidance and fast retransmit. Fast retransmit algorithm was received continuous modification in later TCP implementations. This will be covered in the following paragraphs.

Reno TCP refers to TCP with the earlier algorithms plus fast recovery, first implemented in 4.3 BSD in 1990. In Reno TCP implementation, fast recovery [20] was included in the modified fast retransmit algorithm. It is thought that a single packet loss does not represent severe network congestion, thus resuming from slow start seems too conservative for a single packet loss. Tahoe TCP enters slow start after the fast retransmit, thus the "pipe" between the source and the receiver becomes empty first, then it is refilled by slow start until half previous effective window size is reached. Moreover, the receipt of the duplicate ACKs tells TCP more than just a packet has been lost. Since the receiver can only generate the duplicate ACK when another segment is received, that

segment has left the network and is in the receiver's buffer. That is, there is still data flowing between the two ends, and TCP does not want to reduce the flow abruptly by going into slow start. In other words, the 'conservation of packets' principle should not be broken. In Reno TCP, the fast retransmit and fast recovery algorithms are usually implemented together as follows [21].

1. When the third duplicate ACK in a row is received, set *ssthresh* to one-half the current congestion window, *cwnd*, but no less than two segments. Retransmit the missing segment. Set *cwnd* to *ssthresh* plus 3 times the segment size. This inflates the congestion window by the number of segments that have left the network and which the other end has cached.
2. Each time another duplicate ACK arrives, increment *cwnd* by the segment size. This inflates the congestion window for the additional segment that has left the network. Transmit a packet, if allowed by the new value of *cwnd*.
3. When the next ACK arrives that acknowledges new data, set *cwnd* to *ssthresh* (the value set in step 1). This ACK should be the acknowledgment of the retransmission from step 1, one round-trip time after the retransmission. Additionally, this ACK should acknowledge all the intermediate segments sent between the lost packet and the receipt of the first duplicate ACK. This step is congestion avoidance, since TCP is down to one-half the rate it was at when the packet was lost.

Reno's fast recovery algorithm is optimized for the case when a single packet is dropped from a window of data. However, when there are multiple packet losses in a window of data, Reno TCP suffers performance problems as reported in [22, 23]. The

main reason for the performance problems is that Reno goes out of fast recovery when a 'partial ACK' (that acknowledges some but not all of the packets that were outstanding at the start of the fast recovery period) is received, thus Reno TCP is vulnerable to several fast retransmits if multiple packets are lost in a window of data. Congestion window is basically cut to half each time the fast retransmit algorithm is executed, thus throughput performance is degraded.

In 1999, Floyd et al. proposed a modification to Reno, called NewReno [24]. NewReno was based on Hoe's work in [25]. NewReno stayed in fast recovery until all packets, which were outstanding when fast recovery was initiated has been acknowledged. However, due to the limitation of the cumulative acknowledgment scheme used in Reno/NewReno TCP, the sender can retransmit at most one dropped packet per round-trip time. This is because a TCP sender does not know exactly those buffered segments at the receiver's buffer.

In order to retransmit more efficiently when multiple drops occur in a window of data than Reno TCP, SACK TCP [26], which was based on the TCP extension [27] proposed by Jacobson et al. in 1988, was proposed by Mathis et al. in 1996. By using selective acknowledgments, the data receiver can inform the sender about all segments that have arrived successfully, so the sender need retransmit only the segments that have actually been lost. After extensive simulation-based comparisons of Tahoe, Reno, NewReno and SACK TCP, Fall et al. [23] proposed to use SACK TCP in the Internet.

It was in August 1996 that Forward Acknowledgment [28] (FACK) was proposed by Mathis et al. to refine the implementation of SACK TCP. Forward Acknowledgment uses TCP SACK option to keep updated a state variable *snd.fack* at a TCP sender of the

highest sequence number received by a TCP receiver. This is the origin of the name "forward acknowledgment". Using the *snd.fack*, TCP can measure precisely the amount of outstanding data, thus TCP can regulate the amount of data outstanding in the network to be within one MSS of the current value of *cwnd*. In addition to use three duplicate ACKs, FACK uses reassembly queue at the receiver to determine if fast recovery should be entered. This mechanism is useful for small window TCP when multiple segments are lost in a window a data.

Independent to the development of the NewReno, SACK and FACK modifications, a new version of TCP - TCP Vegas - was introduced by Brakmo et al. [29] in 1995. As we know, TCP [18] uses packet losses as indication of congestion, and the congestion window is increased until a packet loss or timeout occurs, whichever first. From this point of view, we can see that previous versions of TCP are reactive, i.e. react to congestion only when congestion occurs. TCP Vegas tries to reduce the unnecessary retransmission by predicting the incipient congestion. Thus TCP Vegas is proactive. Three modifications were made in TCP Vegas to increase throughput and decrease losses: retransmit, congestion avoidance and slow start mechanisms. TCP Vegas uses a fine-granular RTO timer, thus it can make more timely decision of retransmission than TCP Reno, which uses a coarse-granular RTO timer. A linear increase/decrease congestion window adjustment policy is used in the congestion avoidance algorithm. TCP Vegas compares the difference between the expected throughput (the ratio of the window size to the smallest RTT) and the actual throughput (the ratio of the window size to the latest RTT). If the difference is less than the threshold α, the congestion window is increased linearly per RTT, if the difference is larger than another threshold β, then the

congestion window is decreased linearly per RTT. The congestion window is unchanged if the difference falls between α and β. The increase/decrease policy is based on the following assumption: The farther away the actual throughput gets from the expected throughput, the more congestion there is in the network, which implies that the sending rate should be reduced. On the other hand, when the actual throughput rate gets too close to the expected throughput rate, the connection is in danger of not utilizing the available bandwidth. The congestion estimation mechanism using the expected throughput and actual throughput is also incorporated into the slow start algorithm in TCP Vegas. TCP Vegas allows exponential growth of congestion window every other RTT. In between, the congestion window stays fixed so a valid comparison of the expected and actual rates can be made. When the actual rate falls below the expected rate by the threshold α, Vegas changes from slow start mode to linear increase/decrease congestion avoidance mode. It is reported in [29] that Vegas achieves between 37 and 71% better throughput on the Internet, with one-fifth to one-half the losses, as compared to the implementation of TCP in the Reno distribution of BSD Unix. Other recent papers [30, 31, 32] confirmed the advantages of TCP Vegas in the performance of throughput, stability and fairness. However, Bolliger's work [33] in 1999, which was conducted on a set of Internet hosts in North America and Europe for six months, showed that FACK-based TCP outperformed Reno and Vegas TCPs. This conclusion contradicts to earlier work in [29, 30, 31, 32].

2.3 Overview of SCTP and Its Congestion Control

Stream Control Transmission Protocol (SCTP) [9, 34] is a message based new transport protocol of Internet standards track. It was motivated by weaknesses for TCP in dealing with telephone call control signaling over IP-based packet network, and was

standardized by the IETF as a proposed general-purpose transport protocol in October 2000. Figure 2 gives the position of SCTP in IP reference model. Telephony signaling has rigid timing and reliability requirements that are often set forth through government regulations and must be met in order for the phone service provider to be qualified. However, neither TCP nor UDP can meet these requirements because of their limitations.

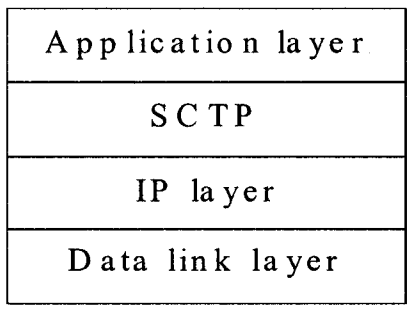


Figure 2. SCTP in IP reference model

TCP limitations [34]:

- TCP provides both reliable data transfer and strict order-of-transmission delivery of data. Telephony signaling application often need reliable message transfer but not globally strict sequence maintenance. What is often more desirable for telephony signaling is a partial ordering of the data, such as maintaining ordering only within some subflows of the data. The strict sequence maintenance in TCP not only makes such type of partial ordering of the data impossible, it can also potentially introduce unnecessary delay to the overall data delivery service. Loss of a single TCP segment can block delivery of all subsequent data in a TCP stream until the lost segment is delivered. This is sometimes called **head-of-line blocking**.
- The byte-oriented nature of TCP is often an inconvenience to the messaged telephony signaling. Applications must add their own record marking to delineate their

messages, and they must make explicit use of the push facility to ensure that a complete message is transferred in a reasonable time

- TCP has no built-in support for multi-homed IP hosts. This brings an immense challenge to the system designers who want to build link or path-level redundancy. Without link or path-level redundancy, the data transfer service in a network can become vulnerable to link failures. This is often unacceptable for a telephony signaling network, because providing highly available data transfer capability is one of its primary requirements.
- For telephony applications, especially commercial phone services, security against malicious attacks intended to cause failure or interruptions to the services is often a top priority. But TCP is known to be relatively vulnerable to some denial of service attacks, such as the SYN flooding attack.

UDP limitations:

- UDP provides an unreliable data transfer service to the application. An application cannot know whether data sent to a peer application is received by the other end or not.
- UDP has no build in congestion management mechanism to detect congestion and to recover from congestion.

SCTP overcomes limitations existing in TCP and UDP; it also has its unique features. Three new features – multi-streaming, multi-homing and enhanced security – are most important to SCTP. Figure 3 shows how multiple streams are used in a single SCTP association. In this case, host A has three outbound streams and host B has 3 corresponding inbound streams; host A has one inbound stream and host B has

corresponding outbound stream. For simplicity, one can imagine that each of these streams is like a TCP connection. These streams are independent to each other; there is no sequence requirement between them. Multi-streaming is extremely useful when application layer has multiple blocks of data to send and does not require sequential delivery of these blocks of data. If TCP is used, the application layer will have to send one block of data after another. By using the multi-streaming feature of SCTP, the application layer can use one stream for each block of data, and send these blocks of data simultaneously. SCTP endpoints can negotiate how many streams they want to setup during the association setup period.

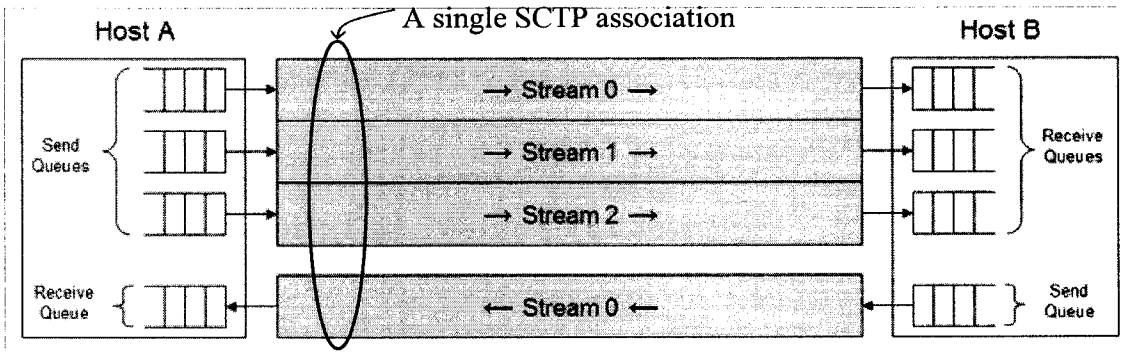


Figure 3. SCTP multi-streaming

Figure 4 gives an example of using multi-homing in wireless networks. In this case, each host has two wireless interfaces. Each wireless interface can use different radio technologies or operate at different radio frequencies. Multiple interfaces can provide network level redundancy for a host. However, without the transport layer support of multiple interfaces, this feature cannot be fully utilized. As shown in Figure 5, when TCP is used as the transport layer, host A and host B can use only one interface at a time due to the limitation of TCP (TCP socket allows only one source IP address and one

destination IP address). If the interface in use fails, application layer has to start another TCP connection by using the other available interfaces. Unlike TCP, SCTP allows multiple interfaces involved in an SCTP association. Application layer can use all the available destination addresses to reach remote endpoint. By using SCTP, application traffic can be smoothly switch to another destination interface without having to restart another SCTP association if the current destination interface is unreachable.

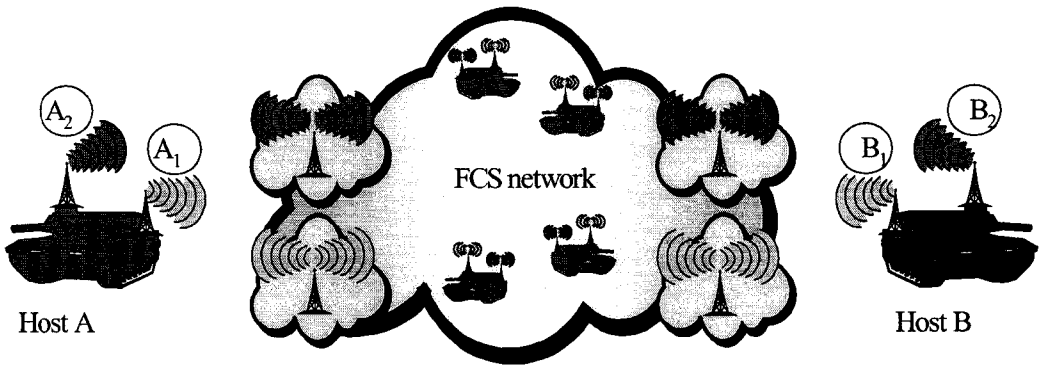


Figure 4. Multi-homing example in wireless networks

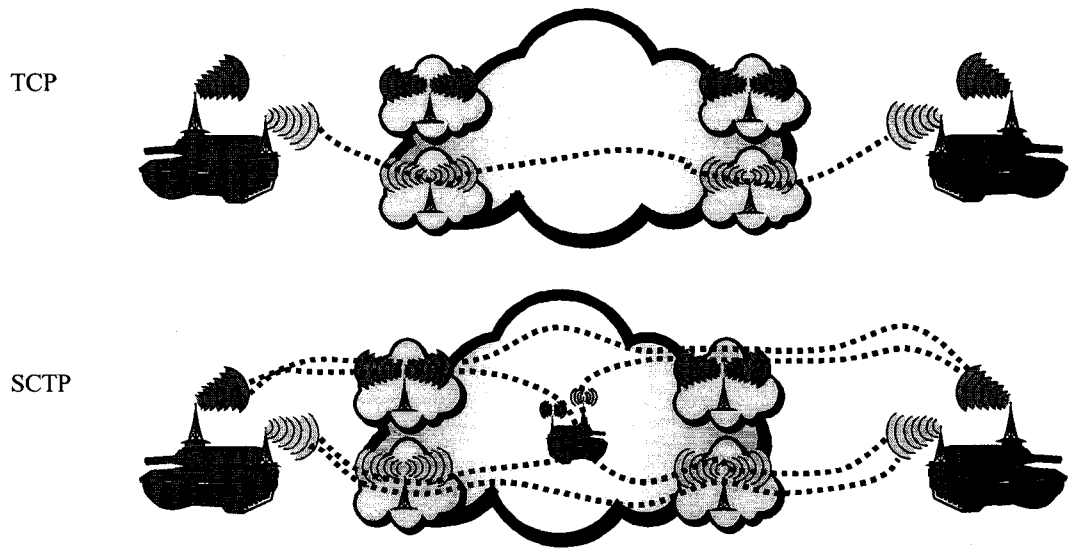


Figure 5. TCP vs. SCTP for multi-homed hosts

In order to fight against flooding and masquerade attacks, SCTP uses four-way handshake, INIT, INIT ACK, COOKIE ECHO, and COOKIE ACK, to setup a logical connection between two endpoints. No memory is allocated, i.e. the Transport Control Block (TCB) is not built, at the SCTP server until COOKIE ACK is received. This mechanism is able to efficiently avoid SYN-flooding that TCP faces. In addition, the use of Verification Tag in each SCTP packet prevents SCTP from in the middle attacks.

Next, we will give a brief overview of congestion control algorithms in SCTP. SCTP follows the same design rules as TCP on congestion control [35], and adopts enhancements on TCP, such as SACK and FACK etc. Like TCP, SCTP also includes slow start, congestion avoidance, and fast retransmit algorithms. In general, an SCTP endpoint begins its data transmission in slow start phase. When congestion window (*cwnd*) is greater than slow start threshold (*ssthresh*), it enters congestion avoidance phase. If packet losses occur, either fast retransmit or timeout handle procedures will be

executed. Which phase it will stay next depends on the values of *cwnd* and *ssthresh*. An endpoint enters the slow start phase when it begins data transmission or after a sufficiently long idle period. The initial congestion window (*cwnd*) for SCTP is recommended to be 2*MTU. If an incoming SACK advances the cumulative TSN ACK point and the current window is full utilized, the *cwnd* is increased by at most the lesser of 1) the total size of the previously outstanding data chunks acknowledged, and 2) the destination's path MTU. This is different from TCP in the slow start phase. TCP increases the *cwnd* by MSS whenever a new ACK comes in. When the *cwnd* is equal to the slow start threshold (*ssthresh*), SCTP is in slow start phase, while it is optional for TCP to be either in the slow start phase or in the congestion avoidance phase.

The congestion avoidance phase starts when the *cwnd* is greater than the slow start threshold. And the AIDM window adjustment algorithm is used in this phase. During this phase, the *cwnd* is increased by one MTU per RTT when the sender has the *cwnd* or more bytes of data outstanding for the corresponding transport address. In other words, the *cwnd* is increased by one MTU only when the current window is fully utilized and window size or more data is acknowledged. When a packet loss is detected by the fast retransmit algorithm, the *cwnd* is cut to half the previous value, but no less than two MTU.

Fast retransmit is triggered when the fourth missing report of a TSN is received (required by the RFC 2960). This implies that the minimum congestion window needed to trigger fast retransmission is 5*MTU, while in TCP, the minimum congestion window for fast retransmission) is 4*MTU (three duplicate ACK triggers fast retransmission). Moreover, only new missing reports are counted, and duplicate SACK is simply

discarded. After retransmission of the missing TSN(s), the *ssthresh* is set to the maximum (cwnd/2, 2*MTU) and the *cwnd* is equal to the *ssthresh*. Basically, fast retransmission causes *cwnd* to be cut in half. There is no explicit fast recovery algorithm specified in SCTP. But the protocol parameter *Max.Burst* is used after the fast retransmit to avoid flooding the network. *Max.Burst* limits the number of SCTP packets that may be sent after processing the SACK, which acknowledges data chunks that has been fast retransmitted. Like Reno TCP, the fast retransmit algorithm of standard SCTP has performance problems if multiple packets are lost in a window of data. A similar solution specified in NewReno can be used for SCTP to solve this problem.

SCTP is receiving more and more studies in the literature. Jungmaier et al. [36] evaluated SCTP performance in their test bed. They showed that SCTP traffic has the same impact on TCP traffic as normal TCP traffic, and SCTP and TCP connections share resources equally. This means SCTP is neither more nor less aggressive than TCP.

Brennan et al. [37] used simulation studies to explore the behavior of SCTP congestion control. They compared the Reno TCP and several possible variants of SCTP congestion control. They proposed three modifications to the fast retransmit algorithms of SCTP. First, they proposed that Gap Ack Blocks, which was an option at that time in RFC 2960, must be included in the SACK chunk in order to recover efficiently after a packet loss. Second, they proposed that lost packets should be fast retransmitted immediately without considering the current congestion window size, while SCTP was permitted to transmit a packet only if congestion window allows at that time. The third one was about how many times a packet can be fast retransmitted. They suggested that a

packet should be fast retransmitted only once in order to avoid halving the congestion window multiple times when multiple packets were lost in a window of data.

Caro et al. [38] developed a two-level threshold recovery mechanism for SCTP to explore the multi-homing feature of SCTP. They suggested that traffic should be switched to alternative path before the failure of the primary path if high throughput is pursued. However, they assumed there was no failure at the alternative path, and their solution is parameter-sensitive.

In the current specification, RFC 2960, alternative destination addresses are used mainly for retransmission or fail over. Iyengar et al. [39] studied the changeover behavior of SCTP, i.e. switch traffic to alternative path while the primary path is active. TCP-unfriendly growth of the sender's congestion window was identified during certain changeover conditions. The problem was caused by the limitation of current SCTP's inability to distinguish between SACKs for transmission or retransmission. The Rhein algorithm is proposed to solve this problem. However, this solution cannot eliminate the unnecessary retransmissions caused by the changeover.

Chapter Three

3 SCTP Performance in Ad Hoc Networks

In this chapter, we evaluate SCTP throughput performance over IEEE 802.11 wireless LAN protocol using different SCTP receiver side window sizes and the different number of hops between the source and the destination. Earlier work in [40, 41] has shown the limitations of TCP performance over IEEE 802.11 wireless LAN protocol. Through thorough simulation studies, we find the throughput of SCTP degrades when the number of hops increases; Increasing the window size does not help to increase the throughput of SCTP; On the contrary, it amplifies the hidden node problem and the exposed node problem at MAC layer, worsening the throughput. The virtual carrier sense does help to improve throughput when the number of hops is small, but this is not true when the number of hops increases. Packet loss may cause SCTP timeout and degrade its performance. It is rather unfair for those SCTP associations with small window size if the packet loss is not caused by network congestion. We call this phenomenon "small window syndrome". An algorithm is proposed to overcome this problem, and the simulation results show that the SCTP throughput is improved.

A brief introduction of issues related to IEEE 802.11 MAC layer is given in section 3.1. In section 3.2, the instability problem and the analysis are reported. The unfairness problem and the analysis are presented in section 3.3. The proposed algorithm for "small window syndrome" and simulation results are presented in section 3.4.

3.1 Related Issues in IEEE 802.11

The fundamental access method defined in the IEEE 802.11 [42] is a distributed coordination function (DCF) known as the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). The CSMA mechanism works as follows: in order to transmit, a station shall sense the medium first. If the medium is busy, then the station defers its transmission to a later time. If the medium is determined to be free, then the station is allowed to transmit. The chances of collision still exist using this mechanism because multiple stations may determine the medium to be free and transmit simultaneously. In order to minimize such collisions, the virtual carrier sense mechanism is used to distribute channel reservation information by announcing the impending use of the medium. For data frames with sizes greater than the Request To Send (RTS) threshold, control frames RTS/CTS (request to send/clear to send) exchange is required before data transmission. All the other stations hearing RTS or CTS shall update their network allocation vectors (NAV), which reflect the current state of the medium usage. Physical carrier sense and NAV are used together to determine whether the medium is idle or not. For data frames whose size is less than or equal to the RTS threshold, no RTS/CTS exchange is required before data transmission.

3.2 Instability Problem

We implement SCTP (sections 6 and 7 of RFC2960 with the modification in [43]) into the commercially available network simulation tool - OPNET [44]. Dynamic Source Routing (DSR) protocol [45] is used at the routing layer. The MAC layer uses the IEEE 802.11 based wireless radio with a bandwidth of 1 Mbps. A string topology (Figure 6) is

used in the simulation studies. The wireless LAN range is set to be 300m. The distance between any two neighboring nodes is equal to 250m, which allows a node to communicate directly solely to its neighboring nodes. Nodes are static (No mobility), because we want to focus on the interaction between SCTP and the IEEE 802.11 MAC layer. In all the simulation runs, SCTP data chunk size is set to 512 bytes, and the path MTU is 544 bytes. Delayed SACK is used. The initial *cwnd* is 2*MTU. There is only one SCTP association in the network, no background traffic. Node 0 is the source. Nodes 1 through 3 are selected as the destination respectively. Meanwhile, the SCTP receiver side window size and the RTS threshold of IEEE 802.11 are adjusted to show their effect upon the SCTP throughput. Large file transfer from the source to the destination (i.e. the source node always has data to send out) is assumed.

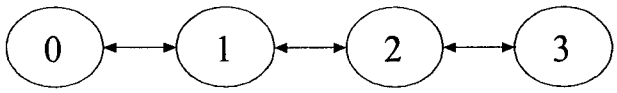


Figure 6. String Topology

Figure 7 shows the results for the simulation run between node 0 and node 1. As is shown in Figure 7 (a), when the window size is 2, no matter what the value of the RTS threshold is, SCTP association achieves the highest throughput (about 660Kbps) it can achieve. Although smaller value of the RTS threshold should lead to lower throughput (more control traffic), the difference is negligible in this case because of the short simulation duration. Since the receiver side window is 2*MTU and delayed SACK is used, SCTP actually acts like the stop-and-wait protocol. It starts sending two data chunks, then waits for the acknowledgment; after it receives the acknowledgment, it

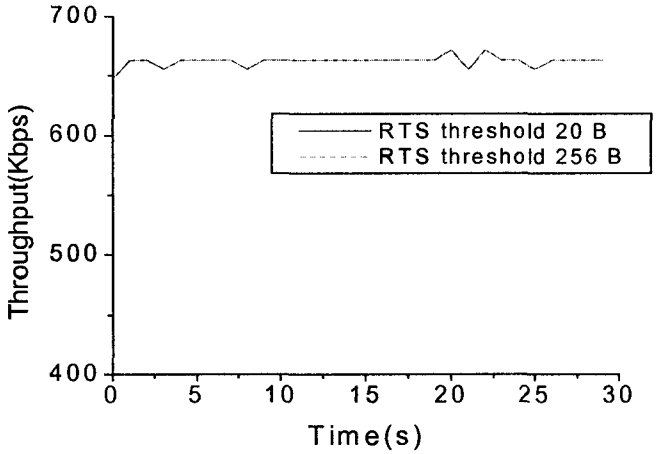
sends another two data chunks. Two nodes use the wireless channel in turn. When the transmission time dominates the end-to-end delay of the network, stop-and-wait protocol should achieve very good link usage efficiency. This condition is met in our simulation setting. Thus SCTP should achieve maximum throughput in this case. Our simulation results agree with the theory analysis.

In Figure 7 (b), we observe that SCTP throughput goes down to about zero several times during the simulation time when the RTS threshold is 256 bytes. And we also observe that each throughput degradation period is associated with a timeout at the SCTP source. By debugging the simulation carefully, we find that RTS frames collide with SACK chunks at the MAC layer, and the collision causes drop of SCTP data chunks. When the RTS threshold is 256 bytes, there is no RTS/CTS exchange before SACK transmission (SACK chunks are usually less than this threshold). Thus SACK chunks may collide with RTS frames sent by the sender for data chunks when both nodes sense the medium to be free and send simultaneously. When the window size is increased to 4 or more, forwarding SCTP data chunks and reversing SACK chunks will compete for the wireless channel. Whenever the sender has data to send, it sends RTS first to reserve the channel. If this RTS collides with SACK sent by the receiver, both endpoint will back off and retransmit later. The retry limit for SACK chunks is 7 (short retry limit), and it is 4 (long retry limit) for data chunks. Thus SACK chunks get more chances to win the competition. In our simulation setting, we find SCTP data chunks almost always fail the competition. Then link failure is reported to the DSR layer whenever a SCTP data chunk is discarded when the retry counter reaches the limit. The DSR then initiates route discovery procedure again to obtain a route to the destination node. Even the route

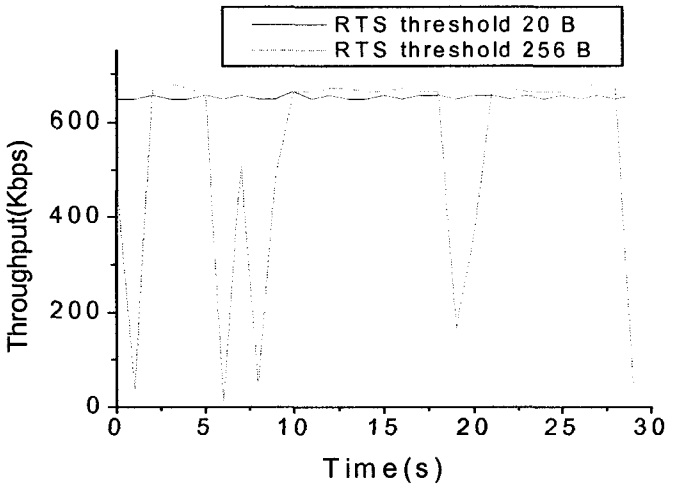
discovery succeeds before SCTP timer runs out; the source cannot retransmit the lost data chunk in advance when the window size is small (no fast retransmit). Thus SCTP times out, and starts from slow start again. This is the main reason for the throughput degradation.

When the RTS threshold is 20 bytes, the virtual carrier sense mechanism greatly reduces collision between the source and destination nodes, thus SCTP association achieves smooth throughput as shown in Figure 7 (b) and Figure 7 (c) for RTS threshold of 20 bytes.

In Figure 7 (c), we observe that the throughput of SCTP association (for the RTS threshold of 256 bytes) is zero for about 30 seconds. This is because of multiple packet losses in one window (SCTP data chunks collide with reverse SACK chunks), and thus SCTP association times out several times in order to retransmit lost packets. This dramatic throughput degradation is very rare when there is only one hop between source and destination nodes, and this does not always happen when we change the simulation seed. We purposely list it here to emphasize the importance of the RTS threshold to the performance of SCTP association when the number of hops is small.



(a) window = 2*MTU



(b) window = 4*MTU

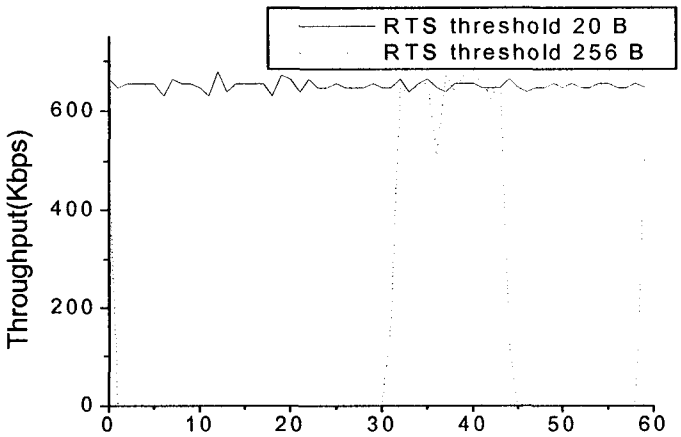
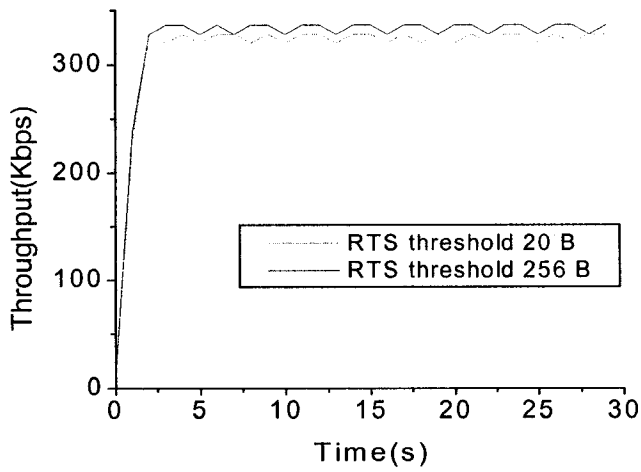


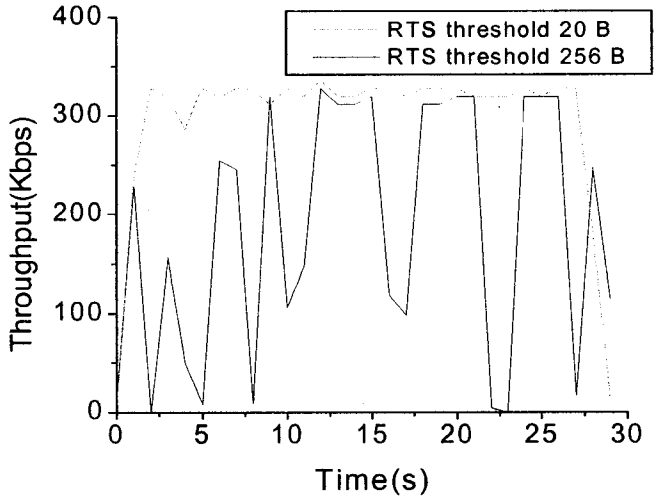
Figure 7. Throughput of one hop SCTP association

Simulation results for the SCTP association covering two hops (from node 0 to node 2) are reported in Figure 8. When the window is 2, similar as in the one hop case, there is only one-way traffic at a time in the network: either data chunks or SACK chunks are in transmission. The only difference is that node 0 and node 1 may compete for the channel when both of them have data to send. But since they are in the transmission range of each other, and they only compete with each other when both have SCTP data chunks to send, the chances of collision is greatly reduced because RTS/CTS exchange is used before data chunk transmission. So, SCTP association achieves rather smooth throughput when window size is 2*MTU. But with the number of nodes increases, more time is spent for control traffic in MAC layer with smaller RTS threshold. This is the reason that SCTP throughput for RTS threshold of 20 bytes is a bit lower than that for RTS threshold of 256 bytes when window size is 2*MTU.

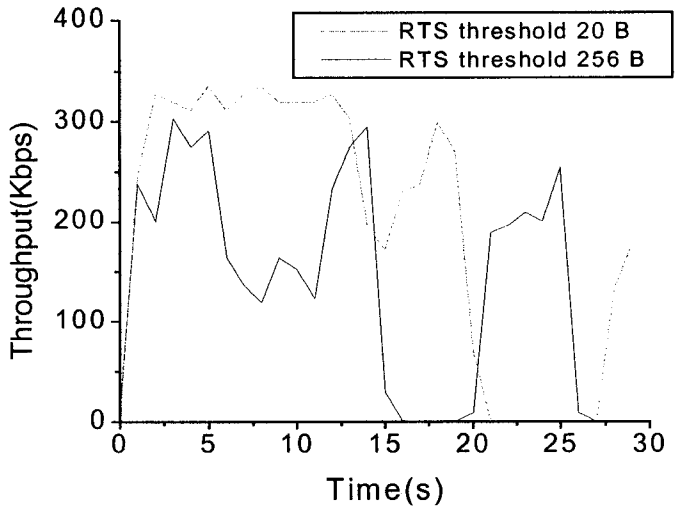
When the window goes to 4*MTU, the hidden node problem arise. Node 0 and node 2 are hidden node to each other as shown in Figure 6, so it is possible that node 0 and node 2 may send frames simultaneously and collide at node 1. For small RTS threshold (20 bytes), the virtual carrier sense mechanism almost eliminates this problem, because all the nodes are either in the source's transmission range or in the destination's transmission range. Thus the throughput is rather smooth for smaller RTS threshold as shown in Figure 8(b). But for larger RTS threshold (256 bytes), there is no RTS/CTS exchange before the SACK transmission, thus data chunks may be lost after failing the competition with reverse SACK chunks. Thus the explanation for the throughput degradation for Figure 7(b) can also be applied for Figure 8(b) with RTS threshold of 256 bytes. But when the network load increases (window goes up to 8*MTU), the hidden node problem becomes more serious, and more link failure happens at the MAC layer, thus SCTP throughput decreases from time to time as shown in Figure 8(c).



(a) window = 2*MTU



(b) window = 4*MTU

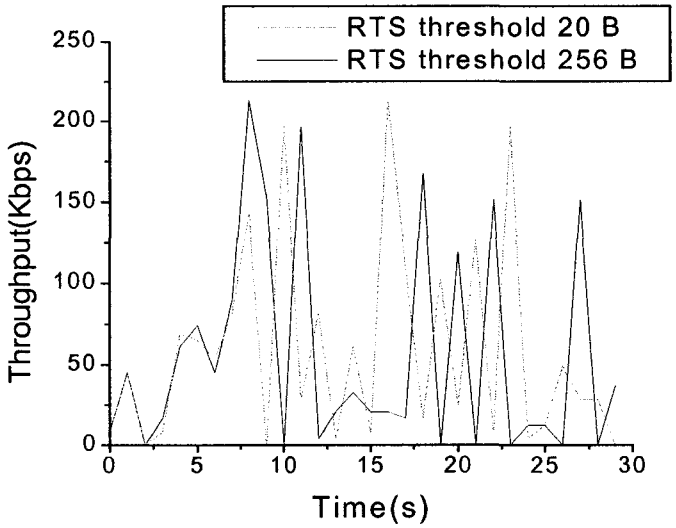


(c) window = 8*MTU

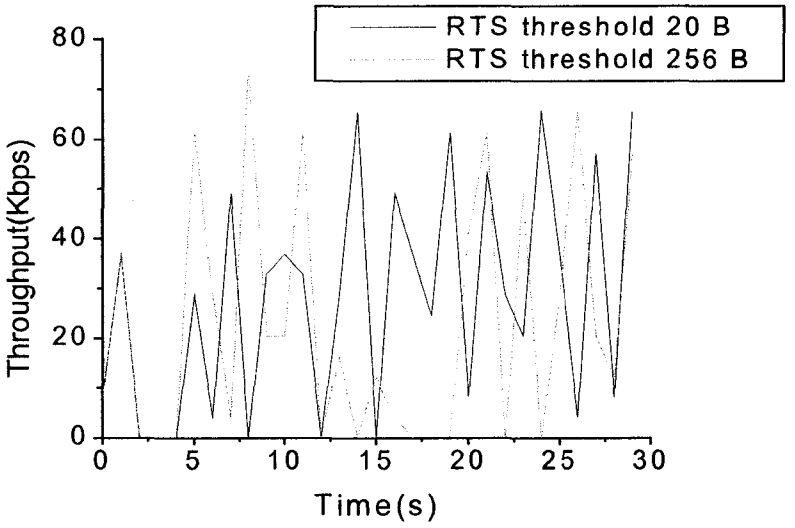
Figure 8. Throughput of two-hop SCTP association

When the number of hops goes to three, the exposed node problem also arises in addition to the hidden node problem. Return to Figure 6, when node 2 is sending a frame to node 3, if node 0 has data to send at this time, it will determine that the medium is free,

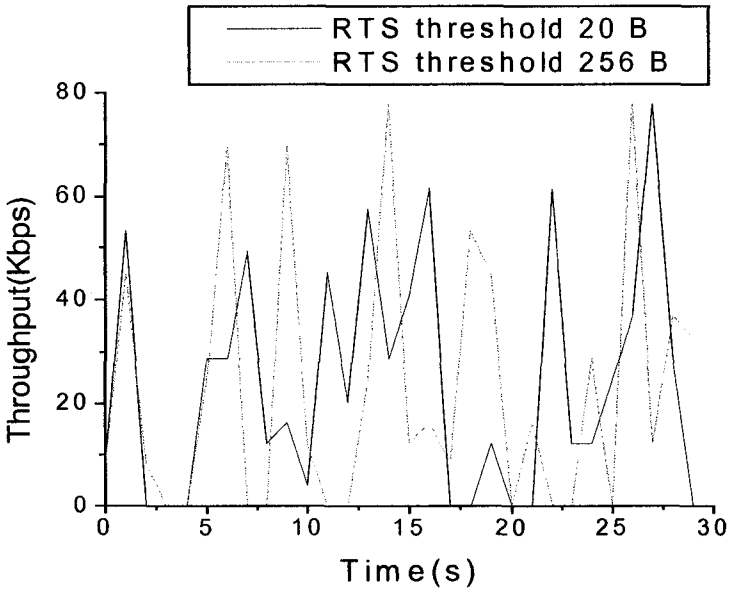
and send out the RTS frame. In this situation, node 1 cannot reply with the CTS frame (node 0 will enter into link failure if it reaches its retry limit), because it can sense the transmission of node 2. This is the so-called exposed node problem. In Figure 9 (a), the SCTP association performance degradation is mostly caused by the exposed node problem. When the window size becomes larger, the hidden node problem and exposed node problem are more serious, leading to poor throughput. And moreover, different RTS threshold values have little effect on the overall throughput, which is shown in Figure 9 (b) and 9(c).



(a) window = 2*MTU



(b) window = 4*MTU



(c) window = 8*MTU

Figure 9. Throughput of three-hop SCTP association

3.3 Unfairness Problem

In this section, we list two scenarios of unfairness problem of SCTP. We use the same topology as in Figure 6 and keep all the parameters except that we have two SCTP associations in the network. The first scenario is shown in Figure 10.

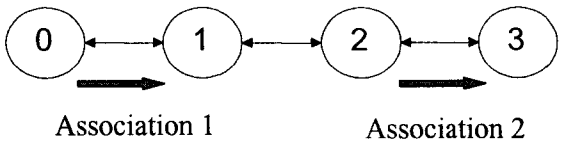


Figure 10. Unfairness problem – scenario one

SCTP association 1 starts at 0.0s transferring file from node 0 to node 1. SCTP association 2 starts at 5.0s transferring file from node 2 to node 3. And we assume both node 0 and node 2 have large file to transfer. We set the window to 4 for these two associations. The simulation result is reported in Figure 11. When association 2 develops at 5.0s, the throughput of the first association goes down to zero and the association 2 captures the channel during the remaining simulation time. We run the simulation many times with different seeds, we always get the same result. We find the reason is rather simple for this scenario after careful analysis. When association 2 develops at 5.0s, chances exist that both node 0 and node 2 sense the medium free and send RTS to reserve the channel. Certainly, these two frames will collide at node 1, thus node 0 cannot get CTS reply from node 1. But node 2 can still get the CTS reply from node 3 because node 3 cannot hear the collision at node 1 and it determines the medium is free. Now, we can see that node 2 will always win the competition with node 0. After association 2 develops, as we pointed out earlier, node 1 becomes the exposed node of node 2, even it receives RTS frame from node 0 properly, it cannot reply with CTS because it can sense

the activity of node 2 (we assume large file transfer). Thus, association 1 almost gets no chance to develop if the association 2 between node 2 and node 3 is active.

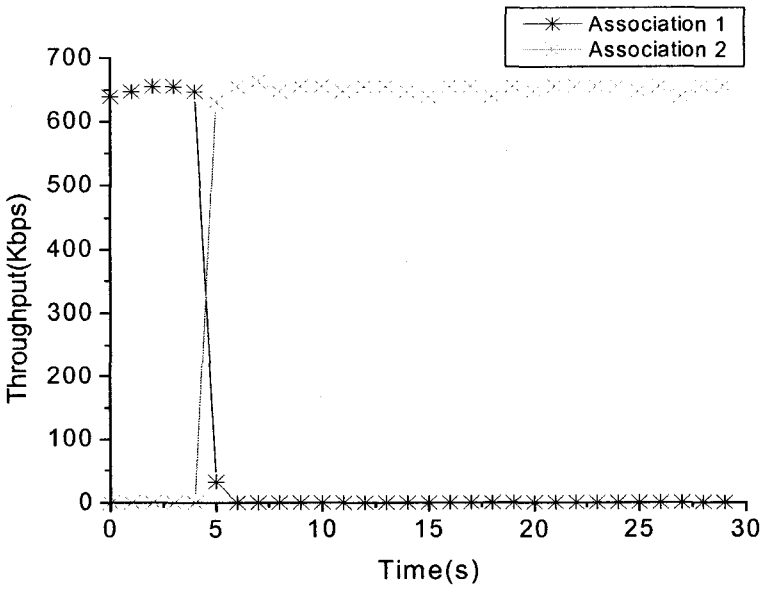


Figure 11. Throughput of two SCTP associations – scenario one

Next, we will illustrate another case of unfairness. As shown in Figure 12, we still have two associations in the network, but in this scenario, we place the two source nodes back to back. SCTP association 1 starts at 0.0s transferring file from node 1 to node 0. SCTP association 2 starts at 5.0s transferring file from node 2 to node 3. Again, we assume larger file transfer and set the window to 4 for these two associations. The simulation result is reported in Figure 13. From Figure 13, we find that association 2 never has the chance to develop. After examining debugging information carefully, we find the reason is that node 2 cannot find a route to node 3. As we know, since there is large file transfer between node 1 and node 0, node 2 gets few chances to see that the medium is free. Even if it finds the channel is free and sends out route discovery packet (a broadcast frame, no RTS frame is sent before broadcast frames) the route reply from

node 3 still gets few chances to arrive at node 2 without collision. Because in this case, node 1 is the hidden node of node 3 and it has data to send almost all the time.

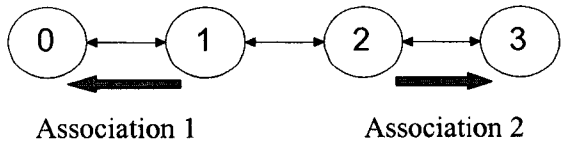


Figure 12. Unfairness problem – scenario two

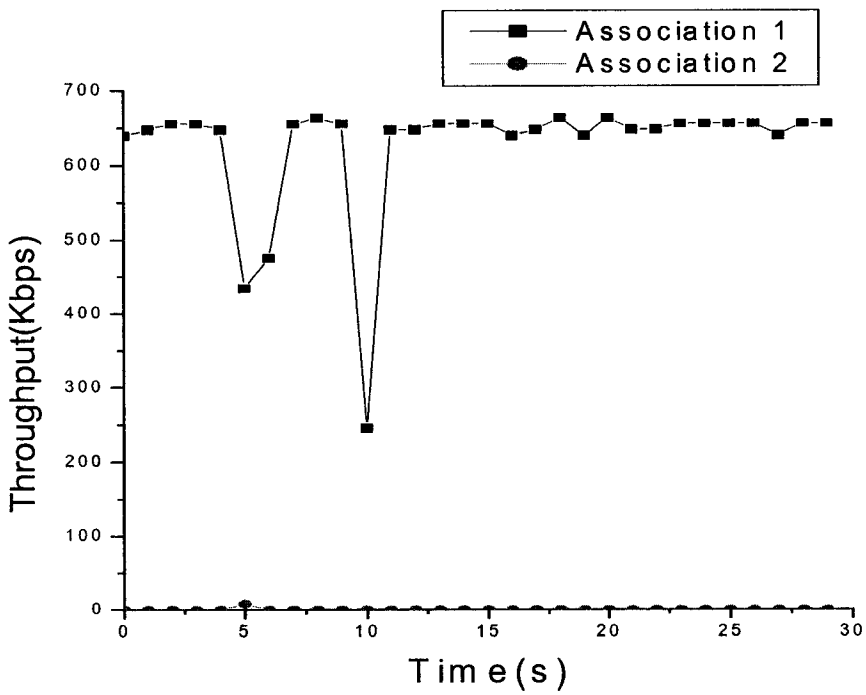


Figure 13. Throughput of two SCTP associations – scenario two

Unfairness problem is not result from any specific flaws of SCTP, but essentially due to the channel capture property of IEEE 802.11 protocol. We believe that the best and efficient way to resolve the SCTP's unfairness problem is to use an alternative MAC layer protocol that has no such a capture property.

3.4 SMALL WINDOW SYNDROME

We have mentioned in section 3.2, that SCTP association cannot retransmit lost packets before timeout when the window size is small. A simple example is given in Figure 14 to illustrate this. Suppose the receiver side window size is 4*MTU (the *cwnd* is 4*MTU too), and the data chunk size is MTU. At certain time, the sender transmits data chunks with Transmission Sequence Number (*TSN*) of 21, 22, 23 and 24. If *TSN* 21 is lost for some reason, then *TSN* 22, 23 and 24 each will trigger a SACK at the receiver. After the sender receives the third SACK, the sender's view of receiver side window size is zero. In fact, there is one data chunk space available at receiver's buffer. The sender cannot send data any more because of zero window size (sender's view). Since no fast retransmission can be triggered when the receiver side window is less than 5*MTU, thus SCTP sender can only resort to timeout to retransmit the lost packet. Timeout is very "expensive", which is visualized in Figure 7(b), 8(b) and 9(b) where throughput goes down to about zero from time to time. As is shown in Figure 14, there is no more traffic of this association in the network during the "idle period". The network is underutilized during this period. We call this the "small window syndrome". The small window problem reported here is different from the one reported in [46], which is caused by small *cwnd*. But the problem we identified here is because of small receiver side window size.

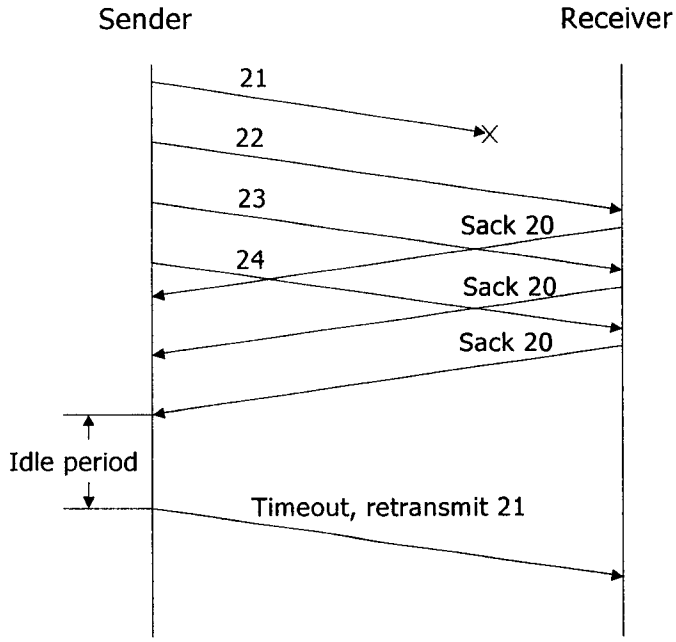


Figure 14. Example of views of receiver's window size

As we have demonstrated in section 3.2, the hidden node problem and exposed node problem can cause packet losses even with small window size in wireless multi-hop environment. It is the major reason of performance degradation for scenarios in section 3.2. Since this kind of non-congestion loss is not rare any more, so it is not wise to use costly retransmission timeout to recover lost packets. Based on this discussion, we propose a new algorithm to alleviate this problem to recover lost packets caused by MAC layer problems. For generality, we assume:

1. Link level reliability (e.g. M-ACK) is provided, and *TSN*s arrive at the destination sequentially. This assumption holds when the routing protocol supports only one path between the source and the destination, and link level reliability is provided. Results in [47] show that link level protection improves the TCP throughput performance in

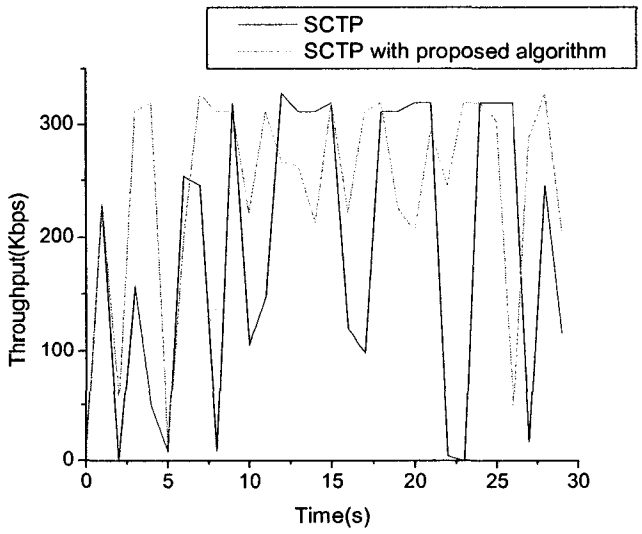
wireless multi-hop networks. It has been widely accepted that link level protection should be supported by the MAC layer protocol for ad hoc networks.

2. Data losses are mostly caused by the hidden node or exposed node problems (MAC layer), and the network is lightly loaded (small window). In other words, network congestion is not the reason for packet losses.

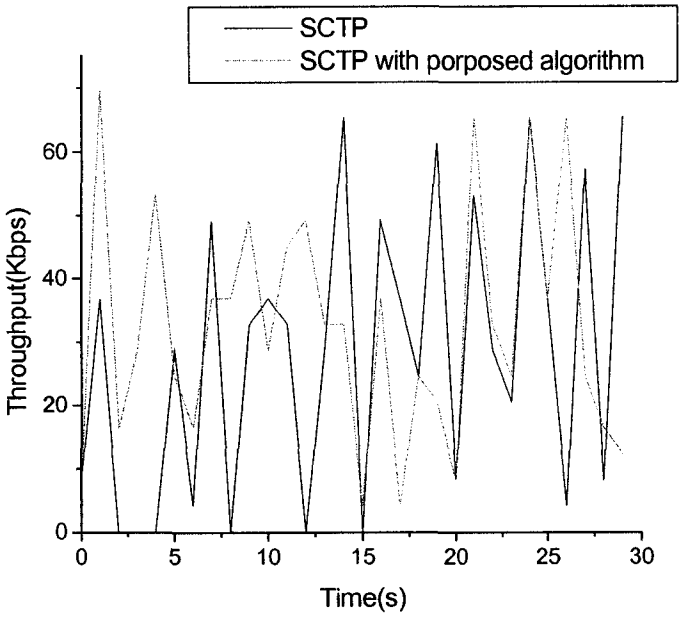
Base on these two assumptions, when the source node receives a SACK and the gap block acknowledges the outstanding data chunk with highest *TSN*, then we can conclude that the outstanding data chunk with lowest *TSN* must be lost somewhere in the network. If the packet loss is because of MAC layer problems instead of network congestion, we may want to retransmit the lost packet before timeout. Because the network may already recover before the retransmission timer runs out at the SCTP layer. In order to recover sooner from this kind of network error, we propose to retransmit the lost *TSN* (lowest *TSN*) during the "idle period" before timeout. Actually, we can determine that a *TSN* is lost whenever a higher *TSN* is gap acknowledged based on our assumption. Thus, it is possible to retransmit the lost *TSN* when a SACK with a gap block is received. But this retransmitted *TSN* may still encounter hidden node problem or exposed node problem at that time. So, we suggest retransmit a lost *TSN* during the "idle period", because MAC layer problems causing packet losses can be avoided during this period. Figure 15 shows that SCTP performs better with this algorithm than without it. The same topology as in Figure 6 is used. We keep all the parameters as they are in section 3.2 except the window size is fixed at 4*MTU. The first run is from node 0 to node 2. The second run is from node 0 to node 3. In Figure 15 (a), the total throughput in 30s is improved by 30%. In Figure 15 (b), the total throughput increases by 15% in 30s. The improved throughput

also proves the correctness of our analysis in section 3.2 for reasons of throughput degradation of SCTP associations.

This algorithm overcomes data chunk losses caused by link failure (rooted in MAC layer). It makes use of the "idle period", thus it has a minor effect upon the normal SCTP transactions. It helps to recover sooner from packet loss caused by link failure, thus improving the throughput performance. One disadvantage of this algorithm is that it cannot overcome SACK chunk losses. If the last SACK chunk is lost (the third SACK chunk in our example), SCTP association still cannot retransmit the lost packet until timeout. Another concern is that this algorithm may make network congestion become worse if the data chunk losses are actually caused by network congestion. However, the network should not be severely congested in this case, because data chunks with larger *TSN*s arrive at the receiver (or no SACK will arrive) and window size is small, and because the lost data chunk is retransmitted only once using this algorithm. Thus, the algorithm should have only minor effect on the network congestion level.



(a) Two hops between source and destination



(b) Three hops between source and destination

Figure 15. SCTP throughput with and without proposed algorithm

Chapter Four

4 ECN-capable SCTP

As discussed in Chapter 3, unfairness problem of SCTP in wireless ad hoc networks is not caused by any specific design flaws in SCTP; however, the "instability" problem is partially due to the inability of SCTP in differentiating non-congestion losses from congestion losses. Like TCP, SCTP uses packet losses as congestion indications. Thus, SCTP is unable to distinguish between congestion losses and non-congestion losses. This makes SCTP vulnerable in lossy wireless environment, such as ad hoc networks, because pseudo-congestion signals (because of non-congestion losses) are pervasive in such environment. From the perspective of transport protocols, ad hoc network or other wireless mobile environment imply more congestion unrelated packet losses than in wired network. This motivates us to consider the common feature of such networks — lossy links. The ultimate goal is to make SCTP wireless-aware in ad hoc networks (discussed in Chapter 5). In this chapter, we discuss how to introduce Explicit Congestion Notification (ECN) into SCTP and how to optimize the performance of ECN-capable SCTP over lossy links. Explicit Congestion Notification (ECN) [48, 49] is another kind of congestion signal. It indicates congestion by marking packets (or other congestion signals like the source quench message) at intermediate nodes, instead of dropping packets. By using ECN as a congestion signal, we find it is possible to differentiate non-congestion losses (e.g. wireless losses) from congestion losses and develop appropriate error recovery mechanism based on the nature of wireless losses [8].

In section 4.1, we discuss a little bit more on the motivation of using ECN. In section 4.2, we discuss mechanisms used to forward ECN. Section 4.3 introduces the ECN mechanism for TCP, which is the basis of the ECN mechanism for SCTP. Section 4.4 discusses additional functioning requirements for SCTP to be ECN-capable. Section 4.5 analyzes the optimal value of the congestion window for an SCTP source in response to ECN messages. In section 4.5, simulation results and analysis are presented.

4.1 Motivation of Using ECN

It's well known that TCP relies on packet losses to indicate network congestion. It uses either fast retransmit or timeout mechanism to recover lost packets. SCTP follows the same design principle as TCP in its congestion control algorithms. It should be noted that timeout mechanism is rather "expensive" in recovering lost packets, because the data source needs to wait for certain amount of acknowledgment to come in before it sends out new data. Thus, it is desirable that the data source can react early before the network is congested so severe that packet losses occur. Explicit Congestion Notification (ECN) is designed to achieve this goal in combination with active queue management (AQM). By using the ECN, packet losses can be reduced if source hosts can respond to the ECN message appropriately. In [48], it has been reported that the ECN mechanism can avoid unnecessary packet delays for low-bandwidth and delay-sensitive TCP connections. Another advantage of the ECN mechanism is that the source host can detect congestion rapidly regardless of coarse granularity of the timer. Here, we focus on the incorporation of Explicit Congestion Notification into SCTP in ad hoc environment. Ad hoc network implies more congestion unrelated packet losses than in wired network from the transport protocol's point of view.

4.2 Reverse or Forward Feedback

ECN provides a way for an SCTP source to react to potential congestion early. Since the source itself has no way to know the potential congestion, we need find a way to inform the source about the potential congestion. Basically, there are two alternatives to transmit ECN messages:

Reverse feedback – ECN messages are generated and sent by an intermediate node to the source when the intermediate node detects potential congestion.

Forward feedback – An intermediate node forwards the congestion signal to the receiver. Then the receiver notifies the sender of congestion by ECN messages.

Choke packet [50] and source quench message [19] are specific forms of the first alternative. Disregarding the extra traffic introduced by this mechanism, it is also hard for the source to decide which connection to choke or quench if the source has multiple connections ongoing with the destinations. It is certainly discouraging if innocent connections are affected by such a feedback. It is possible for the intermediate node to know exactly which connection or application should be notified, however, this requires that intermediate nodes aware the application. Moreover, piggybacking ECN messages in the reverse SCTP packets is forbidden due to security considerations in the protocol design of SCTP.

The second alternative, forward feedback, sends congestion signals in the forward direction (the direction to the receiver). The data receiver then informs the data source to reduce its traffic rate by echoing this signal back. This method is recommended in [11, 49]. The second alternative is easy to implement. We adopt the forward feedback mechanism in ECN-capable SCTP.

4.3 ECN In TCP

We refer readers to [48, 49] for details about the usage of ECN in TCP. However, to facilitate our discussion on the ECN mechanism for SCTP, we give a brief introduction to the ECN mechanism for TCP first.

In TCP, a TCP source and destination decide whether to use ECN or not during the 3-way handshake phase. For a TCP connection using ECN, a new data segment is transmitted with an ECN-capable Transport (ECT) codepoint set at its IP header. Intermediate routers, which support Active Queue Management (AQM) mechanism, such as RED, probabilistically mark (rather than drop) packets with ECT codepoint set when they detect congestion or incipient congestion. If a TCP receiver receives a packet marked as congestion experienced (with CE bit set), it echoes this congestion signal back to the sender by setting the ECN-echo flag in all of its subsequent ACKs until a new data segment with Congestion Window Reduced (CWR) flag set at its TCP header is received. These duplicate congestion signals reduce the risk of ACK losses (which result in the losses of congestion signals) caused by network congestion in the reverse path. In summary, an ECN-capable TCP receiver uses the congestion-experienced packet as a start signal to generate ECN messages; and it uses the CWR flagged segment as a stop signal for ECN messages. If a TCP sender receives an ECN-echo ACK packet (that is, an ACK packet with the ECN-echo flag set at the TCP header), it then halves the congestion window (*cwnd*) and the slow start threshold (*ssthresh*). Since it treats the network as a black box, it has no idea whether the halved congestion window is appropriate or not. If the new window is too small, bandwidth under-utilization will occur. If the new window is too large, new congestion will be developed soon. A TCP sender is also required to

react to congestion indications only once for every window of data. In other words, a TCP source treats all ECN-echo ACKs for a window of data as one congestion indication.

A TCP sender sets the CWR flag at the TCP header of the new data segment sent after the window reduction. Then, the cycle begins for the new window of data. If the new window of data experience congestions, the TCP sender will receive ECN-echo ACKs for this window of data, and reduces its congestion window. The effectiveness of the ECN mechanism of TCP depends on one assumption: congestion is the major reason for packet losses. This assumption is reasonable for wired network where non-congestion losses are very rare. But, this assumption is unpractical in lossy networks where packets may be lost because of link bit errors or link failures. If the stop signal (the CWR flagged segment) is lost because of congestion, ECN-capable TCP works well. The TCP receiver continuously generates ECN-echo ACKs when it receives out of order segments. The TCP sender halves its congestion window in response to these ECNs and fast retransmits (it does not halve *cwnd* or *ssthresh* any more since it does it already for ECNs for the current window of data) the lost segment when three duplicate ACKs are received. However, if the stop signal is lost because of non-congestion errors, this kind of window reduction may be unnecessary, and may result in network underutilization.

Above all, we believe that ECN-capable SCTP cannot simply follow the ECN mechanisms for TCP [48, 49] without considering the specific environment in which it operates.

4.4 Using ECN in SCTP

In this section, we discuss specific considerations for SCTP to be ECN-capable in lossy networks, and the rules for ECN capable SCTP sender, receiver and intermediate routers.

4.4.1 Use the CWR Chunk or Not?

Unlike byte-oriented TCP, SCTP is a message-based transport protocol. SCTP attempts to avoid as much bit fields as possible in its chunk structure because words processing is considered more efficient than bits looking up. This is the reason that the standard SCTP reserves two chunks for future ECN extension: the Congestion Window Reduced (CWR) chunk and the ECN Echo (ECNE) chunk. These two chunks correspond to the CWR flag and ECN-echo flag respectively in ECN-capable TCP. Notice that there will be a big difference between the ECN-capable TCP and the ECN-capable SCTP. The ECN-capable SCTP consumes extra network bandwidth because of using the CWR chunk and the ECNE chunk, while the ECN-capable TCP does not need extra network resources because it uses bit flags at TCP header. This means bandwidth efficiency should be considered for the ECN mechanism of SCTP. Taking account of both the bandwidth efficiency and the characteristics of lossy networks, we believe it is better not to use the CWR chunk in ECN-capable SCTP because of the following reasons.

- The CWR chunk itself consumes valuable bandwidth; and
- If an SCTP receiver is like an ECN-capable TCP receiver discussed in section 4.3, using the CWR chunk as a stop signal for ECN messages (ECNE chunks), the loss of a CWR chunk (due to non-congestion errors) will cause problems both at the sender and the receiver. On the one hand, the SCTP receiver continues

generating ECNE chunks, which consume bandwidth. On the other hand, the SCTP sender may infer these extra ECNE chunks as new congestion indications, thus reduce its congestion window or transmission rate unnecessarily. Then, poor throughput performance is resulted in if the non-congestion loss rate is high.

The drawbacks of using the CWR chunk will be further illustrated in section 5.1, where we compare the performance of our proposed SCTP ECN mechanism with the SCTP ECN mechanism following exactly the rules for TCP ECN mechanism [49]. Simulation results show that ECN mechanism with the CWR chunk is not only inefficient in bandwidth usage but also vulnerable to non-congestion losses (e.g. wireless losses).

4.4.2 Behaviors at SCTP Sender and Receiver

Like ECN-capable TCP, an ECN-capable SCTP source and destination decide whether to use ECN or not during the SCTP association setup period. In more detail, INIT and INIT ACK chunk are used to exchange their ECN capability. In addition to their existing protocol operations, an SCTP source and destination should meet the requirements in Figure 16 and 17 respectively in order to be ECN-capable.

In summary, three major modifications are made in our proposed SCTP ECN mechanism compared with the ECN mechanism of TCP [49].

1) Our proposed ECN mechanism does not use the CWR chunk. The reasons are discussed in section 4.4.1.
2) The congestion window is reduced to max(x, 2*MTU) for an SCTP source responding to ECNE chunks, while in ECN-capable TCP it is required to halve its congestion window in handling ECN messages. We will discuss more on this modification in the next section.

3) Corresponding modification is required at the SCTP receiver since we don't use the CWR chunk any more. Compared with an ECN-capable TCP receiver, the behavior at an ECN-capable SCTP receiver is much simpler. An ECN-capable SCTP receiver generates an ECNE chunk only when a congestion-experienced SCTP packet is received. It does not generate ECNE chunks when there are no congestion-experienced SCTP packets received.

In accordance with the requirements in RFC 2960, the format of the ECNE chunk is designed as in Figure 18. Chunk Flags are zeros at this stage; other values may be used to add more functionality to the ECN mechanism of SCTP in the future.

- Send SCTP packets with ECT codepoint set at the IP header
- When it receives an ECNE chunk, if the congestion window (*cwnd*) has not been reduced for the current window of data, it updates *cwnd* and slow start threshold (*ssthresh*) as follows:

  *ssthresh = max(x, 2*MTU);*

  cwnd = ssthresh;
- The sender ignores further ECNE chunks if it has already adjusted *cwnd* and *ssthresh* for the current window of data.
- When timeout occurs and the window has already been adjusted for the current window of data, the SCTP sender reduces its *cwnd* to one MTU while does no change to the value of ssthresh.

Note: x is defined as the size of congestion window for an SCTP source responding to ECN messages. Its optimal value is to be determined in the next section.

Figure 16. Requirements for an ECN-capable SCTP sender

- If an SCTP receiver receives an SCTP packet marked as congestion experienced, it sends an ECNE chunk immediately.
- It does not generate ECNE chunks if there is no congestion experienced SCTP packets received.

Figure 17. Requirements for an ECN-capable SCTP receiver

Chunk Type	Chunk Flags	Chunk Length
8 bits	8 bits	16 bits

Chunk Type: 12 (according to RFC2960), Chunk Flags: 0, Chunk Length: 4

Figure 18. ECNE Chunk format

4.4.3 Requirements for Intermediate Nodes

As we know, ECN is not an end-to-end congestion avoidance mechanism. It needs the coordination of intermediate nodes or routers. Intermediate nodes should apply appropriate Active Queue Management (AQM) mechanisms to detect incipient congestion. Random Early Detection (RED) [51] is one of these mechanisms. Further discussion on AQM is out of the scope of this work. We refer readers to [51-55] for more information about AQM mechanisms and marking policies.

4.5 The Optimal Value of the Congestion Window

Considering a lossy path for an SCTP association, the throughput performance is highly dependent on the characteristic of the bottleneck link and the packet loss rate along the path. The network can be modeled as a finite queue with a certain service rate (bandwidth of the bottleneck link) and a non-congestion drop rate at the bottleneck node (Figure 19).

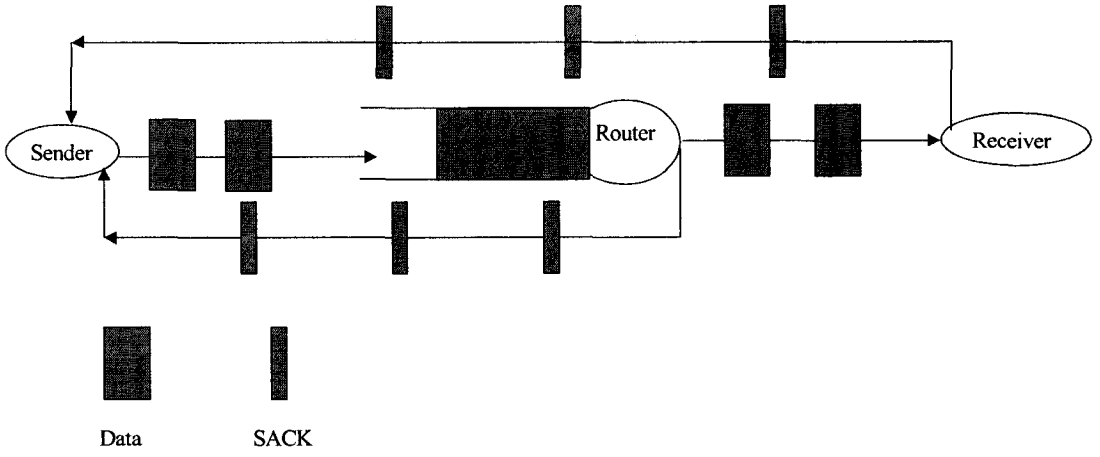


Figure 19. Lossy Network Model

According to the algorithm in Figure 16, the congestion window of an SCTP source in steady state (congestion avoidance phase) will oscillate between x and W (Figure 20). X is the lower bound of the congestion window, which is determined by window reduction policies (standard SCTP uses *cwnd/2*). W is the upper bound of the congestion window, which is mainly determined by the average queue length (if RED is used). And we find that the bottleneck link utilization or goodput performance is primarily determined by the x, because the bottleneck link is always fully utilized when the congestion window reaches W (packets are buffered in the queue at this time). Next, we will discuss the optimal value of x (i.e. the optimal congestion window) to maximize the throughput and maintain a relatively small end-to-end delay for an SCTP association using ECN.

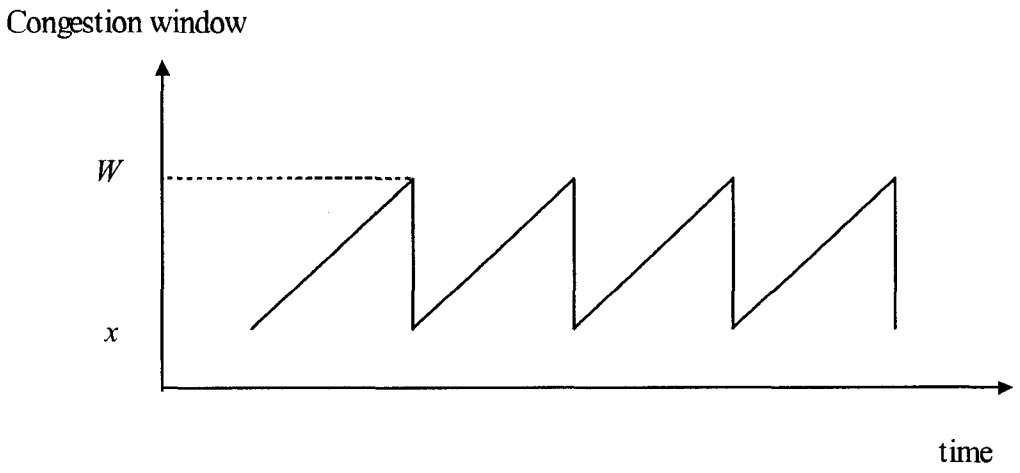


Figure 20. The oscillation of the congestion window

It is apparent that two conditions should be satisfied in order to achieve this goal. The first condition: the bottleneck link should be fully utilized, i.e., the output queue of the bottleneck link should never be empty. An empty queue indicates underutilization of the link. The second condition: x should not be too large; otherwise the end-to-end delay will be high. To meet the first condition, the smallest value of x is the product of bandwidth and delay. The second condition requires that the smallest x meets the first condition should be used. Considering the bursty characteristic of SCTP traffic, we think the best way is to leave at least one packet in the queue to avoid link underutilization. Thus, the optimal value of x is (where RTT is the round-trip time without considering the queuing delay along the path, S is the bandwidth of the bottleneck link):

$$x = RTT*S/8 + MTU \qquad (1)$$

When there are multiple associations sharing the bottleneck link, our analysis can be easily extended to fit this kind of scenarios. The optimal value of x for a specific SCTP association is about $RTT*S/8$. But, in this case, S represents the bandwidth used by a

specific SCTP association at the bottleneck link. Then, the total throughput of these associations is maximized.

Simulation results support our analysis for the optimal value of x for an SCTP source responding to ECN messages. However, it might be difficult to achieve the optimal value of x directly because it requires accurate measurement of the network round-trip-time (RTT) and information about the bottleneck link bandwidth or the bandwidth used by a specific flow at the bottleneck link. Recall that the congestion window is halved in the ECN-capable TCP in response to ECN messages. This means that x equals to *cwnd/2* in ECN-capable TCP. Certainly, this window reduction policy is simple and easy to implement. However, *cwnd/2* cannot automatically become the smallest window that satisfies the first condition we discussed previously. By choosing carefully the threshold value of queues at intermediate nodes, we find that this simple window reduction policy can be used without sacrificing the throughput performance of SCTP associations in a network with a bottleneck link. In fact, this implies that we have

$$cwnd/2 = RTT*S/8 + MTU \qquad (2)$$

Next, we discuss in details about how to achieve this for RED queue at intermediate routers. Similar methods can be developed when other queuing mechanisms are used. For simplicity, we assume the RED queue measures the queue length in packets and all packets are in size of MTU. As we discussed earlier, an SCTP association with congestion window of W (in bytes) starts to receive ECNE chunks from the receiver. When delayed SACK is disabled, W is the total number of bytes in flight, which is the sum (in bytes) of packets in the RED queue and packets in transmission. Suppose *q_len* is

the number of packets in the queue, *RTT* is the round-trip-time without queuing delays and *S* is the service rate of the queue, we have

$$W = q\_len*MTU + RTT*S/8 \quad (3)$$

According to the marking policy for a RED queue, the value of *q_len* usually falls between the min_th and max_th. Since large maximum marking probability is used in our case, approximately, we have

$$q\_len = min\_th \quad (4)$$

Recall the equation (1), we have

$$x = RTT*S/8 + MTU \approx RTT*S/8 \quad (5)$$

However, as we discussed earlier, it is hard to achieve x directly through equation (5). If we assume *W/2* is the optimal value of x, together with equation (3) and (5), we will have

$$W/2 = (q\_len*MTU + RTT*S/8)/2 \approx RTT*S/8 \quad (6)$$

and get

$$q\_len*MTU \approx RTT*S/8 \quad (7)$$

Using equation (6), we have

$$min\_th*MTU \approx RTT*S/8 \quad (8)$$

This means that if *min_th* is equal to *RTT*S/(8*MTU)*, then *W/2* is approximately equal to *RTT*S/8,* the optimal value of x.

So, by carefully choosing the minimum threshold of RED queue as *RTT*S/(8*MTU)*, *W/2* will be approximately equal to the optimal value of x. In this way, we don't need real-time accurate information of the round-trip-time and the bandwidth of the bottleneck

link, but set the minimum threshold value according to known information about the network parameters through measurement or configuration logs.

4.6 Simulation Results and Analysis

In this section, we use simulations to support the effectiveness of the ECN mechanism for SCTP discussed in section 4.4 and section 4.5. In section 4.6.1, simulation results show that ECN mechanism with the CWR chunk is not only inefficient in bandwidth usage but also vulnerable to non-congestion losses (e.g. wireless losses). In section 4.6.2, simulation results verify the analysis about the optimal value of x for an SCTP source in response to ECNE chunks. In section 4.6.3, we report a RED enhancement that overcomes a drawback of the marking/dropping policy of the RED queue. In section 4.6.4, we use simulations to verify the simplified method that realizes the optimal value of x discussed in section 4.5. In section 4.6.5, we examine the effect of network load upon the window reduction policies for SCTP in response to congestion notifications (ECN in this work).

In the simulation network (Figure 21), we can have 15 SCTP associations simultaneously. S1 to S15 are source endpoints, and D1 to D15 are destination endpoints respectively. We name the association between S1 and D1 association 1, association between S2 and D2 association 2, and so on. Every endpoint attaches to an access router using a link with 10 Mbps and 1 ms propagation delay. The link between node R4 and R6 is a bottleneck link with bandwidth S of 2 Mbps. According to the configuration, associations 1 to 5 have the same round-trip-time (RTT) of 79.6 ms; associations 6 to 10 have RTT of 120 ms; associations 11 to 15 have RTT of 161 ms. RED method is used in the routers. There are four parameters associated with RED method: queue weight w,

minimum threshold, maximum threshold and maximum marking probability. Large value of maximum marking probability is used in the RED method during our simulations in order to maintain the average queue length around the minimum threshold. The RED parameters of the access routers (R1, R2, R3, R8, R9, R10) are: minimum threshold = 5 packets, maximum threshold = 10 packets, w = 0.002, maximum marking probability = 50%. The RED parameters of other routers (R5, R6, R7) are: minimum threshold = 10 packets, maximum threshold = 20 packets, w = 0.002, maximum marking probability = 50%. At R4, the maximum threshold is 2 times the minimum threshold, and the maximum marking probability is 50%; the minimum threshold may vary in each simulation run. The path MTU is 1500 bytes. We assume each SCTP data chunk is at the same size of 1468 bytes. The SCTP sources always have data to send. These setting will be used in all the simulation runs except that changes are notified. All the simulations are made with the network simulator, OPNET Modeler [44].

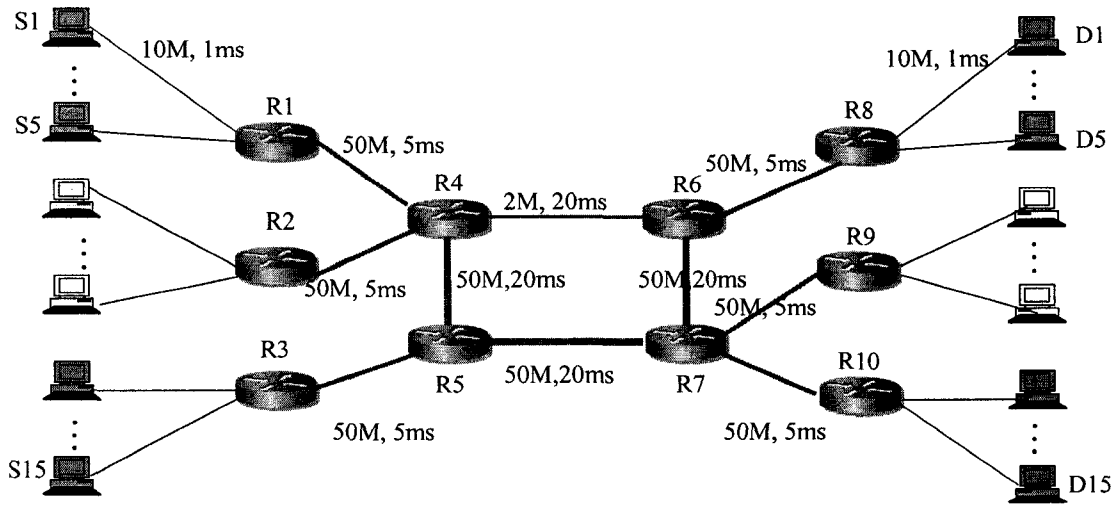


Figure 21. Simulation topology

4.6.1 The Drawback of Using the CWR Chunk

In order to demonstrate the drawback of using the CWR chunk, the performance of two ECN-capable SCTP variants is compared. One is our proposed ECN SCTP, which is described in section 4.4; the other is the ECN SCTP, which exactly follows the rules for ECN TCP in [49]. Non-congestion packet losses associated with CWR chunks are emulated at router R4. In this case, only deterministic drop pattern is used. Random drop pattern may be used if one has interest in the relationship between the throughput and the drop pattern. Since we focus on the effect of the CWR chunk losses, we only use deterministic drop pattern. In this case, there is only one active SCTP association in the network. The throughput of the SCTP association between source S1 and destination D1 using the two ECN SCTP variants is measured. To minimize the effect of slow start phase during the setup period of the SCTP association, only the throughput during the last 2 minutes of the simulation time (3 minutes) is counted. Although simulation results vary when different RED parameters at node R4 are used, the performance trend is the same. Here, we only list the simulation results (Figure 22) for one set of RED parameters (w = 0.002, minimum threshold = 14, maximum threshold = 28, maximum marking probability = 50%).

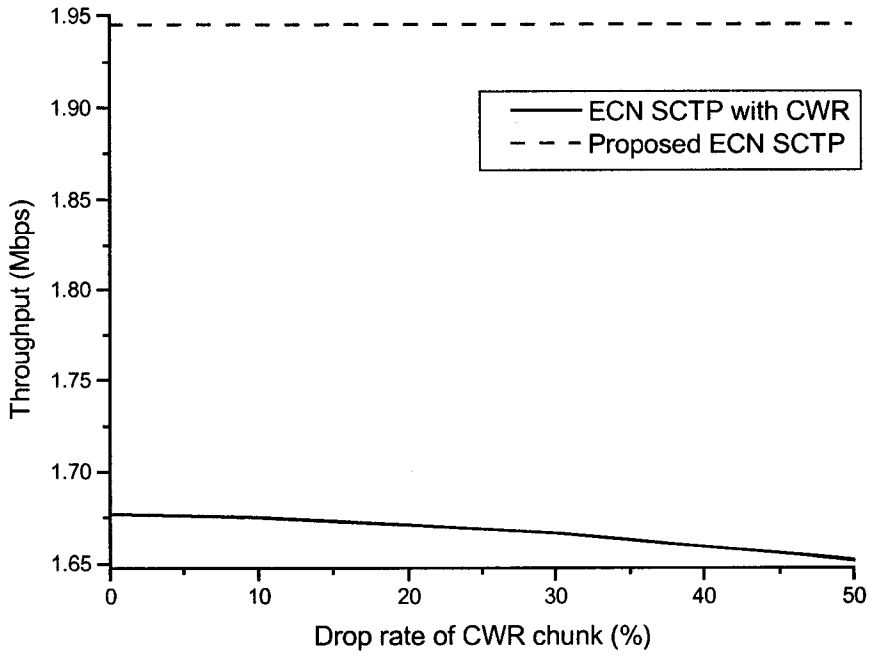


Figure 22. Throughput for two ECN SCTP variants

From Figure 22, we can see that the throughput performance of ECN SCTP with CWR chunk is poor comparing with the performance of the proposed ECN SCTP mechanism. The throughput of the ECN SCTP with the CWR chunk decreases when the drop rate of the *CWR* chunk is increased. The simulation results adhere to the analysis in section 4.4. Our proposed ECN SCTP is superior to the ECN SCTP with CWR chunk in two ways.

1) Our proposed ECN mechanism generates fewer ECNE chunks, thus reduces overhead associated with ECN mechanism of SCTP. Taking a specific simulation run as an example (0% *CWR* drop rate), our proposed ECN mechanism generates only 224 ECNE chunks in 3 minutes, while the ECN mechanism with the CWR chunk generates 1521 ECNE in 3 minutes, which is about 579% more.

2) The ECN SCTP with CWR chunk suffers performance degradation when CWR chunks are lost due to non-congestion errors, which is not an issue in our proposed ECN SCTP mechanism since it does not use the CWR chunk.

4.6.2 The Optimal Congestion Window

In this section, we use simulations to verify the correctness of the expression of the optimal value of x (equation 1) for an SCTP source in response to ECNE chunks. Simulations are conducted over the SCTP association between the node S1 and node D1 in Figure 21. Delayed SACK option is disabled, which means the SCTP receiver acknowledges every packet right after receiving it. In this case, the optimal value of x is 21400 bytes (79.6*2000/8+1500) according to equation 1. Goodput, which is defined as the number of packets successfully received by the receiver excluding duplicate packets, is used as the evaluation metric. The goodputs are collected for a set of x. To minimize the effect of slow start phase during the setup period of the SCTP association, only the goodput during the last 2 minutes of the simulation time (3 minutes) is counted. Although simulation results vary for different settings of the RED parameters at R4, the performance trend is the same. We have tested the goodput for a large number of RED parameters at R4; the trends of the results are similar. Because of the space limitation, we list only the results (Figure 23) for one set of the RED parameter (w = 0.002, minimum threshold = 10, maximum threshold = 20, maximum marking probability = 50%). Simulation results in Figure 23 agree with our analysis in section 4.5: the optimal value of the congestion window x should fully utilize the bottleneck link and it should be as small as possible. By using the optimal value of x (21400 in this case), the ECN-capable SCTP association achieves the best goodput performance.

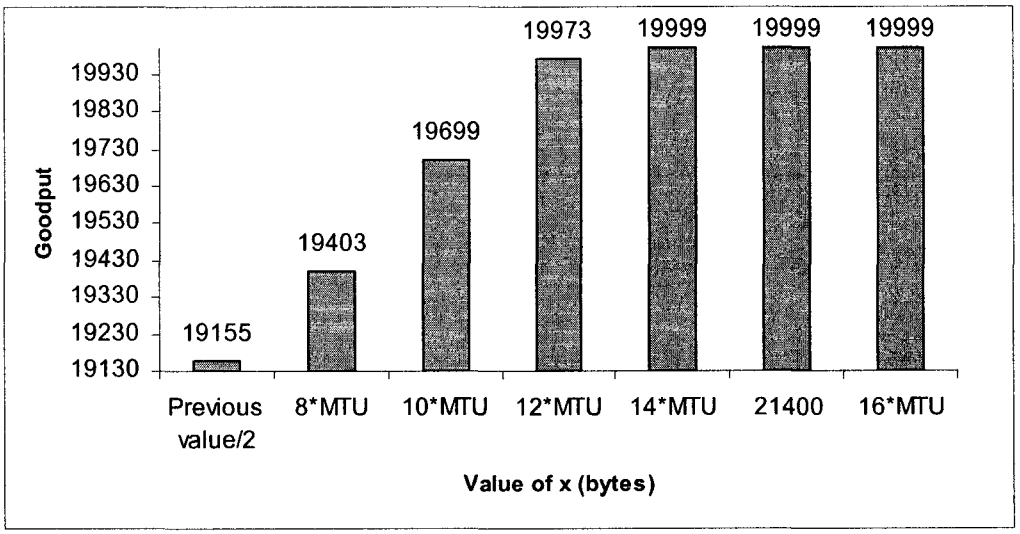


Figure 23. Goodput for the set of x

We notice in Figure 23 that the goodput performance is always the best when x is larger than the optimal value (21400 in this case). In fact, when x is larger than 21400, the bottleneck link is always fully utilized. Thus the goodput performance is determined by the bottleneck link bandwidth, and the bottleneck link bandwidth is unchanged during the simulation. Large values of the congestion window x (larger than the optimal value) are undesirable because packets will experience more waiting time in the queue.

Kwon et al. [56] has studied the less aggressive reduction of the congestion window for TCP in response to ECN messages, and they found the throughput performance of TCP is improved. In fact, when the congestion window is reduced less aggressively in response to ECN, the congestion window falls in the range of x larger than or equal to the optimal value, thus the throughput performance is improved compared with the throughput of TCP halving its congestion window in response to ECN messages. The effect of link underutilization is noticeable when small value of x is used. The goodput drops to 19403 for x of 8*MTU bytes. We can also infer from Figure 23 that half of the

previous congestion window cannot automatically become the optimal congestion window for SCTP in response to ECNE chunks.

4.6.3 RED Enhancement

Before we go further to the simplified realization of the optimal value of x for ECN-capable SCTP. We need introduce an extension to the RED method first, which is proposed to overcome a drawback of the current RED method.

RED method uses a low-pass filter to calculate the average queue length. When the average queue length falls between the minimum threshold and the maximum threshold, it marks incoming packets (assume mark/drop tail policy is used) using a probability proportional to a flow's share of the bandwidth of an output queue at the router. It marks all incoming packets when the average queue length exceeds the maximum threshold, and drops all the incoming packets when the buffer is overflowed. Equation (9) gives the expression that calculates the average queue length in RED method.

$$avgq \leftarrow avgq*(1-w) + w*q \quad (9)$$

avgq: average queue length

w: queue weight

q: actual queue size.

RED tries to accommodate bursty traffic by using small values of w (usually is at the order of 10^{-3}) in equation (9). This avoids interpreting quick fluctuation of network load into congestion, and detects only persistent congestion at the queue. However, it takes long time (more than a RTT) for the average queue length to decrease after a persistent congestion if small values of w are used in the low-pass filter. Since it is possible that the average queue length might remain at high values (between maximum and minimum

threshold) for a certain period of time even there is no congestion at all, some packets arriving at the queue during this period will be marked as "congestion experienced" with certain probability. The situation becomes worse when the maximum marking probability is large. When an SCTP receiver echoes these pseudo congestion notifications back to an SCTP source, it may mistakenly consider them as new congestion notifications. Although an SCTP source treats all ECNE chunks for a window of data as one congestion notification, it considers ECNE chunks for the next or another window of data as new congestion notifications and reduces its congestion window accordingly.

Suppose the average queue length (*avgq*) and the actual queue length (*q*) are at the same order after a persistent congestion. The item *w*q* is much less than the item of *avgq*(1-w)* for small *w*. Approximately, we have

$$avgq \leftarrow avgq*(1-w) \qquad (10)$$

Taking $w = 0.002$ as an example, if the actual queue length remains above zero, it needs about 347 packet arrivals for the average queue length to drop to the half of the previous value. Several windows of data should have been sent by the SCTP source before the average queue length is halved. Thus, it is possible that an SCTP source will unnecessarily reduce its congestion window because of those pseudo congestion notifications discussed earlier. Next, we use a snapshot during our simulation to further illustrate this problem.

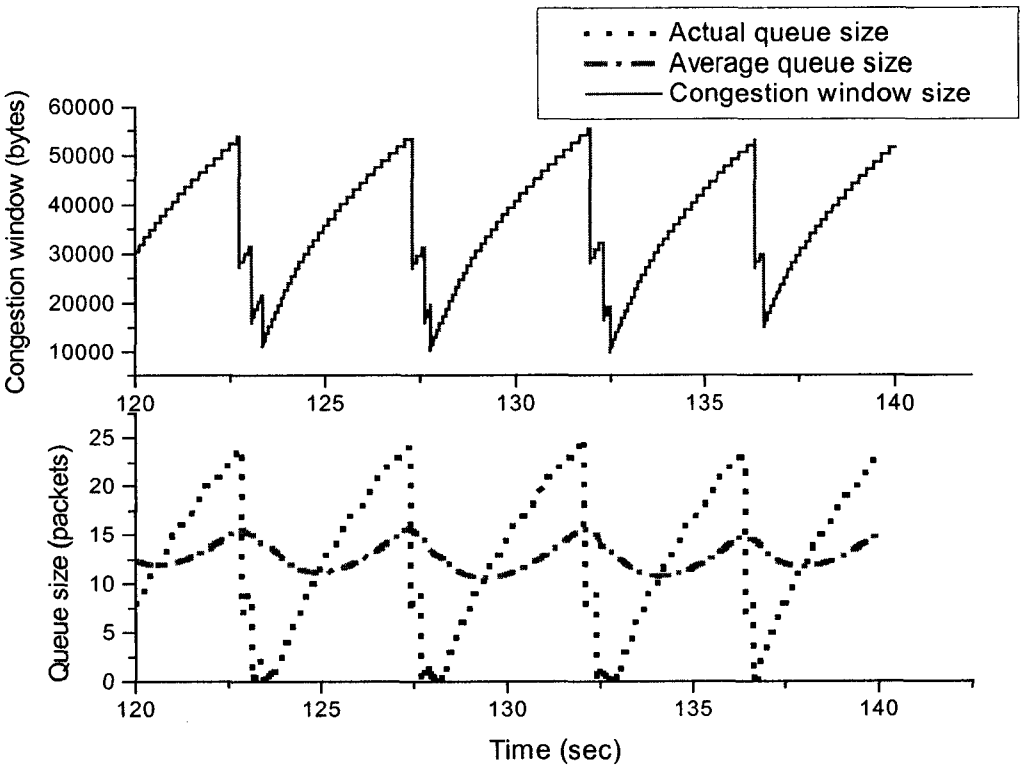


Figure 24. The effect of low-pass filter of RED queue

Figure 24 is a snapshot of the congestion window of the SCTP association between S1 and D1, the average queue length and the actual queue length at R4 from 120s to 140s. In this case, the RED parameters at R4 are: w = 0.002, minimum threshold = 10, maximum threshold = 20, maximum marking probability = 50%. Total simulation time is 3 minutes. In Figure 24, we observe that the SCTP source at node S1 usually reduces its congestion window 2-3 times for a persistent congestion at the RED queue because of the slow decrease of the average queue length by using a small value of w in the low-pass filter. It reduces its congestion window even there are few packets in the queue (small actual queue length).

We can infer from Figure 24 that the average queue length of RED queue cannot exactly reflect the actual congestion level of the network after a persistent congestion. However, we cannot use a large value of w, otherwise, the design goal of RED - to accommodate bursty traffic - is compromised. We propose to consider the actual queue length as well in determining whether there is congestion or not. The proposed marking or dropping policy is as follows (taking mark/drop tail policy as an example).

- When the actual queue length is larger than the minimum threshold and the average queue length falls between minimum threshold and maximum threshold, RED marks or drops incoming packets using a probability proportional to a flow's sharing of bandwidth of an output queue at the router.
- When the average queue length and the actual queue length exceed the maximum threshold, it marks all incoming packets.
- When the buffer is overflowed, it drops all the incoming packets.

4.6.4 Simplified Method to Achieve the Optimal Congestion Window

As we discussed in the section 4.5, if the minimum threshold (in packets) of RED queue is set to *RTT*S/(8*MTU)*, then *cwnd/2* becomes the best value of x for ECN-capable SCTP. In this section, we try to verify this method in a network with a large scope of round-trip-time. The goodput performance of the SCTP association between S1 and D1 in Figure 21 is measured. The link delay of the link between R4 and R6 is continuously changed during the test. This creates a large scope of round-trip-time for the network. Table 1 summarizes those parameters specific to this test. All other parameters are kept the same.

Link delay between R4 and R6 (ms)	*RTT* (ms)	Minimum threshold (packets)	*RTT*S/8+MTU* (bytes)
5	49.6	9	13900
10	59.6	10	16400
20	79.6	14	21400
30	99.6	17	26400
40	119.6	20	31400
50	139.6	24	36400

Table 1. Parameters used in optimal congestion window test

The RED enhancement discussed in section 4.6.3 is used in simulations. For each *RTT* in Table 1, the goodputs of the SCTP association for x of *cwnd/2* and of *RTT*S/8 + MTU* are measured. To minimize the effect of slow start phase during the setup period of the SCTP association, only the goodput during the last 2 minutes of the simulation time (3 minutes) is counted. Simulation results in Figure 25 illustrate that for x of *cwnd/2* and of *RTT*S/8 + MTU,* the goodputs of the SCTP association are very close. This strongly supports the effectiveness of the proposed simplified and practical method in achieving the optimal value of x: by carefully choosing the threshold of a queue that supports ECN, half of the previous congestion window becomes the optimal value of x for an SCTP source in response to ECNE chunks. But, the goodput for x of *cwnd/2* is a little less than the goodput for x of *RTT*S/8 + MTU* sometimes. This is because the RED enhancement still cannot totally eliminate multiple window reduction at the SCTP source during persistent network congestion, especially when the congestion window is small. This will hurt the goodput performance when x = *cwnd/2* is used. But when x = *RTT*S/8 + MTU* is used, the goodput performance will not be affected even the SCTP source is forced to

"reduce" its congestion window multiple times during persistent congestion because each time it picks the same value of x.

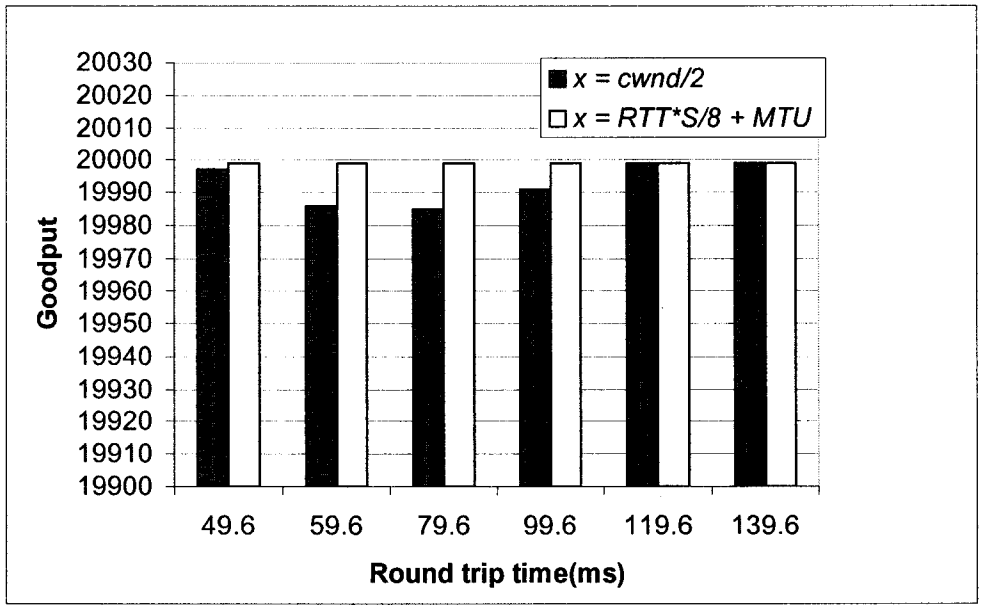


Figure 25. Goodput for x of different expressions

One drawback of the simplified method in achieving the optimal congestion window is the possible long delay for individual packet at the bottleneck node when the bandwidth-delay product is large. However, this drawback should not be a big concern for those delay insensitive best-effort services.

4.6.5 Do the Window Reduction Policies Matter?

In this section, we demonstrate the effect of network load upon window reduction policies for ECN-capable SCTP in response to congestion indications (ECNs in this work). First, we examine the effect of window reduction policies upon the goodput of single SCTP association. In this case, SCTP association 1 (between S1 and D1) is the only active association in the network. The enhanced RED method described in section 4.6.3 is used at the routers. RED parameters at R4 are: minimum threshold = 14 packets,

maximum threshold = 28 packets, w = 0.002, maximum marking probability = 50%. The minimum threshold (14 packets), which is approximately *RTT*S/(8*MTU)*, makes *cwnd/2* approximately the optimal value of the congestion window for the ECN-capable SCTP source in response to ECN messages. Other more aggressive window reduction policies like *cwnd/4*, *cwnd/6* and *cwnd/8* are also used for comparison purpose. To minimize the effect of slow start phase during the setup period of the SCTP association, only the goodput during the last 2 minutes of the simulation time (3 minutes) is counted. As illustrated in Figure 26, the SCTP association achieves goodput of 19984, which corresponds to link utilization (the bottleneck link) of 99.92%, under the window reduction policy of *cwnd/2*. And the goodput decreases when more aggressive window reduction policies than *cwnd/2* are used. The goodput is only 19615 (corresponds to link utilization of 98.1%) under the window reduction policy of *cwnd/8*. This means that an appropriate window reduction policy is necessary in order to achieve high goodput or high link utilization when there is only one SCTP association in the network with a bottleneck link. An aggressive window reduction policy, which backs the congestion window too much, will result in underutilization of the network. When the network load is light (e.g. 2-3 SCTP associations share the bottleneck link), same results are observed: the total goodput of SCTP associations using the optimized congestion window (which is *cwnd/2* when the minimum threshold is properly chosen) in response to ECN messages is higher than the total goodput of SCTP associations using other sub-optimal window reduction policies.

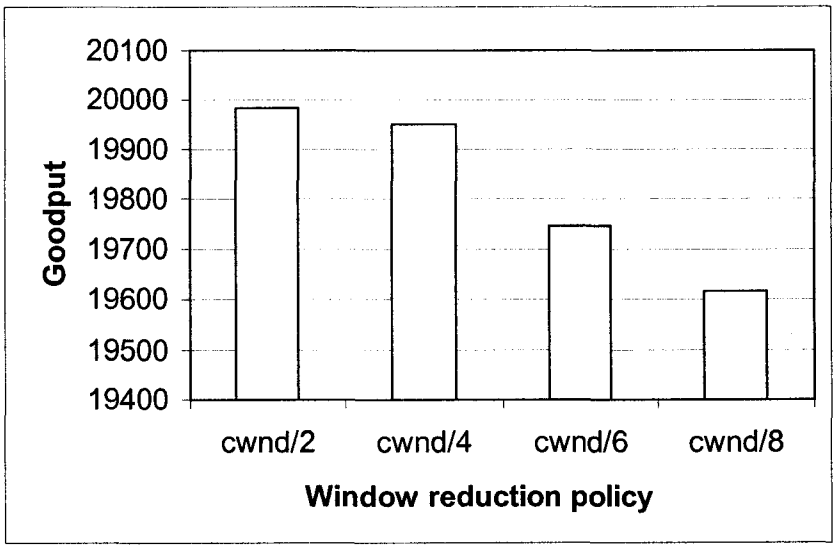


Figure 26. Goodput vs. window reduction policy

However, we find that the total goodput of SCTP associations is not sensitive to the window reduction policies when the network load is heavy enough. Taking the window reduction policy of *cwnd/8* as an example (Figure 27), when the number of SCTP associations is above 4, the total goodput for the SCTP associations in the last two minutes of simulation is about 20000, which is the maximum number of packets (in size of 1500 bytes) that can be transmitted by a bottleneck link with 2 Mbps in 2 minutes. Similar results are found when other aggressive window reduction policies like *cwnd/4* and *cwnd/6* are used.

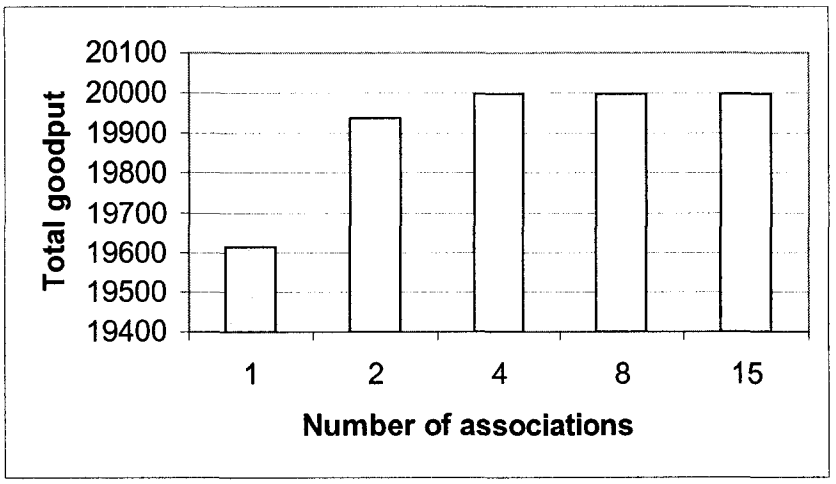


Figure 27. Goodput vs. network load when the window reduction policy is *cwnd/8*

To simplify the analysis of the above phenomenon, we assume all SCTP associations sharing the bottleneck link are homogeneous associations, i.e. they have same minimum round-trip-time RTT_{min}. And we suppose n is the number of SCTP associations and each packet has size of MTU. In fact, each SCTP source has at least 2 packets outstanding (initial window is 2*MTU) when the association starts. When n is above certain value, the throughput, $n*2*MTU*8/RTT_{min}$ (the average round-trip-time can be used in this expression for heterogeneous SCTP associations), will be larger than the bandwidth S of the bottleneck link. Then, Queue will build up at the bottleneck node and the round-trip-time will increase and reach a larger value RTT_{steady}. This RTT_{steady} should be at least so large that $n*2*MTU*8/RTT_{steady}$ equals to S (Little's law). We should note that $n*2*MTU*8/RTT_{min} > S$ is always true when the network load is heavy. This guarantees that the queue at the bottleneck router will never be empty. Thus, the bottleneck link will be fully used all the time, and the window reduction policy has no effect upon the goodput performance. This explains the phenomenon – total goodput performance is not

sensitive to window reduction policies when the network load is heavy ($n*2*MTU*8/RTT_{min} > S$).

In summary, an optimized or a fine-tuned window reduction policy for SCTP source responding to congestion indications helps to achieve higher link utilization than other sub-optimal methods when the network load is light; however, there is no benefit when the network load is heavy. This implies that complicated methods should not be advocated to fine-tune SCTP or TCP's congestion window in response to congestion indications (e.g. ECNs or three duplicate ACKs) because the increase in performance (when network load is light) may not be worth the increase in complexity of the protocol.

Chapter Five

5 Wireless-aware SCTP

As we have mentioned, the standard SCTP, RFC 2960, assumes that all packet losses are congestion losses. Thus, it does not perform well in ad hoc wireless networks. Based on the ECN-capable SCTP discussed in Chapter 4, we designs in this chapter an alternative of SCTP, which is Wireless-aware (W-A). The W-A SCTP is the first that simultaneously considers (i) the nature of wireless losses (random or bursty losses), (ii) the energy efficiency of battery powered ad hoc nodes, and (iii) the goodput performance.

5.1 Introduction

SCTP bases its congestion control on TCP congestion control principles [35] and uses the Selective Acknowledgment (*SACK*) extension of TCP [26]. Moreover, SCTP makes the same assumption as TCP does — network congestion is the primary reason of packet losses. Thus, SCTP cannot differentiate two totally different kinds of packet losses, packet losses due to network congestion and packet losses due to wireless errors (we refer this kind of losses as wireless losses thereafter), and it treats all wireless losses as congestion losses. SCTP relies on fast retransmit and timeout mechanism to recover these two kinds of packet losses. There are three drawbacks using congestion control mechanism to recovery wireless losses: i) the retransmission policy is not optimized to recover wireless losses since it is targeted to congestion losses; ii) unnecessary window reduction results in poor throughput performance when the wireless loss rate is high; iii) energy-efficiency is not considered while energy conservation is an important issue for

battery-powered ad hoc nodes. Since wireless losses may be pervasive in MANET environment due to the usage of radio channel, the lacking of efficient Loss-Recovery mechanism for wireless losses in SCTP becomes the bottleneck of performance improvement for SCTP in such environment. This analysis is also applicable to TCP because TCP also treats wireless losses as congestion losses.

Broadly, there are two categories of wireless losses: i) random loss, packet losses are independent, and ii) bursty losses, packet losses are correlated. Many efforts have been made to improve TCP performance in wireless environment [2, 3, 4, 6, 7, 57]. However, finding appropriate error-recovery strategy based on the nature of wireless losses (random or loss) is till an open research problem [8]. This chapter is to provide a solution to this problem. Although mechanisms are designed in the context of SCTP, they should also be applicable to TCP in MANET environment because both SCTP and TCP follow the Additive Increase and Multiplicative Decrease (AIMD) principle [18].

5.2 Related Work

During the last decade, there have been many proposals for improving TCP's performance in wireless network. Basically, two strategies are used, (1) hiding the feature of wireless links from TCP, thus avoiding any modification to the TCP [2-4], and (2) adjusting TCP to fit the wireless network requirement [6, 7]. Because TCP has been widely used as a transport protocol in Internet, we must take those legacy applications into consideration when we are trying to improve the performance of TCP. So, it won't be difficult to understand that the first strategy is preferred over the second strategy, and methods applying the second strategy are usually restricted by the requirement to be backward compatible with standard TCP. There are no legacy applications of SCTP, so

we are free to use either strategy. The first strategy assumes that an intermediate node (e.g. base station) can do local retransmission or other remedies to hide the wireless link from TCP sources, thus TCP sources operate the same as in wired networks. However, this assumption is not valid in MANET environments due to its infrastructureless feature. In ad hoc network, each node can act as a host and a router to relay traffic for other nodes. There is no centralized node, like base station. In ad hoc network, all nodes are mobile. The second strategy sounds good, however, the existing work applying the second strategy falls short of error-recovery procedures designed specifically for wireless losses.

Work in [5, 7] is the latest effort to improve TCP's performance in mobile wireless networks. Model analysis on TCP over wireless links has been done in literature. TCP performance analysis over random loss wireless link, in which the random loss is modeled as i.i.d packet level loss, can be found in [59, 60]. TCP performance analysis over bursty loss wireless link, which is modeled as a two-state Markov chain [61], can be found in [62, 63, 64]. Both i.i.d loss model and two-state Markov chain are used in this work to emulate the wireless losses in the MANET environment.

Our work is philosophically close to the work of ad hoc TCP in [57] on the aspect of differentiating wireless losses from congestion losses (Explicit Congestion Notification is used). But we focus on the design of the loss recovery mechanism that emphasizes handling wireless losses appropriately according to their nature (random or bursty loss).

5.3 Overcome Wireless Losses

In this section, we discuss three important issues in designing efficient loss recovery mechanism for SCTP in MANET. Firstly, a mechanism should be developed to

differentiate wireless losses from congestion losses, which is to be discussed in section 5.3.1. The second issue is about the retransmission timer, i.e. how to set the retransmission timer at times of retransmission and consecutive timeouts when the SCTP operates in the Loss-Recovery mode, which is to be discussed in the section 5.3.2. The third issue is on the retransmission policies for random packet losses and bursty packet losses, which is to be discussed respectively in section 5.3.3 and 5.3.4.

5.3.1 Identify Wireless Losses

Since the assumption that network congestion is the only reason of packet losses is not valid any more for SCTP in lossy wireless networks, we try to find an alternative of network congestion signal (we cannot rely intermediate nodes like base stations to hide the wireless link from SCTP source). Explicit Congestion Notification (ECN) [48, 49] is adopted in this work to differentiate wireless losses from congestion losses. We choose ECN is because on one hand, ECN mechanism has been extensively studied in TCP, thus many results may be reused to make SCTP ECN-capable, and on the other hand, ECN is a proposed standard in the IETF now. However, the ECN mechanism for TCP cannot be adopted directly into SCTP because of the concerns of bandwidth efficiency and wireless errors in MANET environment. This work is based on the ECN-capable SCTP discussed in Chapter 4, details on ECN mechanism for SCTP in MANET environment can also be found in [65]. In the next paragraph, we discuss how to differentiate wireless losses from congestion losses by using the ECN mechanism.

Considering that congestion is highly correlated with "build up" and "die down" phases, a lost packet with no "mark" (ECN echo chunk in SCTP) in adjacent packets represents a strong indication that this packet was lost because of wireless errors.

According to the time correlation between wireless losses and congestion losses (indicated by ECN messages in this work), there are two scenarios in which wireless losses occur.

- Wireless losses occur when there are no ECN messages received for the current window of data at the SCTP source endpoint.
- Wireless losses occur when there are ECN messages received for the current window of data at the SCTP source endpoint.

In the first scenario, the current ECN-capable SCTP [65] or SCTP [9] source will reduce their transmission rate when they detect packet losses (wireless losses). But, in fact, the network is not congested (currently) since no ECN messages are received (the current network load is light). It is inappropriate for the SCTP source to trigger congestion control mechanism and reduce its transmission rate in the presence of no network congestion. Ideally, the SCTP or ECN-capable SCTP source must have the ability to correctly identify non-congestion losses in this scenario in order to perform well. Failing to identify non-congestion losses results in unnecessary reduction of the congestion window, i.e. the transmission rate, thus results in bad performance. By using ECN as the congestion notification, we can claim that packet losses without ECN received for the current window of data are wireless losses.

In the second scenario, network congestion coexists with non-congestion errors. Since it is possible that packets may be lost due to network congestion in this scenario, we cannot claim that all packet losses as wireless losses in this scenario. Then how to handle lost packets in this scenario? We propose first to apply the rule of **reacting to congestion only once for a window of data** [49], which eliminates the necessity of

identifying the nature of packet losses (wireless loss or congestion loss) for an SCTP source. Then, the SCTP source can act as if all packets were lost due to wireless errors, and execute appropriate retransmission procedure without applying congestion control. To those packet losses due to wireless errors, executing appropriate retransmission procedure without applying the congestion control mechanism is the right choice. To those packet losses due to the network congestion, treating them as if they were wireless error is also a good choice, because the congestion control mechanism is just applied (because of the reception of the ECN messages for the current window of data) and will be applied again if the current window (or traffic load) is still too much for the network. The rule, reacting to congestion only once for a window of data, is originally designed to help TCP overcome multiple packet losses in a window of data. By applying this rule in Wireless-aware SCTP, we can have a universal loss recovery mechanism for all the packet losses (do the loss recovery without applying congestion control).

An extreme case of loss pattern is network congestion indications (ECN messages) are lost because of wireless error or sever network congestion. This case falls in the first scenario above. We consider there is no network congestion, and do appropriate retransmission without applying congestion control. But, our retransmission policies for random losses and bursty losses won't make the congestion condition worse. For the random losses, we retransmit those lost packets and full utilize the current congestion window. If the ECN messages are lost due to random wireless errors, ECN messages should be received by the SCTP source for the following window of data if the network congestion is persistent, thus congestion control will be applied. Moreover, ECN message is in small size with loss probability much less than data chunks. For the bursty losses, we

send a small probing packet first to test the channel condition, then retransmit lost packets and restore the full congestion window if the channel is good. If the ECN messages are lost due to the bad state of wireless link, the effect of the small probing message upon network load should be ignorable. After the window restoration, situation in bursty losses becomes the same as in the random loss case, and ECN messages will be received if persistent network congestion exists.

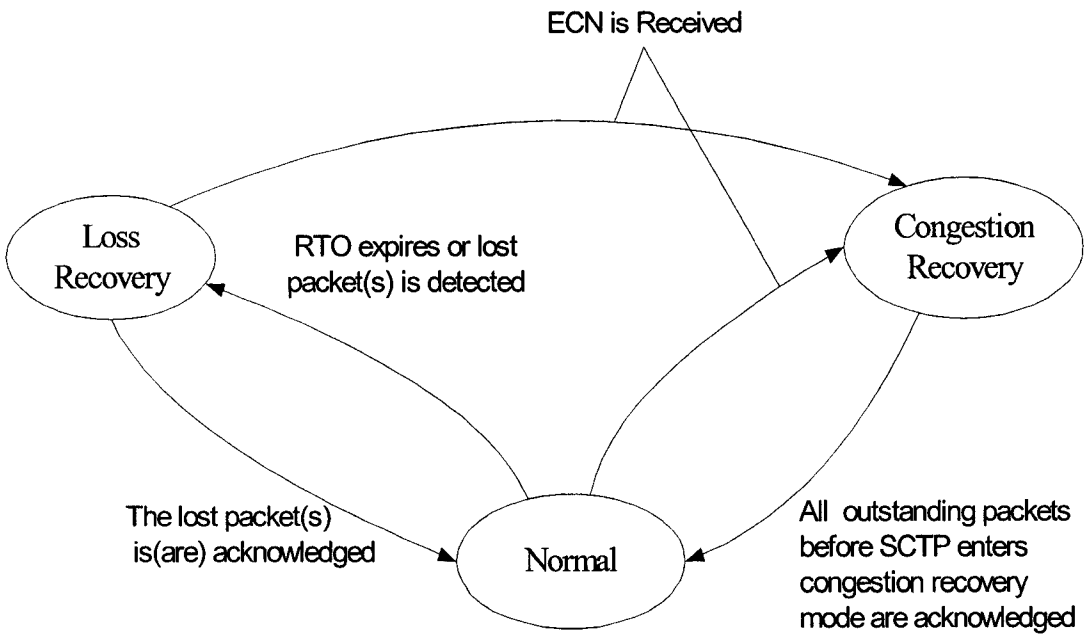


Figure 28. Mode transition of Wireless-aware SCTP

In Wireless-aware SCTP, we add a Loss-Recovery mode into the established state of the standard SCTP. This modification is made only at SCTP sources. In the normal mode, Wireless-aware SCTP operates as required by the RFC 2960. They work in either slow start phase or congestion avoidance phase. When network congestion happens, i.e., an ECN message is received, the Wireless-aware SCTP enters into the congestion recovery mode, and applies the congestion control mechanism as discussed in Chapter 4 (the

congestion control is allowed only once for a window of data). When packet losses associated with wireless errors are detected, the Wireless-aware SCTP enters Loss-Recovery mode, lost packets are retransmitted as per section 5.3.3 and 5.3.4 based on their nature (random or bursty loss).

5.3.2 The Retransmission Timer

An SCTP endpoint uses a retransmission timer to ensure data delivery in the absence of any feedback from its peer. The computation of SCTP's retransmission timer is based on the computation of TCP's retransmission timer [66, 67]. The duration of this timer is referred to as Retransmission Timeout (RTO). Upon timeout, the RTO will be reset according to the binary exponential back off scheme. In the loss-recovery mode, this timer is also necessary because we need to make sure that the retransmission is successful.

According to RFC 2960, new RTT measurements are contributed to the calculation of the RTO. The newly calculated RTOs are required to round up to the minimum RTO (MinRTO) if they are less than the MinRTO. The MinRTO recommended by the current SCTP specification is 1 second. In fact, this MinRTO is usually larger than RTTs of typical ad hoc networks. Since the distance between ad hoc nodes usually is at the order of one hundred meters, thus the propagation delay in the air can be ignored compared to the relatively much larger transmission delay due to the low radio bandwidth. For example, the RTT for an ad hoc network with 10 hops, 1 Mbps transmission rate and 1500 MTU is about 240ms, which is much less than the MinRTO (1 second) recommended by the current specification. Thus, it is very likely that the actual RTO of a SCTP association will always be MinRTO or 2^n*MinRTO after n consecutive timeouts,

and these values cannot reflect the real round-trip-time of the network. So, it would be a waste to do RTT measurement in MANET in most cases. Large value of MinRTO unnecessarily increases the waiting time for the SCTP to retransmit packets, which combined with the binary exponential back-off scheme of retransmission timer may result in "unsampled" good state at the wireless link. "Unsampled" good state is a good state at the wireless link, which is not utilized by the upper layer data transmission because of the idleness at the upper layer during this period.

The standard SCTP uses binary exponential back off scheme to reset the retransmission timer in the case of consecutive timeouts, which occur when retransmitted packets (due to timeout) are lost. The binary exponential back off scheme, designed to effectively avoid congestion collapse, is used in standard SCTP and TCP. However, we believe this scheme should be avoided if good performance, such as throughput, is preferred in Wireless-aware SCTP. Otherwise the W-A SCTP source will wait unnecessarily for a long time if the retransmitted packet (due to timeout) is lost because of random errors at wireless links. This timeout does not represent a congestion signal (ECN is the congestion signal), in other words, this timeout does not mean that there is a dramatic traffic increase in the network. So, network collapse should not be a concern when W-A SCTP operates in the Loss-Recovery mode. It is rare that consecutive timeouts occur because of random wireless losses because the probability that retransmitted packets are lost again is very small. However, the consecutive timeouts due to the bursty wireless error are not negligible especially when the bad state duration of wireless channel is larger than round-trip-time. In [64], it is found that some good states may be "unsampled" by the TCP after a visit to a bad state at a wireless link. The result is

that the effective bad state duration of the wireless link is much larger than the actual average bad state duration from the perspective of TCP. Thus further degrades the throughput performance. Figure 29 illustrates similar phenomenon in SCTP over bursty wireless channel. The wireless channel enters bad state at 40.4s and all packets in transit are lost in the early stage of the bad state, which results in two consecutive timeouts at SCTP source at about 41.3 and 43.3 respectively. After the first timeout, packets are retransmitted, however, they cannot go through the channel since it is still in the bad state, which results in the second timeout. Although the channel goes back to a good state at about 41.5s, it is not utilized because the SCTP source is idle at this time due to the burst of SCTP traffic – data packets are sent at early stage of the round-trip-time.

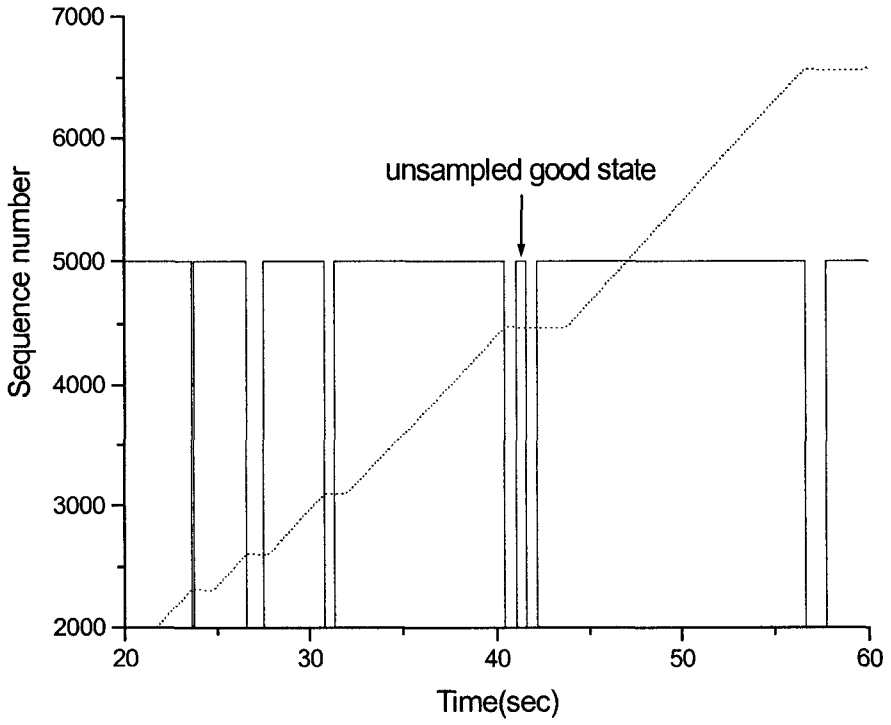


Figure 29. An example of unsampled good state

Through in-depth analysis and extensive simulation verifications, we find two factors determine the length of the effective bad state, i) the retransmission timeout value (RTO); and ii) the binary exponential back off of the RTO in the case of consecutive timeouts.

We recommend using the latest RTO value (which is calculated based on the latest RTT measurement) in the case of repeated timeout in Wireless-aware SCTP. However, a random factor should be added to this constant value in order to avoid periodic interference to the radio channel. Above all, we use a random value within the range of (RTO, 1.5 * RTO) as the retransmission timer when there is a timeout in the Loss-Recovery mode of wireless aware SCTP. We call it limited back off scheme in this work. In the congestion recovery mode, the traditional exponential back-off mechanism for the retransmission timer is followed to be compatible with normal SCTP.

Based on the previous discuss, we proposed that a smaller value of the minimum RTO should also be used for the Wireless-aware SCTP in MANET environment where round-trip-time usually is small and network congestion and wireless error coexist. Smaller value of the minimum RTO has already been adopted in Linux. In Linux version 2.4, the minimum value of RTO is 200 ms, and the retransmission timer granularity is of 10ms in default [68]. We do not provide rules about how to specify this value, however, we believe it should reflect the network round-trip-time and the effect of link level retransmission mechanism (if exists).

5.3.3 Random Losses

We use modified fast retransmit algorithm to handle random losses, which are characterized as the i.i.d packet level losses. The modified fast retransmit algorithm is the same as the fast retransmit algorithm in standard SCTP except that congestion window is

not halved in modified fast retransmit algorithm. In Wireless-aware SCTP, we use four missing reports to detect a packet loss. If there are no ECN messages received for the current window of data, this packet loss is determined as a random wireless loss. Upon detecting a random packet loss, the Wireless-aware SCTP enters the Loss-Recovery mode and retransmits the lost packet immediately without adjusting the congestion control parameters. The Wireless-aware SCTP source transmits new data chunks if the current congestion window allows.

It is possible that a Wireless-aware SCTP source may detect multiple packet losses at the same time. Although the probability that multiple packet losses in a window of data (less than 60 packets) is very small (Figure 30), this issue might not be negligible when the congestion window is very large. As to the retransmission policies under this condition, two solutions are considered, a) retransmit as many packets as possible (we call it AMAP policy) at a time if the current congestion window allows, and b) retransmit one packet per RTT. The AMAP policy is very aggressive, it aims to recovery those lost packets as soon as possible. The reasoning behind this policy is the congestion window should be fully utilized since there is no congestion in the network, i.e. no ECN message is received for the current window of data. The policy of retransmitting one packet per RTT is much more conservative than the AMAP policy. The policy of retransmitting one per RTT emphasizes the stability of the network, i.e. avoiding congestion collapse because it is possible, even the probability is small, that congestion losses may be mistakenly treated as wireless losses due to the loss of ECN messages. Although, congestion control will be applied in the following window of data if network congestion is persistent, we need to be cautious when doing retransmission. We try to understand

how these two policies affect the performance of Wireless-aware SCTP in recovering random packet losses by simulation in section 5.4. Other possible policies should have the effect in the middle of the polices we considered.

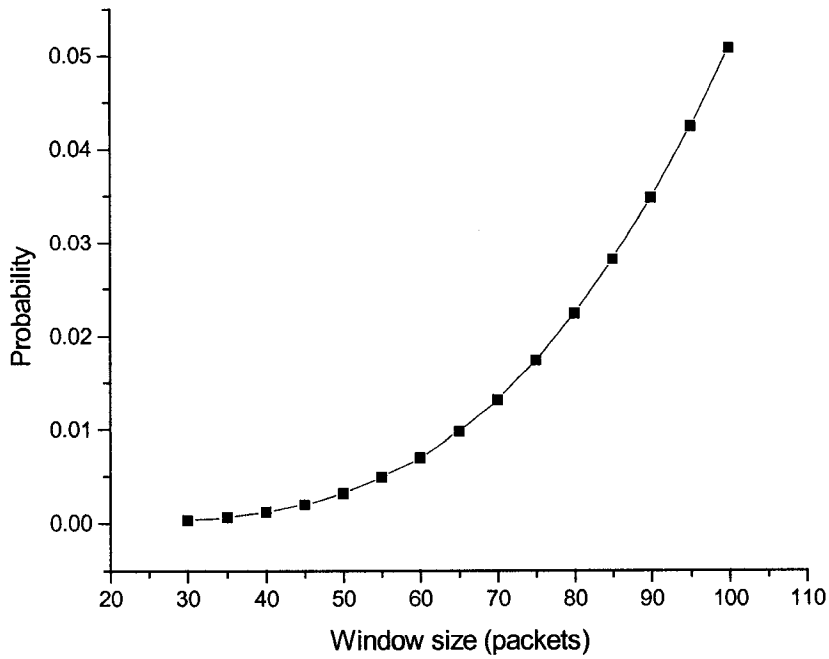


Figure 30. The probability that more than 4 packets are loss in a window (i.i.d packet loss rate $p = 0.02$)

5.3.4 Bursty Losses

As we discussed in section 5.2, a bursty wireless link can be modeled as a continuous time two-state alternating Markov chain [64]. The durations of time in good states are independent and identically distributed with exponential cumulative distribution function and mean G. The durations of time in bad states are independent and identically distributed with exponential cumulative distribution function and mean B. Let f represents the fraction of time the channel spends in the bad state.

$$f = \frac{B}{B+G}$$

In this work, we only consider wireless links with B at the order of the value of round-trip-time. Thus, a visit for the link to the bad state usually results in one or more timeout at the Wireless-aware SCTP source. For those B much less than the RTT, a visit at the link to the bad state has the same effect upon Wireless-aware SCTP source as random losses, thus it can be treated as a random loss. In Wireless-aware SCTP, a timeout event without ECN messages received for the current window of data is used to determine bursty losses. Since timeouts caused by random losses are rare, i.e. the probability that a retransmitted packet is lost again due to random error is very small, it is practically safe to associate timeouts with bursty losses. Next, we consider how to do the retransmission after a timeout event when Wireless-aware SCTP operates in the Loss-Recovery mode.

The standard SCTP blindly retransmits lost packets after each timeout event and uses binary exponential back off scheme to reset the retransmission timer. We think this retransmission scheme is neither energy efficient nor bandwidth efficient. Firstly, the binary exponential backing off mechanism may unnecessarily make the SCTP source wait too long (such as the "unsampled" good state discussed in section 5.3.2) for retransmission thus results in the reduced throughput performance. Secondly, it has no idea whether the underlying wireless link is in good state or not at the time of retransmission. It would be a waste of energy and bandwidth to retransmit when the underlying wireless link is in bad state.

To overcome the blindly retransmission in standard SCTP, we suggest that the Wireless-aware SCTP source send a small probing packet first after each timeout. The limited back off scheme is also used for the probing packet when there is a timeout. After an acknowledgement is received for the probing packet, the Wireless-aware SCTP source then retransmit as many packets marked for retransmission as possible at a time as long as the congestion window allows.

For small fading wireless link (B is at the order of RTT and f is small), we have very high confidence that retransmission will be successful, i.e. the wireless link is in good state during the time of retransmission. We start from the simplest case: there is only one hop between the SCTP source and destination. Suppose, at time t_0 an acknowledgement is received for the probing packet, i.e. the channel is in good state at this time. Apparently, the probability that retransmission is successful (if there is not other interference) is equal to the probability that the channel is still in good state at t_0+RTT given it is in good state at t_0. This is equal to the probability that good state duration is at least RTT according to the memoryless feature of exponential distribution.

$$Prob(good\ state\ duration > RTT) = exp(-RTT/G) \approx exp(-f)$$

$exp(-f)$ is close to 1 when f is small. When there are multiple hops (see n hops) between source and destination, for independent wireless links, the probability that retransmission is successful is equal to $exp^n(-f)$. For moderate n, we still have high confidence that retransmission will be successful. With the increase of hop number n and f, we believe no transport protocol can perform well in such an extreme case without the assistance from the link layer. We think certain intelligent link layer transmission scheme should be developed that can temporarily store data for transmission when the wireless link is in

bad state, especially when f is high. Certainly, this delay effect at the link layer should be considered when calculating the RTO at the transport layer otherwise spurious timeouts will become a big issue.

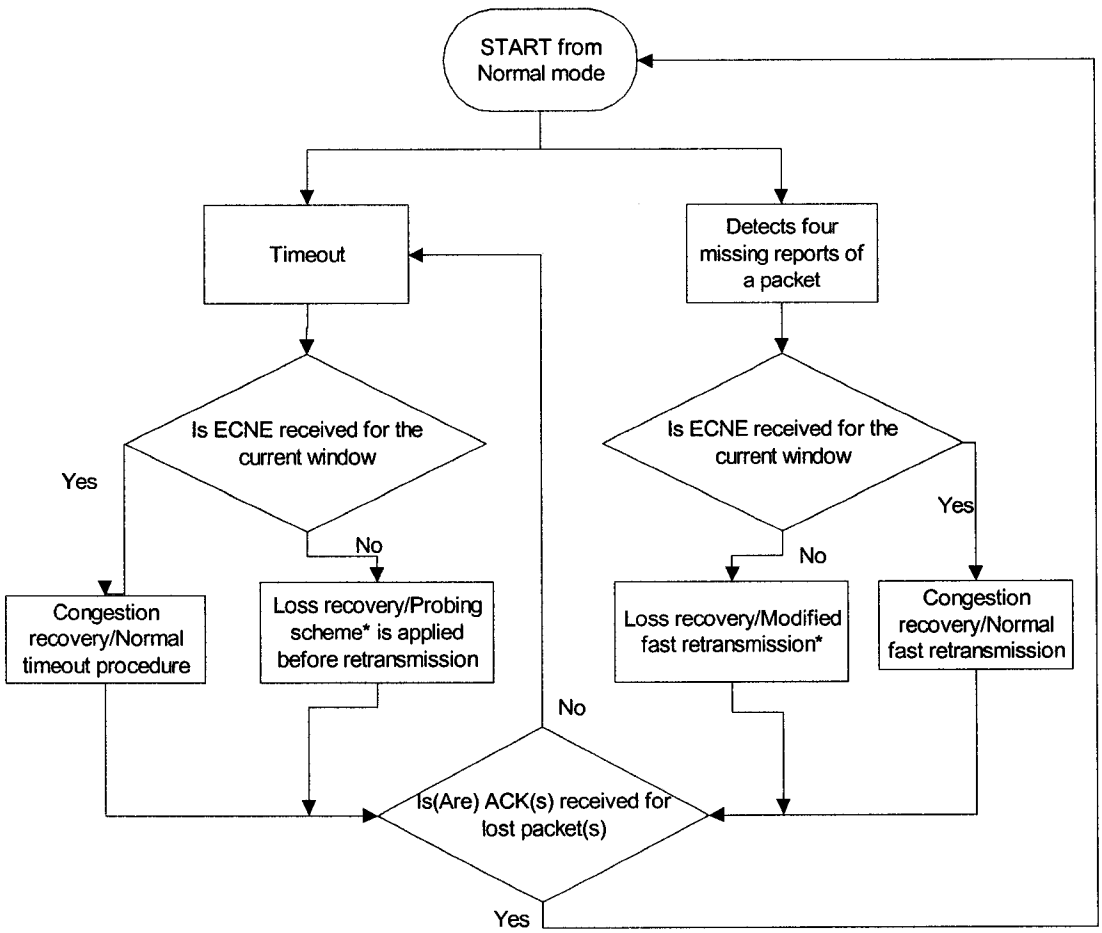


Figure 31. Details of Wireless-aware SCTP

Figure 31 shows details of the operation of Wireless-aware SCTP. Suppose Wireless-aware SCTP starts from normal mode, which means neither network congestion nor wireless error exist in the network currently. When a missing packet is detected (using four missing reports as required in standard SCTP), Wireless-aware SCTP checks if an ECN message has been received for the current window of data or not. If yes, this

packet loss is considered congestion loss, congestion recovery mode is entered and normal congestion control procedure is executed; If no ECN message is received for the current window of data, this packet loss is considered as random wireless loss, the modified fast retransmit algorithm discussed in section 5.3.3 is used to recover the lost packet. When the retransmission timer at the Wireless-aware SCTP sender runs out, the Wireless-aware SCTP first check if any ECN message has been received for the current window of data. If yes, this timeout is considered network congestion, thus congestion recovery mode is entered and normal timeout procedure required by the standard SCTP is followed. If there is no ECN message received for the current window of data, this timeout is considered caused by bursty wireless errors, the probing scheme discussed above is applied first before retransmission.

5.4 Performance Evaluation

5.4.1 Methodology

We choose simulation to have a reproducible and controllable environment with our proposed Wireless-aware SCTP. All simulation is conducted in the network simulator OPNET [44].

The topology in Figure 32 is used in our simulation studies. Although multi-hop effect of wireless link can be abstracted as higher random loss rate and longer bad state duration than that of single wireless link, we purposely choose a 5-hop simulation topology to make it close to real network. The wireless loss associated with radio channel is emulated at the node R1-R4. Two-state Markov alternating chain is used at the link level. When the channel is in good state, packets in transit are dropped randomly with probability p. All packets in transit will be dropped when the channel is in bad state.

Large file transfer is assumed between the source and destination. RED enhancement [65], which considers both the average queue length and the actual queue length at times of marking/dropping packets, is used at the node R2. The RED parameters are: queue weight $w = 0.002$, minimum threshold = 3, maximum threshold = 10, maximum marking probability = 50% (to maintain the average queue length around the minimum threshold). On-off constant bit rate (CBR) traffic between the CBR source and receiver is used to periodically flood the link between node R2 and R3, which creates periodic congestion in the network. The CBR traffic starts every 5 seconds with duration of 200 ms. Propagation delay at each link is zero since propagation delay is almost negligible compared to transmission delay in ad hoc networks. Path MTU of 1500 bytes is used in the simulations. All links has the bandwidth of 2 Mbps.

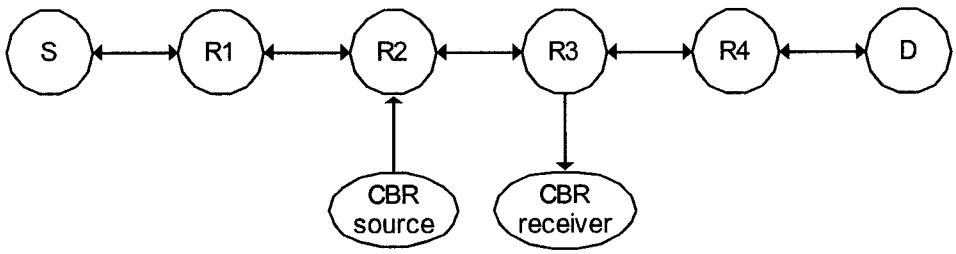


Figure 32. Simulation topology

5.4.2 Results and Discussions

As we have discussed in section 5.3.1, by using ECN as the congestion notification, wireless losses can be differentiated from congestion losses, and thus congestion control mechanism won't be applied for wireless losses. As we know, the congestion window of Wireless-aware SCTP or standard SCTP is either halved or reduced to one MTU when network congestion occurs. Thus, the change of congestion window is a good indication

of applying congestion control mechanism. Figure 33 and 34 are examples of the congestion window size in two scenarios. In the first scenario, wireless error (random error or bursty error) is the only reason for packet losses. We have a total of 1% random loss rate and B = 0.1s, G = 9.9s at each link. We expect that the congestion window of Wireless-aware SCTP should remain constant in the steady state of the association. Simulation result is in accordance with our expectation, the congestion window of W-A SCTP remains at 33*MTU after the slow start of the association, which is exactly one MTU plus the maximum receiver side window size according to the RFC 2960. This implies that the W-A SCTP 100% correctly identifies all the wireless losses; otherwise, congestion window will be reduced if wireless losses are considered as congestion losses. However, the congestion window of the standard SCTP varies dramatically compared to that of the proposed algorithm. This is because that standard SCTP treats all wireless losses as congestion losses and applies congestion control algorithms when it detects packet losses (all packet losses are random wireless losses in this case).

In the second scenario, network congestion and wireless error co-exist. The settings for the wireless links are the same as in the scenario 1. However, we specially choose the CBR traffic to periodically generate network congestion, thus the oscillation of the congestion window of our Wireless-aware SCTP should have similar periodic feature. The CBR traffic starts at simulation time 10s. As we expected, the congestion window of the W-A SCTP is reduced only at times of 5s, which is the period of network congestion caused by the CBR traffic. Moreover, the oscillation of the congestion window does not vary as dramatically as that of standard SCTP because of the rule of applying congestion control only once for a window of data, which is not used by the standard SCTP.

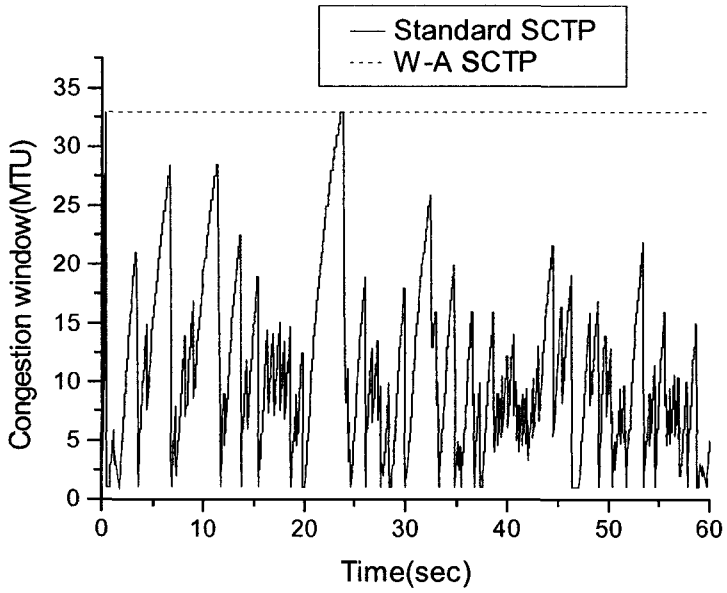


Figure 33. A snapshot of congestion window in the presence of only wireless losses

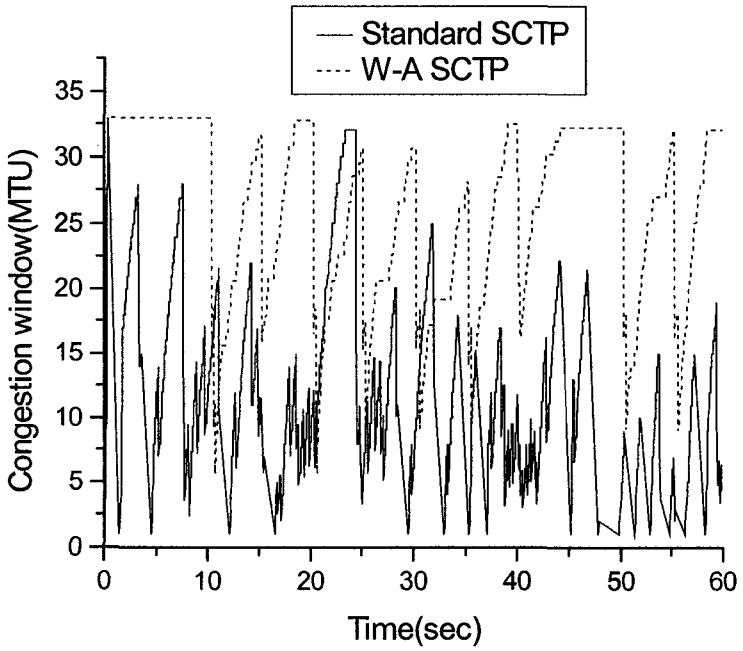


Figure 34. A snapshot of congestion window in the presence of both congestion and wireless losses

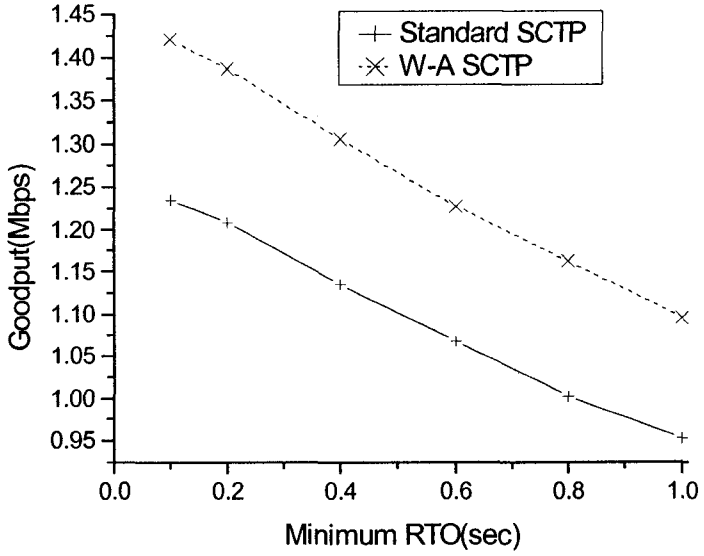


Figure 35. The effect of minimum RTO upon goodput

The effect of the minimum RTO upon the SCTP performance is illustrated in Figure 35. Goodput, which is defined as the average receiving rate excluding duplicate packets at the SCTP receiver, is used as the evaluation metric. At each wireless link, the bursty loss rate is fixed with average bad state duration of 0.1s and average good state duration of 9.9s. The total random loss rate at the wireless channels is varied from 1% to 10%. The simulation results strongly support the use of small values of MinRTO. As we expected, the larger the MinRO, the worse the goodput performance. This is because that the interval between timeouts of SCTP with larger MinRTO is larger than that of SCTP with smaller MinRTO in the case of consecutive timeouts. One concern of using small values of MinRTO might be the spurious timeouts, which refers to the timeouts that are not resulted from packet losses but from the sudden increase of the RTT to the extent that it exceeds the retransmission timer that had been determined a priori. Eifel algorithm, which is based on the TCP's timestamp option [72], might be adopted into SCTP to deal

with spurious timeouts [68-71]. As discussed in section 5.3.4, a small probing packet is sent first after each timeout when the SCTP operates in the Loss-Recovery mode, and those lost packets are retransmitted only if the probing packet is acknowledged. If this timeout is a spurious timeout, acknowledgments for outstanding packets should arrive before the acknowledgment of probing packet, thus the retransmission will be canceled. So, spurious timeouts should not be a concern when the Wireless-aware SCTP operates in the Loss-Recovery mode. If SCTP operates in a very dynamic environment where spurious timeouts are pervasive. One may consider using conservative MinRTO (e.g. one second) when SCTP operates in Normal and Congestion Recovery mode, and using a smaller MinRTO when SCTP is in Loss-Recovery mode. In practical, one needs to record the latest RTO calculation in a state variable before it enters the Loss-Recovery mode, this RTO should round up to the MinRTO (the smaller one) if it is less than this value.

Table 2 shows the effect of retransmission policies for random losses and retransmission timer resetting schemes upon the goodput performance for a Wireless-aware SCTP association. To isolate the effect of bursty losses, only random losses are introduced in the network in this case. The total random loss rate (p) associated with wireless links varies from 1% to 6%. Each goodput value in the table is the average of 5 runs of the simulation lasting for an hour. In general, the AMAP policy performs better than the one per RTT policy, and the limited back off scheme is better than the exponential back off scheme in terms of goodput. However, the difference is small as shown in Table 2. Since the back-to-back loss (i.e. more than two conjunctive packets are lost in a window of data) probability is very small, and the probability that the retransmitted packet is lost again is very low, two retransmission policies operate almost

identically when random loss rate is small. With the increasing of p, the advantage of using the limited back off scheme over the exponential back off scheme starts to stand out because of the increased number of timeout events. For example, about 0.5-1.1% goodput gain can be achieved by using the limited back off scheme over the exponential back off scheme when p=6%. However, we should note that this gain in goodput is small. Considering the comparative performance of the two retransmission-schemes within the typical loss rate of radio channel [73], and the conservativeness of the one per RTT retransmission policy discussed in section 5.3.3, we recommend using the one per RTT retransmission policy for random wireless losses.

p(%)	AMAP		One per RTT	
	Exponential back off	Limited back off	Exponential back off	Limited back off
1	1.53359	1.53358	1.53341	1.53356
2	1.39398	1.39427	1.39315	1.39407
3	1.28014	1.28065	1.27858	1.28033
4	1.18369	1.18555	1.18025	1.18467
5	1.10331	1.10643	1.09774	1.10488
6	1.03228	1.03773	1.02358	1.03515

Table 2. Effects of retransmission policies and retransmission timer

For mobile users and operators, the battery power consumption and radio resource preservation are often as important as the goodput performance of data transmission. We

therefore used the goodput performance and the T-R ratio, which is defined as ratio of total traffic transmitted (by the sender and receiver) and the total traffic received (by the receiver that contributes to the goodput), as equally important performance metrics. To demonstrate that the probing scheme discussed in section 5.3.4 for good channel state is superior over the blindly retransmission scheme in standard SCTP, we design a scenario where there exist only bursty errors at wireless links, and there is no network congestion. Small fading channel is emulated, where the $B+G$ is much larger than the RTT. Two retransmission schemes for timeout events are compared. In the first scheme, a probing packet is sent after a timeout event to probe the state of the channel. The limited back off scheme is used for probing packets in the case of timeout. Only after the receiver acknowledges the probing packet, then the Wireless-aware SCTP source retransmits as many packets marked for retransmission as possible. The Heartbeat chunk and Heartbeat acknowledgment chunk are used as the probing packet and acknowledgment for the probing packets respectively. Thereafter, we call the first scheme HB+AMAP scheme. In the second scheme, no probing packet is used in the implementation of Wireless-aware SCTP. In this scheme, the Wireless-aware SCTP source retransmits as many packets marked for retransmission as possible after each timeout event (we call it AMAP scheme). The second scheme follows the blind retransmission feature of standard SCTP and it tries to retransmit as many packets as possible, which should result in highest goodput performance. Figure 36 shows the goodput of Wireless-aware SCTP under these two different retransmission schemes upon timeout. As we expected, the Wireless-aware SCTP using AMAP scheme achieves higher goodput than the Wireless-aware SCTP using the HB+AMAP scheme. However, the difference is small (in the range of 6kbps to

12kbps) and the difference seems have no direct relationship with the fraction of bad state. Figure 37 shows the T-R ratio of Wireless-aware SCTP under these two different retransmission schemes upon timeout. The T-R ratio of Wireless-aware SCTP using the AMAP scheme is persistently higher than the Wireless-aware SCTP using the HB+AMAP scheme, which means that HB+AMAP scheme requires less use of the transmitter (i.e. less energy) than the AMAP scheme to achieve same goodput. On the contrary to the goodput performance in Figure 36, the difference of T-R ratio under these two retransmission schemes increases with the increase of the fraction of bad state *f*. From Figure 37, we can see that the T-R ratio of Wireless-aware SCTP using the HB+AMAP scheme remains almost a constant value under the various fraction of bad state used in the simulation, which means that the HB+AMAP scheme is rather stable under different network conditions. This is a desirable feature of HB+AMAP scheme. However, the T-R ratio of Wireless-aware SCTP using the AMAP scheme increases linearly with the increase of the fraction of bad state, which shows that the AMAP scheme is not stable when network condition changes, and it consumes more energy (or battery power) when network condition is bad.

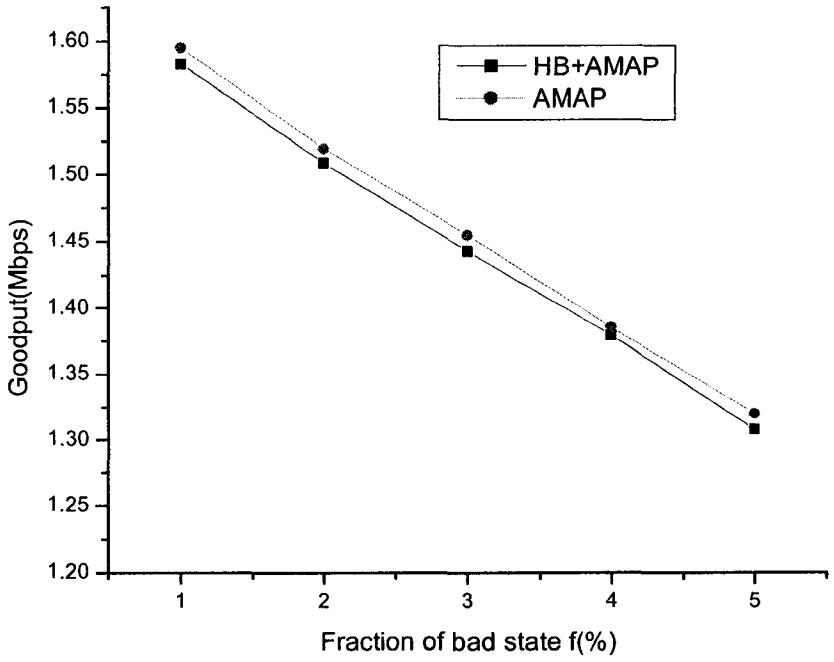


Figure 36. Goodput vs. fraction of bad state *f*

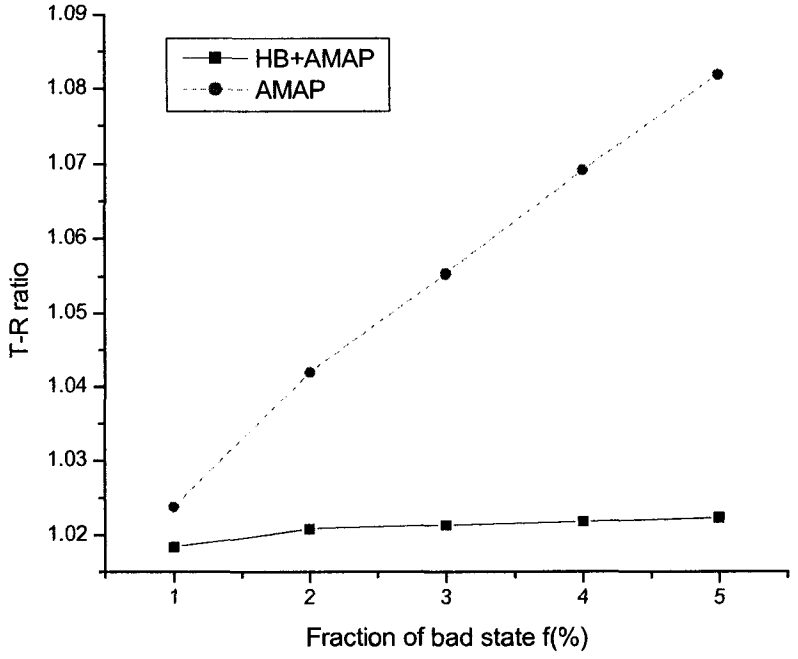


Figure 37. T-R ratio vs. fraction of bad state *f*

Chapter Six

6 Independent per Path Congestion Control

For multi-homed hosts, which can be reached through multiple paths, a new challenge would be how to efficiently and simultaneously use multiple paths between them to explore potential benefits of multi-homing feature, such as to improve connection reliability, to increase throughput etc. In this chapter, we identified and analyzed limitations of the congestion control mechanism of standard Stream Control Transmission Protocol (SCTP) in the case of simultaneously using multiple paths. Simultaneously using multiple paths is inherent in multi-homing applications. We proposed an independent per path congestion control framework for standard SCTP to overcome these limitations. The proposed Independent per Path Congestion Control (IPCC) SCTP can be used for various applications of the multi-homing feature when appropriate data-striping algorithms are used. Section 6.1 gives a brief introduction of the motivation of this work. Section 6.2 discusses related work on multi-homing. Section 6.3 introduces the congestion control mechanism of the standard SCTP and illustrates its limitations in supporting multi-homing efficiently. Section 6.4 presents details on how to realize the proposed independent per path congestion control in the standard SCTP. Section 6.5 reports the performance evaluation of IPCC-SCTP. Section 6.6 discusses issues related to the implementation of IPCC-SCTP.

6.1 Introduction

Typically, a multi-homed entity can be reached through more than one physical interface, and each interface has a different IP address if IP is used as the network protocol. Such a multi-homed entity can be an endpoint, a router/switch and a cluster of endpoints/routers, which are formed as one entity to fulfill complicated tasks, e.g. the concept of reliable server pool [74]. With the advancement of communication technologies, especially the wireless technology, a host may be equipped with multiple wired and/or wireless interfaces. It is natural to consider concurrent transmission using multiple paths between multi-homed communication hosts in order to fully explore potential benefits of the multi-homing feature, such as to improve connection reliability, to increase throughput etc. However, concurrent transmission using multiple paths between multi-homed hosts has not been well studied in the literature, especially issues related to congestion control mechanism when striping data at the transport layer.

A transport layer protocol that supports multi-homing usually consists of two components, a data-striping algorithm and a data delivery mechanism. The data-striping algorithm is responsible for distributing data to different paths according to certain QoS requirements. The data delivery mechanism is responsible for data transmission and reception, and its core is congestion control. Although the first component may vary among different applications of the multi-homing feature, the second component should be quite similar because they share a common feature, using multiple paths simultaneously. In this work, we proposed an enhancement to the standard Stream Control Transmission Protocol (SCTP). The enhanced SCTP can be used for various applications of the multi-homing feature with appropriate data-striping algorithms.

Standard Stream Control Transmission Protocol (SCTP) allows a transport layer connection to cross multiple source and destination addresses. Suppose one destination address represents a path to the destination, there would be multiple different paths to reach a multi-homed hosts. Different destination addresses (or paths) can be used for data transmission without the application awareness. This is the so-called multi-homing feature of standard SCTP. The standard SCTP uses the multi-homing feature only for retransmission and failover, i.e., using alternate destination address for retransmission in order to increase the possibility to reach remote endpoint and switching to another destination address without interrupting the end-to-end transmission when one destination address fails. Other multi-homing applications such as load sharing and load balancing are not supported by standard SCTP. Although, standard SCTP, in functionality, supports some multi-homing applications, we found its congestion control algorithms, which are derived from TCP congestion control [35], are not designed with multi-homing in mind. The main drawback is the per association congestion control framework, which becomes unfeasible when multi-homing feature is used.

We propose that a novel framework: **independent per path congestion control**, must be strictly followed by standard SCTP in order to simultaneously and efficiently use multiple paths between communication hosts. However, the design of the standard SCTP breaks this framework at several aspects. Further discussion of these limitations is presented in section 6.3. Thereafter, we refer to SCTP conforming to RFC 2960 and its latest Implementer's Guide [75] as standard SCTP, and SCTP supports our proposed independent per path congestion control as Independent per Path Congestion Control SCTP (IPCC-SCTP).

6.2 Related Work

A multi-homed endpoint can utilize the multi-homing feature at application layer, transport layer, network layer or link layer. Application layer can simply use multiple connections to explore the benefits of multi-homing. In [76, 77], approaches have been proposed to improve the application throughput by creating multiple TCP sockets between two hosts, however in a different context where multiple TCP connections share same path with large bandwidth delay products. If an application uses multiple TCP/SCTP connections for bandwidth aggregation between multi-homed hosts, it needs to implement its own data striping and resequencing schemes. And vast bandwidth/delay differences at different paths usually require very large resequencing buffer at the application layer in order to avoid head-of-line block. Otherwise, fast connections would be stalled by slow connections because of no buffer space at the receiver side.

If the multi-homing were implemented at layer 2/3, extra efforts are needed at the receiver side to handle synchronization; otherwise, out-of-order arrival would inversely affect the congestion control performance of transport layer (e.g. TCP). Multi-link PPP [78-80] is designed to aggregate multiple logical data channels into one. If IP addresses of different interfaces are assigned by different independent Internet Service Providers (ISPs), it is highly unlikely that the independent ISPs will allow arbitrary users to bundle their links into "one logical link".

Multi-path and QoS routing have been considered in wired and wireless networks, such as work in [81-85]. However, these work concentrates mainly on the network layer, and the issue of splitting a single connection into multiple streams is not considered. [86] proposed a new mechanism to aggregate bandwidth of multiple IP links by splitting a

data flow across multiple network interfaces at the IP layer for mobile equipments. The authors also suggested using an out-of-order transmission scheme at TCP to prevent recording at the receiver side, thus avoid TCP fast retransmission, but this scheme works well only when there is no packet loss along each path. Since it is a trend for transport protocols to be "TCP friendly" [87], supporting multi-homing at layer 2/3 becomes undesirable because of its side effect to the congestion control algorithms at transport layer.

[88] is a measurement-based analysis of multi-homing, which provides quantified study of the extent to which multi-homed networks can leverage performance and reliability benefits from connections to multiple service providers. The study suggests that while multi-homing offers substantial performance benefits, however a careful choice of upstream providers is crutial. Load sharing is also known as channel striping. A summarization of various approaches of load striping can be found in [89]. A model of channel striping is reported in [90], in which a channel is defined as a logical FIFO path, and fair queuing algorithms are proposed to be used in load sharing. Load balancing has been extensively studied in the context of web-server selection over Internet [91-94].

[95-96] proposed a new load sharing scheme for standard SCTP. However, the proposed load sharing scheme requires changes of the standard SCTP at both the sender and receiver sides, and introduces extra overhead by adding new fields into SCTP DATA chunk. Our work does not consider a specific data-striping algorithm for a particular multi-homing application. However, we try to design a general congestion control framework that should make SCTP feasible for a large number of multi-homing

applications that share a common feature, simultaneously using multiple paths. And we also try not to add extra overhead to the network when the multi-homing feature is used.

6.3 Limitations of the congestion control of standard SCTP

The standard SCTP bases its congestion control on TCP congestion control principles and uses the SACK extension of TCP. Unlike TCP, the standard SCTP is a message-oriented transport protocol. It uses Transmission Sequence Number (TSN), which represents the order of data chunks leaving the transport layer, to track the transmission and reception of each data chunk. Congestion control of the standard SCTP is applied to an entire association (RFC 2960, section 7) although it might use multiple source/destination IP addresses, or multiple paths.

The retransmission performance of the standard SCTP was studied in [97, 98]. The authors revealed an interesting phenomenon: retransmission through another path is not always beneficial to throughput performance of the standard SCTP. The throughput performance of the standard SCTP, which transmits retransmitted packets through another destination interface (or path), is degraded when the retransmission path has high packet loss rate (larger than the packet loss rate at the path for original transmission). The authors considered the inaccurate RTT measurement at the retransmission path is the reason for such performance degradation. However, the result is contrast to the intuition: the performance should be better given more resources, the primary path plus the retransmission path. Although the retransmission path has high packet loss rate and inaccurate RTT measurement, it represents extra resource to the primary path. We think there must be limitations of the retransmission algorithm of the standard SCTP when the multi-homing feature is used. In fact, the standard SCTP enters fast recovery mode after

it executes fast retransmit algorithm at the primary path. It cannot exit fast recovery mode if the retransmitted packet(s) is(are) not acknowledged by the destination. The loss of retransmitted packet(s) at alternate path delays the standard SCTP exiting the fast recovery. Thus, the throughput performance is degraded. The larger the RTT at alternate path, the more apparent is the performance degradation.

[39, 99, 100] reported a congestion window overgrowth problem of standard SCTP during a changeover, changing the primary destination address (or the primary path) while there is traffic in flight to the original primary destination address. The authors found that when the end-to-end delay at the new primary path is smaller than the end-to-end delay at the original primary path, the congestion window at the new primary path may overgrow. The authors think the congestion window overgrowth problem is caused either by the sender's inability to distinguish SACKs for transmissions from SACKs for retransmissions, or by the sender's unawareness of the occurrence of a changeover. However, they failed to point out that it is the per association congestion control of standard SCTP that fails to efficient support using multiple paths simultaneously.

By studying the retransmission problem and the congestion window overgrowth problem carefully, we find these problems share a common feature: they occur when SCTP simultaneously uses two or more destination addresses (or two or more paths) for a short period of time. In fact, simultaneously using multiple paths is inherent in the multi-homing feature. We find load sharing and load balancing at the transport layer would also need the data-delivery mechanism to simultaneously use multiple paths, but load sharing and load balancing may require longer time of simultaneously using multiple paths. When an SCTP association simultaneously uses multiple paths, per association

congestion control of the standard SCTP becomes unfeasible because each path may have different characteristics, such as available bandwidth, round-trip-time (RTT), packet loss rate etc. Using per association congestion control, transmission/reception at one path may affect the transmission/reception at another path when multiple paths are simultaneously used by an SCTP association, and thus problems like those reported in [97-100] may occur. The standard SCTP cannot efficiently support the multi-homing feature because the congestion control algorithms of the standard SCTP break the principle of independent per path congestion control at the following aspects.

1) The congestion control of the standard SCTP is applied to an entire association. Thus, TSNs transmitted at one path may affect the transmission of TSNs at another path when multiple paths are simultaneously used. Next example helps to illustrate this limitation. As shown in Figure 38, the SCTP source endpoint has two interfaces S_1 and S_2, and the SCTP destination endpoint has two interfaces D_1 and D_2. Suppose there are two disjoint paths between them, see path 1 (S_1- D_1) and path 2 (S_2- D_2), and the transmission rate at path 2 is 4 times that of path 1. Assume at certain time, both destination addresses have congestion window that allows 4 DATA chunks outstanding. Assume that the source endpoint sends to D_1 4 SCTP packets, each packet contains one DATA chunk (TSN 20, 21, 22, 23), and sends to D_2 4 SCTP packets (TSN 24, 25, 26, 27) too at the same time for load sharing purpose. Certainly, TSNs 24, 25, 26 27 arrive at D_2 before the arrival of TSN 20 at D_1 due to the larger transmission rate at path 2. These out-of-order TSNs will trigger four SACK chunks at the receiver reporting the missing of TSNs 20, 21, 22, and 23. When these four SACK chunks arrive at the source endpoint, TSNs 20, 21,

22, 23 will be fast retransmitted according to the SCTP specification, RFC 2960. On the one hand, fast retransmission of TSNs 20, 21, 22, and 23 unnecessarily reduces the transmission rate at path 1 (in fact, there is no network congestion at path 1 at all); on the other hand, it wastes valuable network bandwidth to transmit TSNs 20, 21, 22, and 23 again. This example shows that the transmission rate at path 1 is halved because of transmissions at path 2 although there is no network congestion at path 1 at all. In order to eliminate this kind of mutual effect between paths, we propose that per path congestion control should be supported when an SCTP association simultaneously uses multiple paths. Under the per path congestion control, each path has its own congestion control mechanism and the operation of the congestion control at each path depends only on the transmission/reception of TSNs at its own path.

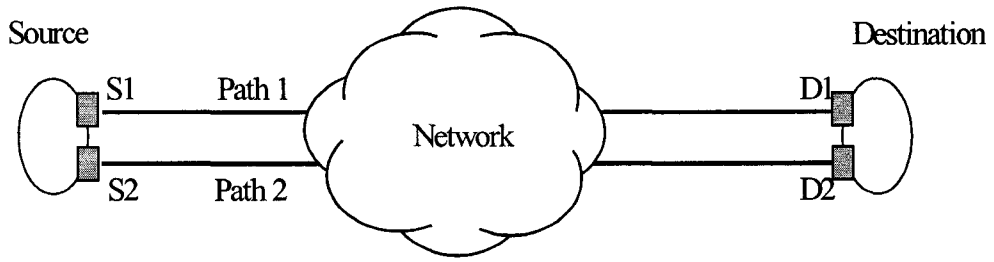


Figure 38. An SCTP association with two paths

2) The update of congestion control parameters for each destination address is not solely dependent on the transmission condition of the path they belong to. The standard SCTP defines separate congestion control parameters, such as congestion window and slow start threshold etc., for each destination address that an SCTP association uses. It seems that independent congestion control is applied to available paths (assume each destination address corresponds to a different path to

reach the destination endpoint). However, the per association congestion control mechanism of standard SCTP makes the update of congestion control parameters for each path not solely dependent on the transmission condition of the path they belong to. This will cause problems, such as the one reported in [99]. Continuing the previous example, suppose TSNs 20, 21, 22, and 23 are retransmitted through path 2. If original transmissions of TSNs 20, 21, 22 and 23 through path 1 arrive before their retransmissions, SACKs for these TSNs will make the source increase the congestion window for path 2 although these SACKs are for TSNs received from path 1.

3) It is stated in the standard SCTP (section 7, RFC 2960) that congestion window cannot be increased without the advancement of the Cumulative TSN Ack Point. This means that at times of retransmission, due to fast retransmission or timeout, the SCTP transmission procedure refrains from increasing the corresponding congestion window of the destination address at which the fast retransmission or timeout recovery algorithm are executed until the retransmitted packets are acknowledged. This conservative mechanism works well for SCTP associations between endpoints with only one interface. And it is advocated by TCP too. However, for SCTP associations using multiple paths simultaneously, this retransmission scheme may make the transmission procedure at one path wait for the retransmission of TSNs at another path if retransmissions are scheduled at alternate path. If the network condition at the path for retransmission is not good enough, it may result in the retransmission problem reported in [98]. If we apply

the rule of independent per path congestion control to the standard SCTP, these problems will not occur.

6.4 Congestion Control for IPCC-SCTP

In order to efficiently support multi-homing, we propose that IPCC-SCTP needs to meet the following requirements:

1) Per path congestion control, each path applies its own congestion control algorithms.
2) Congestion control scheme for one path should be independent to that of others, i.e. packet transmission or network condition at one path should not affect the congestion control algorithms at other paths.
3) Additive increase, multiplicative decrease (AIMD) principle should be applied at each path in order to be TCP friendly. For IPCC-SCTP, the congestion control algorithms at each path should follow the rules specified in section 7 of RFC 2960 and extensions proposed in this Chapter.

In section 6.4.1, we introduce the concept of Path Sequence Number, which helps to break down the per association congestion control framework of standard SCTP and facilitates the using of per path congestion control. In section 6.4.2, we discuss in detail on how to achieve independent per path congestion control in standard SCTP.

6.4.1 Using Path Sequence Number

The standard SCTP uses Transmission Sequence Number (TSN) as sequence number in its congestion control algorithms. TSN is used by the entire SCTP association although the SCTP association might simultaneously use multiple paths. In order to have

independent congestion control over each path, we should decouple TSN from congestion control. Thus, we need a new sequence number that can be used for congestion control. We propose to use Path Sequence Number (PSN). PSN represents the sequence of SCTP DATA chunk transmitted through a path; it has similar function as TSN in standard SCTP and sequence number in TCP. There are two ways to incorporate PSN into standard SCTP, a) explicit PSN, and b) implicit PSN. Explicit PSN means that each SCTP DATA chunk carries a PSN explicitly. In implicit PSN, an SCTP DATA chunk does not need to explicitly carry a PSN; however an SCTP source maintains a local mapping of TSNs and PSNs for each path. The usage of PSN in [95] belongs to the former. The explicit PSN scheme needs to modify the format of SCTP DATA chunk and introduces extra overhead to network. In this work, we use the implicit PSN scheme, which does not have the drawbacks of the explicit PSN scheme.

By using implicit PSN, an SCTP source tracks the path an SCTP DATA chunk sent through, and allocates a unique PSN from the path for the DAT chunk. Thus a local mapping between TSNs and PSNs is created for each path. Since PSN is used only by SCTP sources to keep track of the status of data chunks sent through each path, it does not need to be sent to SCTP receivers. Thus, we don't need to modify the format of SCTP DATA chunk, and no extra overhead as in [95] is introduced. Another benefit of using implicit PSN scheme is that no modification is needed at an SCTP destination endpoint. So, an SCTP receiver can act the same as specified in RFC 2960 if the two level flow control buffer structure discussed in section 6.6 is not applied. To facilitate per path congestion control, the requirements for PSN assignment at each path are:

1) Each TSN should be assigned a unique PSN from the path it is sent through;

2) The PSN at each path is monotonically increased;

3) PSN should represent the sequence of SCTP DATA chunks sent through a path; and

4) The initial value of PSN at each path can be any integer since it only has local importance.

6.4.2 Modifications at SCTP source side

In this section, we discuss three most important changes made at a standard SCTP source to realize independent per path congestion control.

6.4.2.1 PSN-based congestion control

In order to do per path congestion control, each destination address (or path) needs its own congestion control parameters: congestion window (cwnd), slow start threshold (ssthresh), retransmission timer. And each path needs to execute its own round-trip-time (RTT) measurement, which should follow the rules specified in RFC 2960. The algorithms of slow start, congestion avoidance, fast retransmission, fast recovery and timeout retransmission used by the standard SCTP will all be applied to each path. However, they use PSN to facilitate their operation, instead of using TSN. Accordingly, the Cumulative TSN Ack Point is not used in IPCC-SCTP any more, and the Cumulative PSN Ack Point is used for each path. The Cumulative PSN Ack Point represents the sender's view of the highest PSN in sequence that has been received so far from a path at the receiver.

The slow start algorithm for each path should follow the rules described in section 7.2.1 of RFC2960, except that the increase of congestion window is determined by the advancement of the Cumulative PSN Ack Point. Only when an incoming SACK

advances the Cumulative PSN Ack Point for a path and current congestion window of that path is being fully utilized, then the congestion window of that path is allowed to be increased by at most the lesser of 1) the total size of the previously outstanding DATA chunk(s) acknowledged at that path, and 2) the destination's path MTU. Similarly, the Cumulative TSN Ack Point for congestion avoidance in section 7.2.2 of RFC 2960 should be replaced by the Cumulative PSN Ack Point. Fast retransmission is triggered by four missing reports of PSN(s) at a path. Fast retransmit algorithm follows rules in section 7.2.4 of RFC 2960 and section 2.8 in [75].

6.4.2.2 Handling SCTP SACK chunk

SACK chunks (which can be called TSN-based SACK) received by an IPCC-SCTP source endpoint actually reflect the reception of TSNs sent through all the paths used by an IPCC-SCTP association. However, the information contained in SACK chunks cannot be directly used by the IPCC-SCTP source endpoint, because the congestion control of IPCC-SCTP now is based on PSNs at each path. So, a translation between TSNs and PSNs is needed here. A source endpoint needs to derive the Cumulative PSN Ack Point, PSN-based Gap Ack Blocks (out-of-order PSNs received by the receiver from a path), and duplicate PSN reports for each path based on an incoming SACK chunk. In other words, a source endpoint needs to construct a PSN-based SACK for each destination address (or path) based on an incoming SACK and the mapping between TSNs and PSNs for each path. Then the source endpoint handles these PSN-based SACKs (following the rules by which the standard SCTP handles TSN-based SACK) for each path and updates those congestion control parameters associated with each path. Thus, per path independent congestion control is guaranteed because the congestion control algorithms

at each path execute solely based on the PSNs transmission and reception at that path. Figure 39 illustrates the algorithm used to construct PSN-based SACK for each path.

```
From an incoming SACK chunk, get
Cumulative TSN Ack Point, Gap Ack Blocks, Duplicate TSN blocks,
For each destination address with data chunks outstanding {
Go through its send buffer list, do {
//assume it is in ascending order by PSNs
   if TSN <= Cumulative TSN Ack Point
     temporarily mark its corresponding PSN gap acked;
   if TSN falls in a Gap Ack Block
     temporarily mark its corresponding PSN gap acked;
   if TSN is a duplicate TSN
     mark its corresponding PSN duplicate PSN;
}
Go through its send buffer list again, do {
   Build PSN gap blocks for those PSNs temporarily marked gap acked;
   /*each block has two fields, Start PSN and End PSN */
   if Start PSN of the first PSN gap block == PSN of the first item in the
   buffer {
     Cumulative PSN Ack Point = End PSN of the first
     PSN gap block;
     Delete the first PSN gap block;
      }
      Build duplicate PSN blocks from those PSNs marked as duplicate
      }
}
```

Figure 39. An algorithm to construct PSN-based SACK

6.4.2.3 Adjustment of the Cumulative PSN Ack Point

As we discussed in section 6.3, when a standard SCTP association uses multiple paths and retransmission packets are transmitted through alternate path, the transmission procedure of the standard SCTP at one path may have to wait for the retransmission at alternate path. In IPCC-SCTP, we use the Cumulative PSN Ack Point adjustment mechanism to eliminate this drawback of the standard SCTP during retransmissions. In IPCC-SCTP, when a TSN is retransmitted through another path, it will be assigned a new PSN from the new path. The newly assigned PSN obsoletes the old PSN associated with the TSN at the original path. However, if we don't adjust the Cumulative PSN Ack Point when retransmissions are scheduled at another path, the obsolete PSN would still affect the congestion window growth at the original path since the update of congestion window for that path is dependent on the advancement of the Cumulative PSN Ack Point.

One possible solution is to treat those obsolete PSNs at each path as if they were acknowledged by the receiver, although those TSNs associated with them may still in flight at another path. Thus, retransmissions of TSNs through another path won't affect the future transmission and congestion control at the path those TSNs originally transmitted. In order to achieve this, we propose that the Cumulative PSN Ack Point adjustment (Figure 40) should be made for those paths that need retransmissions but retransmissions are scheduled at another path. The Cumulative PSN Ack Point adjustment has the same effect as acknowledging those retransmitted TSNs (and their corresponding PSNs) at the original path. Thus further transmission of PSNs at the original path won't be affected by those TSNs retransmitted through another path. The Cumulative PSN Ack Point adjustment is executed when a PSN-based SACK is built for

a destination address based on an incoming SACK chunk. For those TSNs retransmitted to the same destination address, no Cumulative PSN Ack Point adjustment is needed, and the same PSN for that TSN should be used again for retransmission.

```
/*Assume psn_adjustment_list stores those PSNs whose TSNs are retransmitted
to another destination address, an the psn_adjustment_list is in ascending order by
PSNs*/

if length(psn_adjustment_list) > 0
for(i = 0; i< length(psn_adjustment_list); i++) {
 psn = Head(psn_adjustment_list);
 if(psn == Cumulative PSN Ack Point + 1) {
 Cumulative PSN Ack Point = psn;
 Delete(psn_adjustment_list, HEAD);
 }
else
break;
}
```

Figure 40. Cumulative PSN Ack Point Adjustment Algorithm

6.5 Performance evaluation of IPCC-SCTP

We use simulations to confirm the effectiveness of our proposed IPCC-SCTP in supporting various multi-homing applications. The IPCC-SCTP implementation conforms to the rules on data transmission (section 6) and congestion control (section 7) of RFC 2960, modifications in [75] and our proposed independent per path congestion control framework. All simulations are conducted in the network simulator OPNET. In

our experiments, we created a network consisting two multi-homed SCTP endpoints: SCTP source and destination as shown in Figure 41. There are two disjoint paths between the SCTP source and destination: path 1 and path 2. Path 1 contains links of L1, L2 and L3. Path 2 contains links of L4, L5 and L6. Link L2 and L5 are bottleneck links of path 1 and path 2 respectively; their raw bandwidth may vary in different simulation runs. There is no propagation delay associated with access links, L1, L3, L4 and L6. For simplicity, propagation delay is only associated with bottleneck links L2 and L5. The path MTU at each path is 1500 bytes, and each data chunk has size of 1468 bytes. In section 6.5.1, we examine the retransmission performance of IPCC-SCTP under a scenario where one path is used for original data transmission and the other path is used for retransmission. In this case, two paths are simultaneously used for a very short period of time. In section 6.5.2, we examine the effectiveness of IPCC-SCTP under load sharing, in which two paths are simultaneously used for a long period of time.

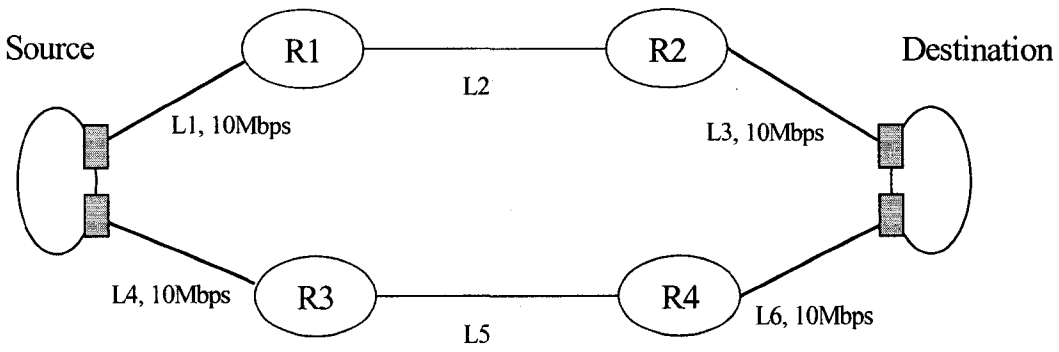


Figure 41. Simulation topology

6.5.1 Retransmission Performance

In this subsection, we examine the performance of IPCC-SCTP for one of the applications of the multi-homing feature recommended in RFC 2960: using alternate path for retransmission. This means that original data transmission uses the primary path, assuming it is path 1, and retransmission uses path 2. The bottleneck link bandwidth at each path is 1 Mbps. 2% packet level loss rate is emulated at path 1, which can be considered as packet drops due to buffer overflow or wireless error etc. The packet loss rate at path 2 varies between 0-10%. Random independent (Bernoulli) packet loss model is used at each path. 50000 SCTP DATA chunks (1468 bytes each) are to be transferred from the source to the destination. We use the file transfer time as the performance evaluation metric. Two scenarios are considered. In the first scenario, there is no propagation delay at L2 and L5, which corresponds to wireless links where propagation delay is usually ignorable compared to transmission delay. In the second scenario, one-way propagation delay at L2 and L5 is 50 ms, which emulates delay in WAN environment.

In our experiments, standard SCTP uses path 1 only, which means both transmission and retransmission use path 1. As is reported in [97], standard SCTP is infeasible in the case of using alternate path for retransmission and its performance sometimes worth than in the case of using only one path for both transmission and retransmission. This is the main reason we don't compare our proposed IPCC SCTP with standard SCTP in the case of using alternate path for retransmission, and we expect the IPCC SCTP always outperforms standard SCTP when it uses only one path. As shown in Figure 42, IPCC SCTP persistently outperforms standard SCTP if IPCC SCTP is given more resources,

i.e. path 2 is used for retransmission. In the first scenario, when there is no propagation delay at L2 and L5, IPCC SCTP spends about 2% less time for the file transfer than standard SCTP. In the second scenario, when there is 50ms one-way propagation delay at L2 and L5, IPCC SCTP spends about 8.6% less time for the file transfer than standard SCTP.

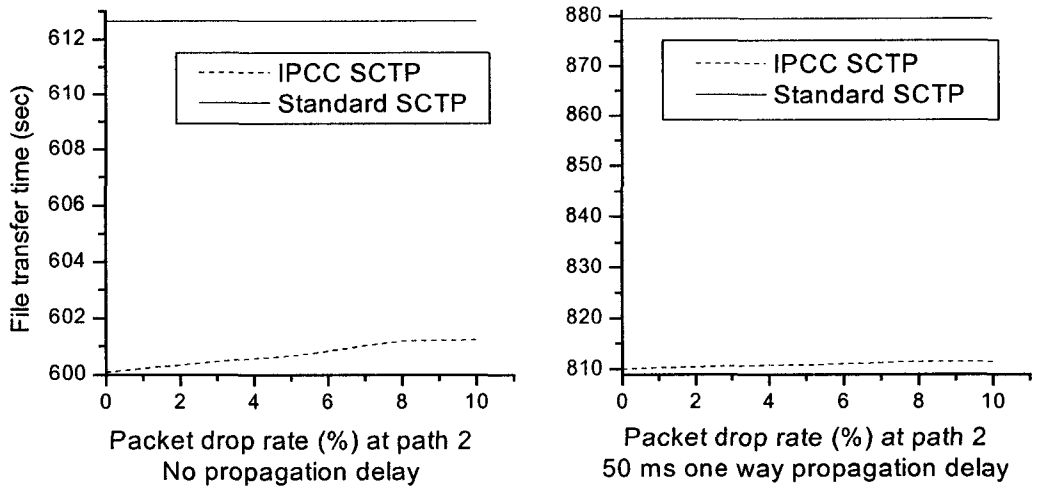


Figure 42. File transfer time, 2% packet drop at the primary path

The improved performance is due to the philosophy of independent per path congestion control when an IPCC-SCTP association simultaneously uses multiple paths. More specifically, adjusting the Cumulative PSN Ack Point at path 1 frees the congestion window of path 1 at times of fast retransmission (those retransmitted packets are retransmitted through path 2). In standard SCTP, the congestion control applies to the entire association. Thus further transmission of new data at path 1 would depend on the success of retransmissions if retransmissions are scheduled at path 2, and the

performance may be worse than in the case of using only one path for both transmission and retransmission if the transmission condition at path 2 is not good.

From Figure 42, we also observe there is a slight increase of file transfer time with the increase of packet drop rate at path 2 when IPCC SCTP is used. The number of packet losses at path 2 represent at most 0.2% of the total number of packets to be transferred (50000 in this experiment), this explains the reason that packet drop rate at path 2 has only minor effect upon the file transfer time when IPCC SCTP is used. As long as the packet drop rate at path 2 is less than 100%, some packets will eventually go through path 2, and we believe that IPCC SCTP will always outperforms standard SCTP. Therefore, by using the IPCC-SCTP, better file transfer performance is guaranteed no matter what is transmission condition at the alternate path except that the alternate path is totally unusable.

6.5.2 IPCC-SCTP in load sharing

In this section, we will examine the performance of IPCC-SCTP in load sharing. In this case, the IPCC-SCTP association will use path 1 and path 2 in Figure 41 simultaneously for data transmission. A simple load-sharing scheme is adopted in our simulation. We use congestion window of each destination address to regulate data transmission at each path. The IPCC-SCTP source starts data transmission at path 1. When the number of outstanding data at path 1 reaches the congestion window of destination address associated with path 1, it then starts transferring data through path 2. In this experiment, we only consider the effectiveness of IPCC-SCTP in support of load sharing. We do not compare IPCC-SCTP with standard SCTP for load sharing because standard SCTP cannot not support load sharing at all.

Two scenarios are studied in simulations. In the first scenario, there is no packet loss at each path. In the second scenario, there are packet losses at each path. Application bandwidth, which is calculated as the ratio of file size to file transfer time, is used as the evaluation metric. Application bandwidth reflects the application layer's view of the available bandwidth between source and destination endpoints. In order to create bandwidth/delay difference at two paths, the ratio of one-way propagation delay of bottleneck link L2 and L5 varies in our simulations. The one-way propagation delay of L2 is fixed at 25ms, and the one-way propagation delay of L5 varies in the range of 25-200ms. The goal is to create a relatively large range of bandwidth-delay product differences at two paths that IPCC SCTP will use for load-sharing application. The ratio of raw bandwidth at L2 and L5 is kept at 1:1 in simulations. The differences of raw bandwidth at L2 and L5 also contribute to the differences of bandwidth-delay product at path 1 and path 2; however, they should have the same effect upon the bandwidth-delay product as we varying the one-way propagation delays at each bottleneck link.

Figure 43 shows application bandwidth as a function of the sum of raw bandwidth at L2 and L5 in the first scenario, i.e. when there is no packet loss at each path. The curve labeled "IPCC-SCTP" represents application bandwidth when IPCC SCTP is used for load sharing. The curve labeled "Ideal" is the sum of application bandwidth at path 1 and path 2 when load sharing is disabled in IPCC SCTP. When load sharing is disabled in IPCC SCTP, IPCC SCTP degrades to standard SCTP. This curve actually represents the sum of application bandwidth that each path can provide to application layer. As shown in Figure 43, the aggregated application bandwidth achieved by IPCC-SCTP for load sharing is very close to the sum of the application bandwidth achieved at each path

separately under all the conditions we tested. When the one-way propagation delay ratio is 1:8, the round-trip-time at path 2 is around 8 times that of path 1; the IPCC-SCTP is still efficient on bandwidth aggregation in this case.

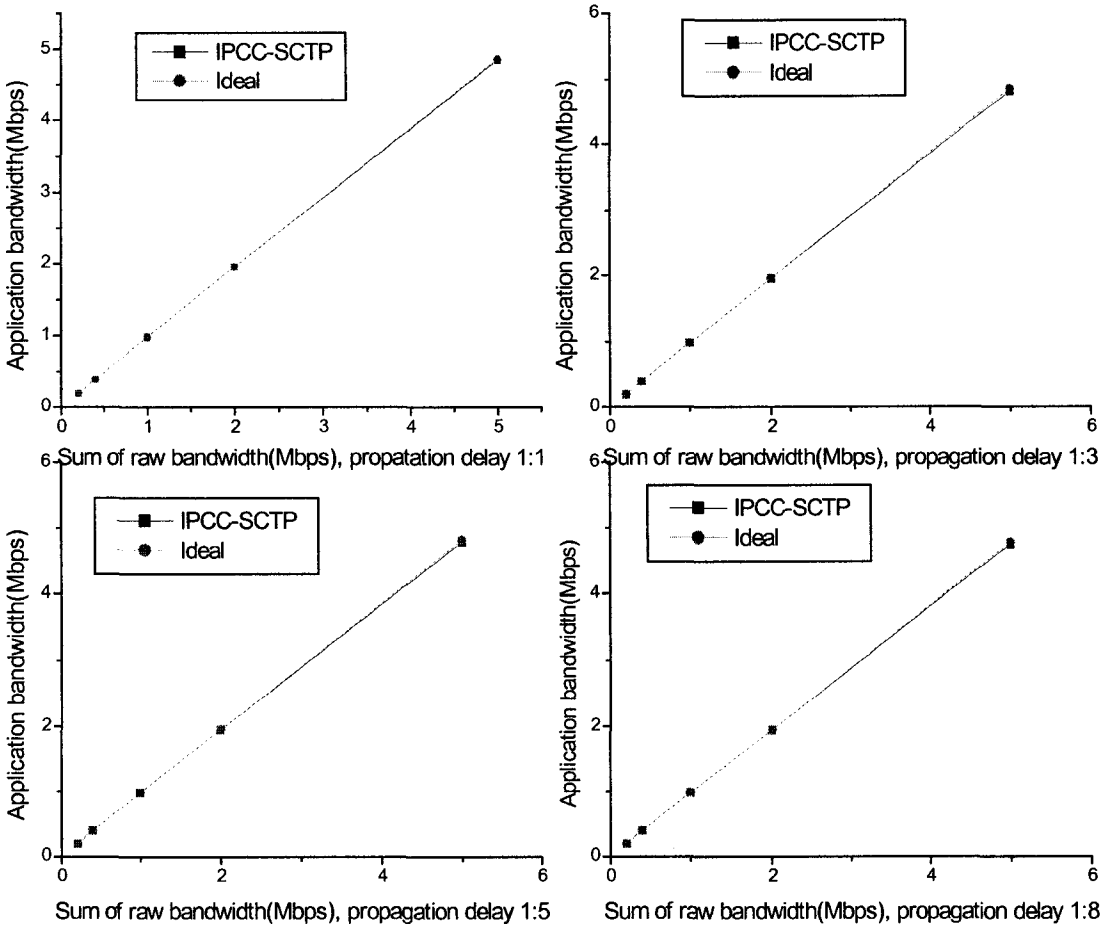


Figure 43. Application bandwidth vs. sum of raw bandwidth – No loss at each path

Figure 44 shows the application bandwidth as a function of the sum of raw bandwidth at L2 and L5 in the second scenario, i.e. when each path has 1% packet drop rate. In this case, the benefit of using IPCC-SCTP for load sharing becomes apparent. We

observed that application bandwidth of IPCC-SCTP is even higher than the ideal application bandwidth these two path can provide when the sum of raw bandwidth increases, and this gain in throughput becomes more apparent with the increase of propagation delay ratio at two paths. In other words, the larger the difference of bandwidth-delay product, the more gain that IPCC-SCTP can achieve in application bandwidth compared to the sum of ideal available bandwidth achieved by standard SCTP which uses one path only at a time. IPCC-SCTP does not steal bandwidth from anywhere; however, it uses bandwidth that is underused if standard SCTP is used instead. Qualitatively, standard SCTP halves its congestion window when there is packet loss detected by fast retransmission algorithm, and its congestion window cannot be reopened until those retransmitted packets are acknowledged by receiver. When the bandwidth-delay product is large, the halved congestion window and the delay in reopening the congestion window usually will cause under utilization of available bandwidth along the path. However, retransmission in IPCC-SCTP is allowed to schedule at another path, and the adjustment of the Cumulative PSN Ack Point reopens immediately the congestion window for a path if retransmission is scheduled at another path. This mechanism makes IPCC-SCTP efficient in utilizing available bandwidth available at all paths between source and destination.

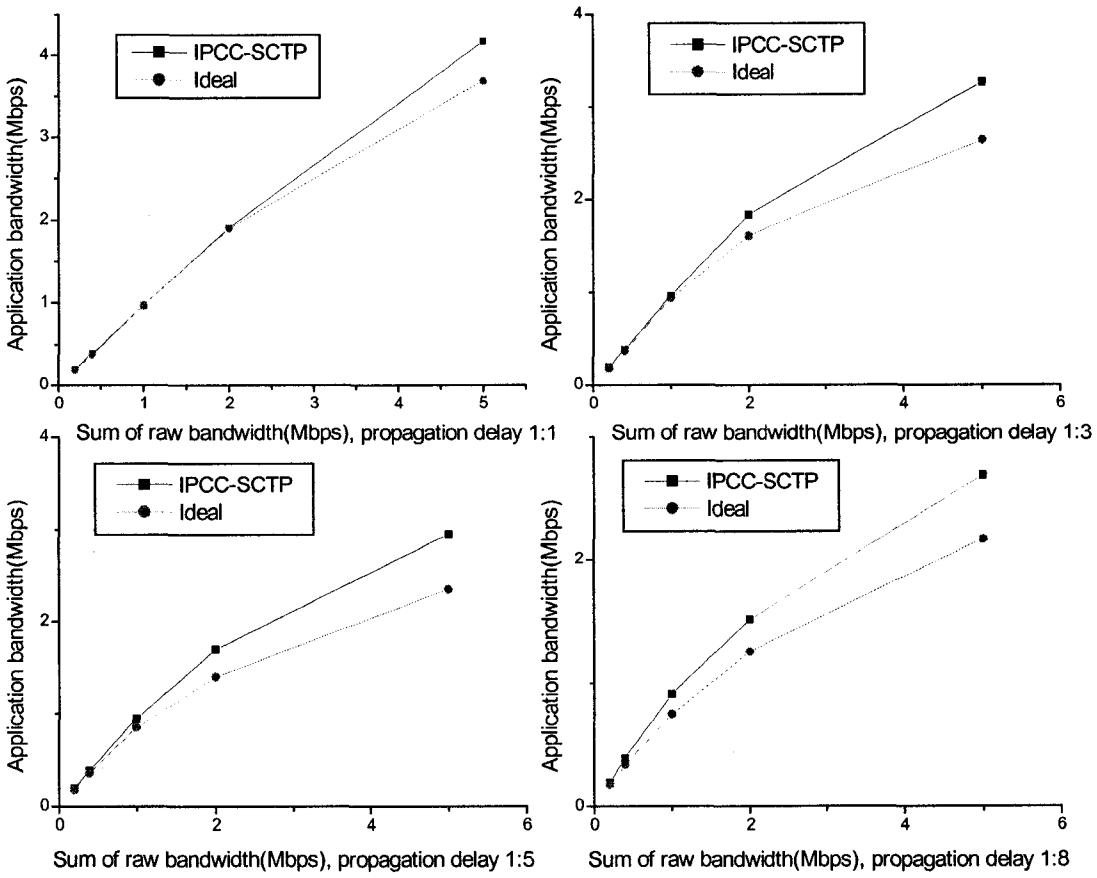


Figure 44. Application bandwidth vs. sum of raw bandwidth – 1% random loss at each path

6.6 Discussions

In this section, we discuss three issues that must be addressed in implementing the IPCC-SCTP. The first is the flow control. Flow control aims to prevent a slow receiver from being flooded by a fast sender. However, flow control will affect the total throughput performance when using IPCC-SCTP for load sharing or load balancing. As we know, either load sharing or load balancing will result in out-of-order arrivals of TSNs at the receiver, which will be very typical when the end-to-end delays at different

paths differ greatly. Since transport layer at the receiver cannot deliver out-of-order TSNs (assume there is only one stream) to application layer, the receiver buffer might be occupied mostly by those out-of-order TSNs. These buffered TSNs and those TSNs in flight will make the sender conclude that there is no room at the receiver to accommodate any more TSNs, thus the sender will be stalled although there is only small amount of TSNs in flight. In this case, the flow control is not to restrain the fast sender from flooding the slow receiver, but to make the concurrent transmission using multiple paths unfeasible. [86] proposed an out-of-order transmission scheme for TCP to avoid out-of-order arrivals, however, this scheme is vulnerable to packet losses and bandwidth variation. We think that a possible way to resolve this problem is for the receiver to advertise a large receiver window (at the order of the product of aggregated bandwidth along all the paths used and the largest retransmission timeout value among the paths) during the four-way handshake of an SCTP association. For large file transfer, the receiver can simply set its receive buffer the size of the file to be transferred to fully explore all available bandwidth.

Another approach to overcome the effect of flow control on the concurrent transmission using multiple paths is to use 2-level buffer structure at the receiver. The first level buffer, which stores in-sequence packets received by far, is used for flow control. When the first level buffer is full, a zero receiver window should be advertised to prevent the source from sending out new data. SCTP sources can assume there is enough buffer space at receivers as long as the advertised receiver side window is not zero. Out-of-order packets can be buffered at the second level buffer, thus the effect of flow control to the concurrent transmission using multiple paths is minimized. Certainly, large second

level buffer is needed when transferring large file using multiple paths that have large differences in bandwidth/delay. We use this approach in IPCC-SCTP.

The second issue is about the SACK generation at a receiver side. Due to the possible large amount of out-of-order arrivals when using multiple paths simultaneously for data transmission, delayed SACK mechanism seems to be useless in reducing the amount of SACK packets. Although small amount of reverse traffic may not be a big deal in wired network, but researchers have found this mechanism is useful in reducing collisions in shared medium such as IEEE 802.11 based wireless ad hoc networks [101]. A possible way to reduce the amount of SACK traffic is to use k-arrival SACK combined with a timer. K-arrival SACK means a SACK is not generated until there are k arrivals of TSNs no matter their orders. The timer guarantees that a SACK will be generated if less than k packets are received when the timer expires. Optimal value of k and the length of timer are for future study.

The last issue relates to the size of SACK chunk. Standard SCTP has no other restriction on the number of Gap ACK Blocks in a SACK chunk as long as the final SCTP packet is less than the current path MTU. Since it is possible that TSNs from different paths may arrive at a receiver in interleaved mode, the number of Gap ACK Blocks may be very large sometimes, especially, when the number of paths involved is large. Gap ACK Blocks reported in previous SACK become redundant information in subsequent SACKs. The standard SCTP requires this redundant information in subsequent SACKs. To be bandwidth efficient, SCTP receivers can report only those Gap ACK Blocks containing recently received TSNs (received after the previous SACK). Based on the previously received information in SACKs, senders still have a global view

of the reception status of TSNs at receivers after receiving a new SACK. However, the renege of SCTP receiver (dropping TSNs already Gap Acked) allowed by the standard SCTP makes this scheme unfeasible. To overcome the renege issue, we propose a minor modification to SCTP SACK Chunk, which reports those TSNs reneged by the receiver. Thus, Gap ACK Blocks of a SACK Chunk can only report those newly received out of order TSNs.

Chapter Seven

7 Conclusion, Future Work and Publications

In this chapter, we first summarize the major results of this thesis, and then discuss some of the directions of our future work; finally, we list publications on this work.

7.1 Conclusion

7.1.1 SCTP Performance in Ad Hoc Networks

In Chapter 3, we evaluate SCTP congestion control performance over IEEE 802.11 wireless LAN protocol through extensive simulation studies. We find the throughput of SCTP degrades when the number of hops increases. Increasing the window size does not help to increase the throughput. On the contrary, it amplifies the hidden node problem and the exposed node problem, worsening the throughput. The virtual carrier sense at MAC layer does help to improve throughput when the number of hops is small, but this is not true when the number of hops increases.

Interestingly, we found SCTP suffers unfairness problem in 802.11-based wireless ad hoc network. The unfairness problem has nothing to do with any flaw in SCTP design, but to do with the hidden node problem and exposed node problem at MAC layer.

Packet loss may cause SCTP timeout and degrade its performance. This is rather unfair if the packet loss is not caused by network congestion and the window size of SCTP is small. We call this phenomenon "small window syndrome". An algorithm is proposed to overcome this problem, and the simulation results show that the SCTP throughput is improved by 15-30%.

7.1.2 ECN-capable SCTP

In Chapter 4, we proposed algorithms to make SCTP ECN-capable in the context of lossy networks, such as ad hoc networks. Both analysis and simulation results are given to support the effectiveness of the proposed ECN mechanism for SCTP.

In order that the ECN-capable SCTP works efficiently in lossy networks, we found that the mechanism in ECN-capable TCP cannot simply followed by ECN-capable SCTP. Our proposed ECN-capable SCTP does not use the CWR chunk, which save precious bandwidth and makes ECN-capable SCTP robust over lossy links.

We found that the current response function of SCTP is not optimal, however, there exists an optimal value of the congestion window for an SCTP source in response to ECN messages. We also developed a simple and practical way to achieve this optimal value by carefully choosing the threshold (e.g. the minimum threshold of a RED queue) of queues that support ECN. The simulation results are in accordance with the analysis results. The simplified method guarantees maximum utilization of the bottleneck link when the network load is light.

Through theory analysis and real network simulations, we found the total goodput performance of SCTP associations is not sensitive to window reduction policies when the network load is heavy. This implies that fine-tuning SCTP or TCP's congestion window in response to congestion indications using complicated methods should not be advocated because the increase in performance (when network load is light) may not be worth the increase in complexity of the protocol.

A drawback of the RED method is also identified in this chapter, and an enhancement is proposed to extend the current marking or dropping policy by considering both the average queue length and the actual queue length.

7.1.3 Wireless-aware SCTP

In Chapter 5, we proposed a variant of SCTP, Wireless-aware SCTP. Wireless-aware SCTP is based on the ECN-capable SCTP discussed in Chapter 4. Wireless-aware SCTP is able to differentiate wireless losses from congestion losses by using the congestion coherence concept, and it can recovery wireless losses according to their nature (random or bursty losses).

Specifically, we introduced a Loss-Recovery mode into standard SCTP. In Loss-Recovery mode, packet losses due to wireless errors are recovered without applying congestion control algorithms. A modified fast retransmit algorithm is used to identify and recover random losses, which uses four missing reports to detect a packet loss and retransmit one packet per RTT when there are multiple packets need to be retransmitted at a time.

We use timeout events without congestion notifications to identify bursty losses. For bursty losses, a probing mechanism is developed to check the channel status first instead of blindly retransmission of lost packets. After the probing packet is acknowledged, the congestion window is fully restored in order to make full use of the good state of the wireless channel. It turns out that the probing scheme is able to help Wireless-aware SCTP achieve rather good throughput performance and be efficient in energy consumption. Simulations results and analysis support the effectiveness of the Wireless-aware SCTP in handling both i.i.d packet losses and bursty losses (small fading). We

believe that some link level intelligence must be used in order for the transport layer to perform well in the case of fast fading or when the number of hops between source and destination is very large.

7.1.4 Independent per Path Congestion Control

In Chapter 6, we investigated an important aspect of transport layer protocols in supporting the multi-homing feature, the underlying congestion control mechanism. Although data-striping algorithms may vary for different applications of the multi-homing feature, the underlying data delivery mechanism at the transport layer should be quite similar under various applications because they share a common feature, simultaneously using multiple paths.

We illustrated the limitations of standard SCTP when multiple paths are involved in an SCTP association. We found that the per association congestion control is the major reason of performance degradation when multiple paths are used by standard SCTP.

We proposed that the independent per path congestion control framework should be followed by a transport layer protocol in order to support multi-homing efficiently. By applying this framework to the standard SCTP, the new SCTP, IPCC-SCTP is invented, which can be used for various applications of the multi-homing feature with appropriate data-striping algorithms. Simulations are conducted to examine the effectiveness of our proposed IPCC-SCTP in different applications of the multi-homing feature. The IPCC-SCTP overcomes existing problems in the standard SCTP when the multi-homing feature is used. In addition, it is robust for a wide range of network conditions in supporting the multi-homing feature, e.g. load sharing.

7.2 Future work

In the end, we find that the effect of dynamic topology changes of ad hoc network upon the performance of transport protocols remains an open issue. The design of Wireless-aware SCTP does not consider those packet losses caused by network topology changes, such as route recomputation, network partition etc. However, we noticed that our proposed loss recovery algorithms for random wireless loss and bursty wireless losses are also feasible to handle topology related packet losses. Network topology changes can be either transient or long-lasting. Transient network topology, which means that network changes from one stable state to another stable state quickly, shall have very similar effect as random wireless errors upon transport protocols. Transport layer may even unaware the occurrence of transient network topology change, or it may detect one or two packet losses due to a quick network topology change. The modified fast retransmit algorithm would be able to detect packet losses due to transient network topology changes. As to the long-lasting topology, which means it takes long time for the network converges to another stable state, transport layer usually will suffer one or more timeout during such network topology changes. It seems that the probing scheme in Wireless-aware SCTP is a good solution for transport layer to survive in such a bad situation. One the one hand, the probing packet will trigger new route discovery at the routing layer (if on-demand routing protocols are used); one the other hand, it keeps the transport layer alive (with little energy consumption) when the network is almost unusable during the topology change.

Although it seems that Wireless-aware SCTP is a sound solution to both wireless errors and dynamic network topology changes in ad hoc network, we believe further

study is needed to see how Wireless-aware SCTP performs in ad hoc networks with dynamic topology changes, and how to optimize Wireless-aware SCTP in such environment.

We have shown qualitatively that Independent per Path Congestion Control is essential for SCTP to be able to efficiently use its multi-homing features. It is desirable to demonstrate quantitatively the performance of IPCC-SCTP when multiple paths are involved in an association. The difficulty is how to describe mathematically those operations, such as Cumulative PSN ACK Point Adjustment, fast retransmit when retransmission is scheduled at another path etc. The work will lead to better understanding of IPCC-SCTP and the popularity of the multi-homing feature of SCTP.

7.3 Publications

Below is a list of publications we have on this work.

- G. Ye, T. Saadawi, M. Lee, "On Explicit Congestion Notification for Stream Control Transmission Protocol in Lossy Networks," *Kluwer Cluster Computing, Special issue on Ad Hoc Networks* (to appear).
- G. Ye, T. Saadawi, M. Lee, "Improving Stream Control Transmission Protocol Performance over Lossy Links," *IEEE Journal on Selected Areas in Communications, ALL-IP WIRELESS NETWORKS,* May 2004.
- G. Ye, T. Saadawi, M. Lee, "IPCC-SCTP: An Enhancement to the Standard SCTP to Support Multi-homing Efficiently," *23rd IEEE International Performance Computing and Communications Conference,* Phoenix, AZ, April 2004.

- G. Ye, T. Saadawi, M. Lee, "Independent per Path Congestion Control for Reliable Data Transmission between Multi-homed Hosts," *IEEE Sarnoff Symposium*, Princeton, NJ, April 2004.
- G. Ye, T. Saadawi, M. Lee, "Using Explicit Congestion Notification in Stream Control Transmission Protocol," *The International Workshop on Mobile and Wireless Networks, 23rd IEEE International Conference on Distributed Computing Systems*, Providence, Rohde Island, May 2003.
- G. Ye, T. Saadawi, M. Lee, "Explicit Congestion Notification in Stream Control Transmission Protocol", *Proceeding of Collaborative Technology Alliances (CTA) Communications & Networks (C&N) Annual Symposium,* April 2003.
- G. Ye, T. Saadawi, M. Lee, "SCTP Congestion Control Performance in Wireless Multi-hop Networks," *MILCOM'02*, Anaheim, CA, October 2002.

Bibliography

[1] http://www.ietf.org/html.charters/manet-charter.html.

[2] R. Yavatkar, N. Bhagwat, "Improving end-to-end performance of TCP over mobile internetworks," *In Proc. of Workshop on Mobile Computing Systems and Applications*, December 1994.

[3] A. Bakre, B. Badrinath, "ITCP: Indirect TCP for mobile hosts," *In Proc. 15th Int'l Conf on Distributed Computing Systems*, May 1995.

[4] H. Balakrishnan, S. Seshan, RH. Katz, "Improving reliable transport and handoff performance in cellular wireless networks," *ACM Wireless Networks,* May 1995.

[5] K. Brown, S. Singh, "M-TCP: TCP for mobile cellular networks," *ACM Computer Communications Review (CCR),* vol. 27, No. 5, 1997.

[6] N.H. Vaidya, M. Mehta et al., "Delayed duplicate acknowledgements: a TCP-unaware approach to improve performance of TCP over wireless," *Technical Report 99-003*, Computer Science Dept., Texas A&M University, February 1999.

[7] T. Goff, J. Moronski, DS. Phatak, "Freeze-TCP: A true end-to-end tcp enhancement mechanism for mobile environment," *Proc. IEEE INFOCOM 2000*, March 2000.

[8] V. Tsaoussidis, I. Matta, "Open issues on TCP for mobile computing," *Wireless Communications and Mobile Computing,* vol. 2, February 2002.

[9] R. Stewart, Q. Xie et al., "Stream control transmission protocol," *IETF RFC 2960*, October 2000.

[10] Telecommunications Magazine, May 2001.

[11] R. Jain, K. Ramakrishnan, "Congestion Avoidance in Computer Networks with a Connectionless Network Layer, Part I: Concepts, Goals and Methodology," *Proc. Computer Networking Symposium*, Washington, D.C., pp. 134-143, April 11-13, 1988.

[12] J. Postel, "Transmission Control Protocol – DARPA Internet Program Protocol Specification," *IETF RFC 793*, September 1981.

[13] V. Cerf, R. Kahn, "A Protocol for Packet Network Intercommunication," *IEEE Transactions on Communications*, vol. 22, no. 5, pp 637-648, May 1974.

[14] J. Nagle, "Congestion Control in IP/TCP Internetworks," *IETF RFC 896*, January 1984.

[15] R. Jain, "A Timeout-based Congestion Control Scheme of Window Flow-controlled Networks," *IEEE Journal on Selected Areas in Communications SAC-4*, 7, October 1986.

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

[17] K.K. Ramakrishnan, R. Jain, "A Binary Feedback Scheme for Congestion Avoidance in Computer Networks with a Connectionless Network Layer," *Proc. ACM SIGCOMM'88*, 1988.

[18] V. Jacobson, "Congestion Avoidance and Control," *ACM SIGCOMM*, 1988.

[19] W. Stevens, "TCP/IP Illustrated, Volume 1," *Addison-Wesley*, 1994.

[20] V. Jacobson, "Modified TCP Congestion Avoidance Algorithm," *end2end-interest mailing list*, April 30, 1990. ftp://ftp.isi.edu/end2end/end2end-interest-1990.mail.

[21] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," *IETF RFC 2001*, January 1997.

[22] S. Floyed, "TCP and Successive Fast Retransmits. Technical report," October 1994. ftp://ftp..ee.lbl.gov/papers/fastretrans.ps.

[23] K. Fall, S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP," *Computer Communication Review*, 1996.

[24] S. Flody, T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm," *IETF Experimental RFC 2582*, April1999.

[25] J.C. Hoe, "Improving the Start-up Behavior of a Congestion Control Scheme for TCP," *ACM SIGCOMM*, August 1996.

[26] M. Mathis et al., "TCP Selective Acknowledgment Options," *IETF RFC 2018*, October 1996.

[27] V. Jacobson and R. Braden, "TCP Extension of Long-Delay Paths," *IETF RFC 1072*, October 1988.

[28] M. Mathis and J. Mahdavi, "Forward Acknowledgment: Refining TCP Congestion Control," *Proceedings of SIGCOMM'96*, August 1996.

[29] L. Brakmo and L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, October 1995.

[30] G. Hasegawa, M. Murata, and H. Miyahara, "Fairness and Stability of Congestion Control Mechanisms of TCP," *IEEE INFOCOM*, March 1999.

[31] J. Mo, R.J. La, V. Anantharam, and J. Walrand, "Analysis and Comparison of TCP Reno and Vegas," *IEEE INFOCOM*, March 1999.

[32] U. Hengartner, J. Bolliger, and T. Gross, "TCP Vegas Revisited," *IEEE INFOCOM*, March 2000.

[33] J. Bolliger, U. Hengartner, and T. Gross, "The Effectiveness of End-to-End Congestion Control Mechanisms," *Technical Report #313*, Department of Computer Science, ETH Zurich, February 1999.

[34] R. Stewart, Q. Xie, "Stream Control Transmission Protocol (SCTP): A Reference Guide," *Addison-Wesley*, 2002.

[35] M. Allman, V. Paxson et al., "TCP Congestion Control," *IETF RFC 2581*, April 1999.

[36] A. Jungmaier, M. Schopp et al., "Performance Evaluation of the Stream Control Transmission Protocol," *Proc. IEEE Conference on High Performance Switching and Routing*, June 2000.

[37] R. Brennan and T. Curran, "SCTP congestion control: Initial simulation studies", presented at the *17th Int. Teletraffic Congr.*, Salvador, Brazil, December 2001.

[38] A. Caro, J. Iyengar et al., "SCTP Failure Detection and Recovery: Smoother Transitions and Increased Throughput," *SCI 2002-6th World Multiconference on Systemics, Cybernetics, and Informatics*, Orlando, July 2002.

[39] J. Iyengar, A. Caro et al., "SCTP congestion window overgrowth during changeover," *SCI 2002 - 6th World Multiconference on Systemics, Cybernetics, and Informatics*, Orlando, July 2002.

[40] S. Xu, T. Saadawi. "Revealing TCP Incompatibility Problem in 802.11-based Wireless Multi-hop Networks," *GlobeCom'2001*, San Antonio, TX 2001

[41] S. Xu, T. Saadawi, "Revealing the Problems with 802.11 MAC Protocol in Multi-hop Wireless Ad Hoc Networks," *Journal of Computer Networks*. vol. 38, no. 4, March 2002.

[42] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *ANSI/IEEE std 802.11*, 1999.

[43] R. Stewart, L. Ong et al., "SCTP Implementors Guide" *IETF internet-draft, draft-ietf-tsvwg-sctpimpguide-02.txt*, November 2001.

[44] http://www.opnet.com.

[45] D. Johnson, D. Maltz, Y. Hu, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR), " *IETF internet-draft, draft-ietf-manet-dsr-09.txt*, April 2003.

[46] M. Allman, H. Balakrishnan et al., "Enhancing TCP's Loss Recovery Using Limitted Transmit" *IETF RFC 3042*, January 2001.

[47] M. Gerla, K. Tang et al., "TCP Performance in Wireless Multi-hop Networks," *Proceeding of IEEE WMCSA '99*, February 1999.

[48] S. Floyd, "TCP and explicit congestion notification," *ACM Computer Communication Review*, vol. 24, pp. 10-23, October 1994.

[49] K. Ramakrishnan, S. Floyed et al., "The Addition of Explicit Congestion Notification (ECN) to IP," *IETF RFC* 3168, September 2001.

[50] J.C. Majithia, et al., "Experiments in Congestion Control Techniques," *Proc. Int. Symp. Flow Control Computer Networks*, Versailles, France. February 1979.

[51] S. Floyd, V. Jacobson, "Random Early Detection gateways for Congestion Avoidance," *IEEE/ACM Transaction on Networking*, vol. 1 no.4, pp. 397-413, August 1993.

[52] B. Braden, et al., "Recommendations on the Queue Management and Congestion Avoidance in the Internet," *IETF RFC* 2309, April 1998.

[53] R.J. Gibbens and F.P. Kelly, "Resource Pricing and the Evolution of Congestion Control," *Automatica* 35, 1999.

[54] C. Liu, R. Jain, "Improving Explicit Congestion Notification with the Mark-Front Strategy," *Computer Networks*, vol. 35, no. 2-3, pp. 185-201, February 2001.

[55] N. Yin and M. G. Hluchyj, "Implication of dropping packets from the front of a queue," *7-th ITC*, Copenhagen, Denmark, October 1990.

[56] M. Kwon and S. Fahmy, "TCP Increase/Decrease Behavior with Explicit Congestion Notification (ECN)," *ICC*, April 2002.

[57] J. Liu, S. Singh, "ATCP: TCP for Mobile Ad Hoc Networks," *IEEE Journal on Selected Areas in Communications, Wireless Communications Series 10*, July 2001.

[58] W. Chen, Y. Hsiao et al., Syndrome: a light-weight approach to improving TCP performance in mobile wireless networks," *Wirel. Commun. Mob. Comput.* 2002; 2:37-57.

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

[60] A. Kumar, "Comparative performance analysis of versions of TCP in a local network with a lossy link," *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, pp. 485-498, 1998.

[61] M. Zorzi, R.R. Rao and L.B. Milstein, "On the accuracy of a first-order markov model for data transmission on fading channels," *in Proceedings of 1995 Fourth IEEE International Conference on Universal Personal Wireless Communications*, 1995.

[62] A. Kumar and J. Holzman, "Comparative performance analysis of versions of TCP in a local network with a lossy link, part II: Rayleigh fading mobile radio link," *Technical Report WINLAB-TR-133*, Rutgers University, 1996.

[63] M. Zorzi, A. Chockalingam and R. Rao, "Throughput analysis of TCP on channels with memory," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 7, pp. 1289-1300, 2000.

[64] A.A. Abouzeid, S. Roy and M. Azizoglu, "Comprehensive performance analysis of a TCP session over a wireless fading link with queuing," *IEEE Transactions on Wireless Communications*, 2(2):344-356, March 2003.

[65] G. Ye, T. Saadawi and M. Lee, "Using Explicit Congestion Notification in Stream Control Transmission Protocol in Lossy Networks," Proc. of ICDCS workshop (The International Workshop on Mobile and Wireless Networks), pp. 704-709, May 2003.

[66] P. Karn and C. Partridge, "Improving Round-Trip Time Estimates in Reliable Transport Protocols," *SIGCOMM*, 1987.

[67] V. Paxson and M. Allman, "Computing TCP's retransmission timer," *IETF RFC 2988*, November 2000.

[68] P. Sarolahti and Alexey. Kuznetsov, "Congestion Control in Linux TCP," *USENIX02*, 2002.

[68] R. Ludwig, M. Meyer, "The eifel algorithm for TCP," *IETF internet-draft, draft-ietf-tsvwg-tcp-eifel-alg-07.txt*, April 2003.

[69] R. Ludwig, M. Meyer, "The Eifel Detection Algorithm for TCP," *IETF RFC 3522*, April 2003.

[70] R. Ludwig, A. Gurtov, "The eifel response algorithm for TCP," *IETF internet-draft, draft-ietf-tsvwg-tcp-eifel-response-02.txt*, December 2002

[71] A. Gurtov, R. Ludwig, "Responding to spurious timeouts in TCP," *IEEE INFOCOM* March 2003.

[72] V. Jacobson, R. Braden, D. Borman, "TCP Extensions for High Performance," *IETF RFC 1323*, May 1992.

[73] D. Mitzel, "Overview of 2000 IAB Wireless Internet Working Workshop," *IETF RFC 3002*, December 2000.

[74] http://www.ietf.org/html.charters/rserpool-charter.html

[75] R. Stewart, L. Ong et al., "Stream Control Transmission Protocol (SCTP) Implementer's Guide," *IETF internet-draft, draft-ietf-tsvwg-sctpimpguide-08.txt*, February 2003.

[76] M. Allman, H. Kruse, and S. Ostermann, "An application-level solution to TCP's satellite inefficiencies," *in Proceedings of Workshop on Satellite-Based Information Services (WOSBIS)*, November 1996.

[77] H. Sivakumar, S. Bailey and R. Grossman, "Psockets: The case for application-level network striping for data intensive applications using high speed wide area networks," *in Proceedings of IEEE Supuercomputing (SC)*, Dallax, TX USA, November 2000.

[78] G. Malkin, "Multi-link Multi-node PPP Bundle Discovery Protocol," *IETF RFC 2701*, September 1999.

[79] K. Sklower, B. Lloyd et al., "The PPP Multilink Protocol (MP)," *IETF RFC 1990*, August 1996.

[80] K. Sklower, B. Lloyd et al., "The PPP Multilink Protocol (MP)," *IETF RFC 1717*, November 1994.

[81] R. Ogier, V. Ruenburg, and N. Shacham, "Distributed algorithms for computing shortest pairs of disjoint paths," *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 443-455, March 1993.

[82] I. Cidon, R. Rom, Y. Shavim, "Analysis of multi-path routing," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 885-896, Dec. 1999.

[83] A. Nasipuri and S.R. Das, "On-Demand Multipath Routing for Ad-Hoc Networks," *in Proceedings of the IEEE ICCN'99*, Boston, October 1999.

[84] W-H Liao et al., "A Multi-Path QoS Routing Protocol in a Wireless Mobile Ad Hoc Network," *Telecommunication Systems* (special issue of best papers from ICN2001), vol. 19, no. 3-4, pp. 329-347, 2002

[85] M. Gerla, S. Lee, "Split Multipath Routing with Maximally Disjoint Paths in Ad hoc Networks," *in Proceedings of ICC'01*, Helsinki, Finland, June 2001.

[86] D. S. Phatak, T. Goff, "A Novel Mechanism for Data Streaming Across Multiple IP Links for Improving Throughput and Reliability in Mobile Environments," *IEEE INFOCOM'02*, 2002.

[87] S. Floyd and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet," *IEEE/ACM Transactions on Networking*, August 1999.

[88] A. Akella, B. Maggs et al. "A Measurement-Based Analysis of Multi-homing," *ACM SIGCOMM*, 2003.

[89] C. Traw and J. Smith, "Striping within the Network Subsystem," *IEEE Network*, pages 22-29, 1995.

[90] H. Adiseshu, G. Parulkar and G. Varghese, "A Reliable and Scalable Striping Protocol," *ACM SIGCOMM*, 1996, pp. 131-141.

[91] V. Cardellini, M. Colajanni and P. S. Yu, "Dynamic Load Balancing on Web-Server Systems," *IEEE Internet Computing*, May/June 1999.

[92] M. Baentsch, L. Baum and G. Molter, "Enhancing the Web's Infrastructure: From Caching to Replication," *IEEE Internet Computing*, Mar-April 1997.

[93] T. T. Kwan, R. E. McGrath and D. A. Reed, "NCSA's World Wide Web Server: Design and Performance," *IEEE Computer*, November 1995.

[94] G. D. H. Hunt et al., "Network Dispatcher: A Connection Router for Scalable Internet Services," *Computer Networks and ISDN Systems*, 1998.

[95] A. Abd El Al, T. Saadawi and M. Lee, "Load Sharing in Stream Control Transmission Protocol," *IETF internet-draft, draft-ahmed-lssctp-00.txt*, May 2003.

[96] A. Abd El Al, T. Saadawi and M. Lee, "LS-SCTP: A Bandwidth Aggregation Technique for Stream Control Transmission Protocol," Computer Communications, Special Issue on Protocol Engineering for Wired and Wireless Networks, to appear in the 1st Quarter of 2004.

[97] A. Caro, P. D. Amer and R. Stewart, "Transport Layer Multi-homing for Fault Tolerance in FCS Networks," *Proceedings of the Collaborative Technology Allicances (CTA)*, 2003.

[98] A. Caro, P. D. Amer and R. Stewart, "Transport Layer Multi-homing for Fault Tolerance in FCS Networks," *MILCOM 2003*, October 2003.

[99] J. R. Iyengar, A. Caro et al., "Making SCTP More Robust to Changeover," *SPECTS 2003*, Montreal, Canada, July 2003.

[100] J. Iyengar, P. Amer et al., "Preventing SCTP Congestion Window Overgrowth During Changeover," *IETF internet draft, draft-iyengar-sctp-cacc-01.txt*, February 2004.

[101] S. Xu, T. Saadawi, "Evaluation for TCP with Delayed ACK Option in Wireless multi-hop Networks," *VTC Fall 2001*, October 2001.