Preferential Treatment of SCTP Streams in a Differentiated Services Environment

by

Jianping Zou

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

2007

UMI Number: 3288884

UMI Microform 3288884

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.

Prof. M. Ümit Uyar

Date — Chair of Examining Committee

Dean Mumtaz K. Kassir

Date — Executive Officer

Prof. M. Ümit Uyar, EE Department, CCNY

Prof. Myung. J. Lee, EE Department, CCNY

Prof. Ibrahim Habib, EE Department, CCNY

Prof. Leonid Roytman, EE Department, CCNY

Dr. Mariusz Fecko, Telcordia Technologies, Inc.

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

Preferential Treatment of SCTP Streams in a Differentiated Services Environment

by
Jianping Zou

Advisor: Prof. M. Ümit Uyar

With its new features such as multi-homing, multi-streaming, and enhanced securities, the Stream Control Transmission Protocol (SCTP) has became a promising candidate to join UDP and TCP as a general-purpose transport layer protocol. Multiple streams in an SCTP association provide an aggregation mechanism to accommodate heterogeneous objects, which belong to the same application but may require different types of QoS from the network. However, the current SCTP specification lacks an internal mechanism to support the preferential treatment among its streams, hindered by the fact that all streams share the flow and congestion control at the association level.

In this thesis, we introduce the concept of subflow and propose to modify the current SCTP such that the streams are grouped into several subflows according to their required QoS. Each subflow in the modified SCTP (SF-SCTP) implements its own flow and congestion control in order to avoid false sharing. With those modifications, the SF-SCTP behaves similarly as an aggregation of SCTP associations, which is too aggressive and can steal bandwidth from the competing network traffics. To make the SF-SCTP more

TCP-friendly and to reduce its aggressiveness, we further integrate the mechanism called Fractional Congestion Control (FCC) into the design of SF-SCTP.

Analytical models for the original SCTP, SF-SCTP without FCC and SF-SCTP with FCC are derived based on existing TCP throughput models. Our models improve the existing TCP models and are suitable for both Drop-Tail and RED queues, and for different packet sizes. In addition, our models can be used to predict the performance of a network designed to support differentiated services.

Performance evaluation and comparison of original SCTP, SF-SCTP without FCC and SF-SCTP with FCC are studied through a set of extensive simulation experiments. The results show that the proposed SF-SCTP design is able to support QoS among the SCTP streams and that false sharing is avoided. The results indicate SF-SCTP's significant benefits in improving the utilization of a bottleneck network. The results also reveal that, with FCC integrated, SF-SCTP is capable of taking advantages of differentiated services offered by a network, while maintaining fairness to other TCP-friendly traffics.

Acknowledgments

I must express my sincere gratitude and thanks to my advisor and committee chair, Prof. M. Ümit Uyar for his guidance, suggestions and encouragement through my Ph.D study and throughout the development of this thesis. Without his constant support, infinite patience and continuous encouragement, this thesis would not be possible. I always feel fortunate for having Prof. M. Ümit Uyar as my Ph.D advisor. He showed me the discipline for successful Ph.D research, taught me skills for professional writing, guided me how to conduct excellent research, and spent countless hours, including the sacrifice of many of his spare time, discussing my research work and provided much valuable knowledge.

I am deeply grateful to Dr. Mariusz Fecko for giving valuable insights into my work, for his constructive comments on the thesis, and for his contributions to our collaborative work. I would also like to thank Mr. Sunil Samtani for his support and contributions of ideas in this thesis.

I would like to express my special thanks to the members of the thesis committee for showing keen interest to the subject matter and reviewing the thesis, and for their support during my Ph.D studies.

Without the scholarships and the financial support from the CUNY and CCNY, the collaborative participation in the Communications & Networks Consortium sponsored by the U.S. Army Research Lab under the Collaborative Technology Alliance Program, Coop-

erative Agreement DAAD19-01-2-0011, and the National Science Foundation grant ECS-0421159, I would not be able to concentrate on this research.

Furthermore, I would like to thank my friends and colleagues Dr. İbrahim Hökelek , Dr. Samrat Batth and Dr. Yongwang Gao for their camaraderie during my graduate studies and for being absolutely awesome friends.

Last, but not least, I would like to thank my parents for their love and support of my studying abroad, my wife - Haoxu Ouyang - for her love, sacrifice and full-hearted support of my work.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The Stream Control Transmission Protocol (SCTP)(RFC 2960) [69, 68, 64, 65, 23] is a reliable, message-oriented transport protocol operating on top of potentially unreliable connectionless networks such as IP. Designed to overcome the shortcomings and limitations of TCP, SCTP is more suitable for applications requiring additional performance and reliability due to its new services such as multi-homing, multi-streaming, message boundary preservation, alleviated head-of-line blocking, and enhanced security features.

Originally, SCTP includes the multi-streaming feature to decouple the reliable transfer of user data from the strict ordered delivery. SCTP's support for multiple, logically independent message streams is an example of partial ordered delivery [26, 17], which is extremely useful for application with strict timing requirement [10, 50] by alleviating the head-of-line blocking problem. This feature reduces buffer requirements at the receiver, making

SCTP an attractive transport protocol for resource limited wireless handhold devices [7]. Multi-streaming enables SCTP to multiplex related, yet independent data streams over a single end-to-end association [63]. With multi-streaming, SCTP is equipped with an internal mechanism to support concurrent transmission of multiple objects [46]. For example, HTTP using SCTP can load a web page with multiple objects by a single SCTP association instead of several TCP connections.

Although multi-streaming lets SCTP accommodate heterogenous objects with different Quality of Service (QoS), the current SCTP does not take advantage of this feature. It is unaware of QoS and lacks an internal mechanism to support preferential treatment of SCTP streams. First, the SCTP association may find it hard to mark an outgoing packet's QoS byte since it could encapsulate messages from streams with different QoS requirements. Second, the current SCTP employs a TCP-like congestion control mechanism at the association level, which means that the streams carrying different objects are treated according to the same congestion state information. This shared congestion information could help improve the overall SCTP performance if the SCTP packets are treated the same by the network [9]. However, when packets with different QoS markings are processed differently by the network (in terms of packet loss rate, delay, etc.), the shared congestion information could result in the so-called *false sharing* [3]. This phenomenon occurs when flows sharing the congestion information do not actually share the same network bottleneck. The occurrence of false sharing could worsen the overall performance

of SCTP association. In the case where streams have different packet drop rates, the transport layer tries to moderate the performance for all streams; hence, the performance of the higher priority streams will be penalized due to the shared information from the lower priority streams. When streams experience different round trip times (RTTs), which may lead to unnecessary Fast Retransmits and time-outs, false sharing severely affects the SCTP performance in that all streams are penalized, regardless of their priorities.

In this thesis, we define the concept of *subflow* [59, 73, 74, 80, 77] and present the necessary modifications to the SCTP specification to support preferential treatment of SCTP streams. In this modified SCTP (called SF-SCTP), each subflow has its own flow and congestion control and consists of SCTP streams that require the same type of QoS from the network. Depending on the QoS required, an SCTP association may have one or more subflows, which serve as the independent transmission channels between an SCTP communication peer. Since the congestion state information is only shared by the SCTP streams requesting the same QoS, our design inherently avoids false sharing. With multiple subflows encapsulated, the SF-SCTP designed in this thesis is targeted to serve applications with different QoS requirements for its data. Different SCTP applications will each use a single association, which may be either original SCTP or SF-SCTP. The inclusion of multiple subflows in one SCTP association effectively eliminates the need for an SCTP application to open and maintain multiple connections when it needs to transmit several objects concurrently. In addition, using SF-SCTP, the application can now give

different priorities to the objects in the same communication session according to their relative importance. Since the design of SF-SCTP takes advantage of the multi-streaming feature of SCTP, it only adds a thin layer into the existing SCTP implementation and is backward compatible.

Due to multiple subflows, the introduced SF-SCTP behavior is similar to the aggregation of multiple parallel original SCTP associations. This behavior has certain advantages and disadvantages. In one aspect, like parallel TCP flows but without the overhead of maintaining multiple connections, SF-SCTP can be used to improve the utilization of a network with high bandwidth and delay product [60] or non-ignorable non-congestion loss [31, 32]. Single SCTP association, like single TCP flow, has its ineffectiveness in utilizing the available bandwidth of a high speed network [60, 47]. The Additive Increase (AIMD) congestion control principle of TCP/SCTP [44] prevent them to access the full bandwidth of a physical network. During the congestion avoidance phase, TCP/SCTP increases the congestion window *cwnd* linearly by approximately one MTU (Maximum Transmit Unit, 1500bytes for ethernet) per Round Trip Time (RTT), while decrease the *cwnd* dramatically to half after a Fast Retransmit event. In the case of a network with 10Gbps bandwidth, a single packet loss could take TCP/SCTP hours to recover to the available network throughput. Empirically, the grid and high performance computing communities have been using parallel TCP flows to improve the end-to-end network performance for applications requiring substantial amounts of network bandwidth [47, 4].

The SF-SCTP in this thesis presents an ideal candidate to be used in situations like this. For a network with plenty available bandwidth, the using of SF-SCTP with multiple subflows can greatly improve the utilization of such a network. Also SF-SCTP can be used by applications required to support parallel flows, such as bbcp [34] and GridFTP [5], resulting in significant throughput improvement.

In another aspect, the aggregate behavior of SF-SCTP with multiple subflows are not TCP-friendly. For a network which is undergoing heavy load and already in congestion, SF-SCTP boosts the throughput of its endpoint applications by unfairly stealing bandwidth from other competing network traffic. This practice, if adopted by most network users, may cause possible spiral of increasingly-aggressive congestion control behavior, leading to increasing network packet drop rate, which in turn will have a negative impact on the network performance [28], and may even lead to network-wide congestion collapse.

SF-SCTP's effectiveness in utilizing network bandwidth and aggressiveness towards other TCP-friendly lies in its aggregated congestion control behavior. Consider a SF-SCTP with N concurrent subflows, which will open its overall *cwnd* N times faster than a single SCTP. In SCTP, a single packet loss results only one subflow to cut its *cwnd* to half, thus, the overall *cwnd* of SF-SCTP is only dropped to $(2N-1)/(2N)$ of its previous value. The same packet loss will result an original SCTP association to reduce its *cwnd* by half. Compared with the original SCTP, SF-SCTP with multiple subflows has a much faster growing congestion window and is more resistant to packet losses. And as the number of

subflows increases, the SF-SCTP behavior gets more aggressive. Our SF-SCTP may thus perform better in terms of its throughput by unfairly harming other TCP-friendly traffic on the network.

To preserve SF-SCTP's effectiveness in utilizing network bandwidth while alleviating its over-aggressiveness, we introduce a mechanism called *Fractional Congestion Control* (FCC). The concept of FCC has been introduced in [32, 33] to preserve the effectiveness of aggregate TCP in an under-utilization network, while improving the fairness of aggregate TCP towards a single competing TCP flow in a network of over-utilization. Under the premise of FCC, during the congestion avoidance phase, the *cwnd* of each subflow increases at most one MTU per Φ RTTs, where Φ is the subflow's window growth parameter directly related to N and adjustable to make aggregate TCP fair to the original SCTP. The larger the Φ is, the slower a subflow's window growth rate becomes. To ensure fairness, Φ must be greater than N so that the overall *cwnd* of aggregate TCP increases less than $1/cwnd$ per RTT to offset aggregate TCP's greater resistance to packet losses. FCC reduces the aggressiveness of SF-SCTP by increasing the subflow congestion window slower than the original SCTP in congestion avoidance phase.

Starting from the existing TCP throughput models [55, 54, 71], we derive throughput models for the original SCTP and several different implementations of SF-SCTP. Our stochastic models effectively capture the congestion control behavior of SCTP and have been verified by simulations that, given the packet drop rate and RTT of a network, can

accurately predict the throughput of SCTP implementations. Our models improve the existing TCP models and are suitable for both Drop-Tail and RED queues, for different packet sizes, and also for various networks, such as wide area network, single bottleneck network, and Diff-Serv network [13, 52, 12, 8, 24]. In addition to the stochastic models which take packet drop rate and RTT as input parameters, for a single bottleneck network with Drop-Tail queue where queue delay plays a large factor, we also derive deterministic models which define the lower and upper boundaries for SCTP's throughput and packet drop rate.

Extensive simulation experiments have been performed to study the SF-SCTP's capability to support QoS in Diff-Serv network, the behavior of SF-SCTP under various network conditions, and the effect of fractional congestion control had on SF-SCTP. Our simulations confirm the capability of SF-SCTP to support preferential treatment among its subflows by avoiding the problem of false sharing and by dynamically adjusting its subflow throughput to network packet drop rate and Round Trip Time fluctuation. We especially studied in detail the behavior of SF-SCTP in single bottleneck networks. Those simulations reveal SF-SCTP aggressive nature in exploring network bandwidth. Compared to original SCTP, a SF-SCTP with n subflows and without FCC opens its congestion window n times faster, while reducing its congestion window by $1/n$ times slower. The simulations which compare SF-SCTP with various network background traffic further illustrate SF-SCTP's capability to seize the network bandwidth. Our simulations for SF-SCTP

with FCC show that, for a network under congestion, FCC effectively reduces SF-SCTP's aggressiveness, making it much more fair to other TCP-friendly traffics compared with SF-SCTP without FCC. Our simulation results even show that original SCTP, in a single bottleneck network with Drop-Tail queues, benefits from the presence of SF-SCTP with FCC because FCC alleviates the problem of Global Synchronization, reduces the network packet drop rate and improves the overall network usage. Finally, the conducted simulations verified the accuracy of our analytic models.

This work contributes to existing research in several ways:

- We present a novel design that combines the concept of subflows and fractional congestion control [59, 73, 74, 80, 77]. To the best of our knowledge, our work is the first to introduce subflow-capable SCTP and the first to introduce FCC into SCTP. In addition, we implemented SF-SCTP with FCC in ns-2 [70] platform and performed extensive simulation experiments to provide insights into performance gains for applications with diverse QoS requirements.
- Given the packet drop rate and RTT of a network, the stochastic models we derived in this thesis can accurately predict the throughput of different cases of SCTP implementations, including original SCTP and SF-SCTP with FCC integrated. Our models improve the existing TCP models [55, 54, 71] and are suitable for both Drop-Tail and RED queues, and for different packet sizes. Our models can be used to predict the performance of a Diff-Serv network designed to support differentiated services.

- Given the link delay, queue size and bandwidth of a bottleneck network, we have the deterministic analytic models to capture SF-SCTP's packet drop rate, throughput and *cwnd* behavior. Those models take queue delay into consideration and are suitable for both Local Area Network (LAN) and Wide Area Network (WAN) with a single bottleneck.
- This work extends the existing research on FCC [32, 33] and studies the effect of FCC on the throughput of SF-SCTP with different congestion window growth rates. We find that FCC, while effective in reducing SF-SCTP's aggressiveness, can also reduce its packet drop rate, a phenomenon not observed in previous research. For different *cwnd* increase rates, we study the effects of FCC on SF-SCTP with multiple subflows under various network configurations. We find that, similarly as observed in [32, 33], FCC can improve the fairness of parallel flows toward other network traffics. We also discover that FCC can alleviate the global synchronization problem of Drop-Tail queue [29]. Both the analytic models and simulation results point out FCC can improve the network utilization of a bottleneck network while keeping a relatively small packet drop rate.
- Our proposed SF-SCTP presents the grid and high speed computing community with a new transport layer protocol which is able to support parallel flows within one association and has FCC integrated if network bandwidth limit is reached.

The rest of this thesis is organized as follows. Chapter 2 summarizes the literature review,

giving an overview of necessary background material and some of the related work. The motivation and the design of SF-SCTP are presented in Chapter 3. In Chapter 4, we derive the stochastic models for four cases of SCTP and deterministic models for SF-SCTP behavior in a single bottleneck network. The results of various simulation experiments are discussed in Chapter 5. Chapter 6 concludes this thesis.

Chapter 2

Literature Review

2.1 Overview of SCTP

SCTP is a reliable, message-oriented transport protocol operating on top of potentially unreliable connectionless networks such as IP. Designed to overcome the shortcomings and limitations of TCP, SCTP is more suitable for applications that require additional performance and reliability due to its new services such as multi-homing, multi-streaming, message boundary preservation, alleviated head-of-line blocking, and enhanced security features.

SCTP's support of multi-homed devices, which can be accessed under several IP addresses, was motivated by the rigid timing and reliability requirement to transport telephony signaling messages over IP networks. SCTP has a built-in path failure detection and

recovery mechanism, which is essential for critical applications to provide uninterrupted service during hardware failures. An SCTP instance monitors all the possible transmission paths between an communication peer. If one path currently providing data transmission is found broken, the SCTP instance can safely switch the transmission to an alternative path within a short period of time without the loss of connectivity. Unlike TCP which requires additional add-on to support fault-tolerance [48], SCTP's multi-homing feature provides transparently support for fail-over and fault-tolerance in the event of network failure [22, 18, 45, 16, 58, 19].

Currently, SCTP uses multi-homing for redundancy purposes only. A primary path is selected as the default transmission path for SCTP packets. The secondary paths between the peer are used only to retransmit lost packets. To achieve a better utilization of the available network resources, it has been proposed in [22, 38, 43, 39, 1, 2] to extend the multi-homing capability of SCTP for concurrent data transmission over several network paths.

Using independent streams in an SCTP association, SCTP decouples the reliable data transfer from the strict order-of-transmission data delivery. The messages from the application layer will be assigned to different streams according to the requirements of an SCTP user. Since ordered delivery is only needed within each stream, if required, SCTP is able to reduce the unnecessary head-of-line blocking between different streams. Also with multi-streaming, SCTP is equipped with an internal mechanism to support transmission

of several objects concurrently. SCTP streams are independent parallel communication channels between SCTP hosts, providing logical data demarcation within an SCTP association [63]. SCTP, with its multi-streaming feature, presents applications a new transport layer protocol to choose when transmitting multiple types of data, avoiding the disadvantages of opening multiple connections, and the need to ensure reliable data transmission at the application layer. Several studies show that the use of SCTP benefits HTTP and multimedia applications [10, 50, 57, 51].

In order to avoid some security issues that TCP faces, such as SYN flooding and masquerade attacks, SCTP uses a enhanced four-way handshake procedure to setup a logical association between two communication endpoints. SCTP endpoints will not allocate resources until the identify of the other party is verified. SCTP also includes a Verification Tag in each of its packets, which effectively prevents SCTP from the "in the middle attacks".

2.2 Research on SCTP

Ref. [14] used simulation studies to explore the behavior of SCTP congestion control. The authors compared several possible variants of SCTP congestion control with that of TCP Reno. The authors identified three flaws in the current SCTP congestion control mechanism: efficient recovery relying on optional Gap Ack reported in SACK messages, vulnerable fast retransmit procedure, and inefficient fast retransmit procedure, and ac-

cordingly, provided suggested remedies for these flaws.

Refs. [40, 41, 42] studied the changeover issue of SCTP switching from a primary path to a secondly path. The authors found that, if following the current SCTP specification (RFC 2960), a path changeover will result in unnecessary retransmissions and aggressive growth (TCP-unfriendly) of the sender's congestion window, primary due to congestion control algorithm of the SCTP sender unaware of a changeover. An algorithm named the Split Fast Retransmit Changeover Aware Congestion Control (SFR-CACC) was introduced as the solution. Under SFR-CACC algorithm, the SCTP sender is able to maintain states on a per-destination basis when changeover happens. This avoids the unnecessary fast retransmissions, and therefore, curb the TCP-unfriendly congestion window growth at the sender.

In Ref. [21], the authors proposed a two-level threshold recovery mechanism to improve the throughput of a SCTP association. Their proposal, taking advantages of the fault tolerance capability of a SCTP association, decouples the failure detection and the recovery processes. They defined a parameter α to control when a SCTP transmission should be moved temporary to an alternate destination path. If the primary path does not recovery, by responding to the heartbeat signal, in a period defined by the second parameter β, the association will be moved permanent to the alternate path, otherwise, it will changeover back to the primary path. using this scheme, the authors found that SCTP achieved higher throughput for both short-term and long-term failures.

The research work in Refs. [38, 43, 39, 37] investigated the usage of SCTP multi-homing for Concurrent Multipath Transfer (CMT). To solve the reordering problem of multi-path transmission, the authors proposed three modifications to the current SCTP specification. The first algorithm, named Split Fast Retransmit algorithm, addresses the problem of unnecessary fast retransmission by introducing a virtual queue per destination within the sender's transmission queue. In this way, the sender can match the received SACKs with the corresponding virtual queue. Only the receipts of four SACKs reporting a TSN gap belonging to the same virtual queue will result in a fast retransmission. The second algorithm tracks the earliest outstanding TSN per destination to update the *cwnd* in order to prevent the sudden increase of *cwnd*. The last modification deals with the delayed SACK. The reordering problem causes SCTP receiver frequently not to delay SACKs. The modified delayed-SACK algorithm specifies a receiver's behavior on receipt of data, and also a sender's behavior when the missing report count for a TSN needs to be incremented. With those improvements, the modified SCTP solves the problem of false sharing and is able to take advantages of Concurrent Multipath Transfer to improve its throughput.

Refs. [1, 2] present LS-SCTP: an SCTP modified for performing transport-layer bandwidth aggregation. LS-SCTP separates the flow control from the congestion control, which are performed on an association and on a path basis, respectively. LS-SCTP lets packets that travel through diverse paths share flow control, although they may arrive out of order

at the receiver. Those unordered packets must be retained in the receiver buffer until all the intermediate packets arrive. When there are big RTT differences between paths, the unordered packets may consume a large percentage of the receiver buffer, thereby negatively affecting the throughput. The SF-SCTP presented here avoids this problem by migrating both the flow and congestion control into the subflow level.

The congestion control parameters in LS-SCTP are defined per transport-layer path for a multi-homed host, with each path corresponding to a sender-receiver interface pair. For example, if a sender node has interfaces A_1 and A_2, and receiver node interfaces B_1, B_2, and B_3, all five bound to a single SCTP association, then there are six possible transport-layer paths. LS-SCTP ignores the fact the multiple paths, despite their different interface addresses, may share the same physical link and thus have the same characteristic. In this case, keeping separate sets of congestion control parameters is wasteful and makes LS-SCTP aggressive. SF-SCTP, on the other hand, does not separate congestion control parameters based on multi-homing in the transport layer—the separation is performed solely due to the expected preferential treatment of streams in the network layer. Therefore, SF-SCTP does not maintain unnecessary congestion control state as long as the network is capable of preferential treatment of flows. Also, SF-SCTP reduces its aggressiveness through the fractional congestion control, whereas LS-SCTP has no mechanism to ensure fairness. For these reasons, SF-SCTP is more reliable and more fair to other network traffic.

2.3 Congestion Control

Several techniques to improve the performance of transport layer have been proposed [3, 9, 28]. In Ref. [9], the authors present an end-system architecture centered around a Congestion Manager (CM) that ensures proper congestion behavior and allows applications to easily adapt to network congestion. The CM is a middle layer responsible for congestion control of all TCP connections between TCP and IP layers. Independent multiple TCP connections cooperate rather than compete with each other. This framework integrates congestion management across all applications and transport protocols. The CM maintains congestion parameters and exposes an API to enable applications to learn about network characteristics, pass information to the CM, and schedule data transmissions. This special centralized congestion control scheme allows for multiple streams to share a network path while avoiding the false sharing problem.

False sharing can severely impair an end-to-end system congestion control which share congestion information across concurrent flows. Ref. [3] has investigated the origin and impact of false sharing on TCP performance. False sharing occurs in networks with QoS enhancements where a flow classifier segregates flows into different queues, or in networks with path diversity where different flows to the same destination are routed differently [3]. Their simulation results show that faster applications are heavily penalized as a result of false sharing. Without a separate congestion control for each subflow in SF-SCTP, the benefits of a network supporting QoS might be forfeited due to transport layer's

unawareness of QoS and inability to avoiding false sharing.

In Ref. [31] aggregate TCP flows are used to improve network utilization. To address the unfairness of competing aggregate TCP flows towards a single TCP flow, the concept of fractional congestion control is introduced in Refs. [32, 33]. In TCP, the aggressiveness of parallel flows is due to a fast congestion window growth rate and a larger resistance to loss. Towards a single flow, parallel flows compete unfairly because they open their congestion windows n (the number of flows) times faster. When there are packet losses, parallel flows absorb the loss over the affected flows, while allowing the rest of the flows to continue normal operation. For high speed networks where packet losses result exclusively in Fast Retransmits, parallel flows could achieve a fair throughput towards a single flow given that each one of the parallel flows increase its congestion window $\frac{1}{n^2}$ time slower [32, 33]. Since our SF-SCTP design behaves similar to an aggregated group of original SCTP flows, we could also apply the fractional congestion control mechanism to SF-SCTP to improve its fairness towards the original SCTP.

In [31], aggregate TCP flows are used to improve network utilization. To address the unfairness of competing aggregate TCP flows towards a single TCP flow, the concept of fractional congestion control is introduced in Refs. [32, 33]. Compared to a single TCP flow, the aggressiveness of multiple parallel TCP flows is due to a faster congestion window growth rate and a larger resistance to loss. Parallel TCP flows compete unfairly towards a single flow in that they open their congestion windows n (the number of flows)

times faster. And when there are packet losses, parallel flows absorb the losses over the affected flows, while allowing the rest of the flows to continue normal operation. For high speed networks where packet losses are exclusively responded by Fast Retransmits, parallel TCP flows could achieve a fair throughput towards a single flow given that each one of the parallel flows increase its congestion window $\frac{1}{n^2}$ times slower [32, 33]. Since the behavior of our SF-SCTP design is similar to an aggregated group of original SCTP flows, we could also apply the fractional congestion control mechanism to SF-SCTP to improve its fairness towards the original SCTP. The analytic models and simulation results of SF-SCTP with fractional congestion control have been studied in Refs. [80, 75, 76].

2.4 Analytic Models

The accuracy of the analytic model depends on how well the models capture SCTP's congestion control mechanism. SCTP's congestion control is inherited from TCP [61, 62] with a few differences [72, 14]. First, when updating its *cwnd*, SCTP depends on the amount of data acknowledged rather than the number of acknowledgements received. This mechanism ensures that SCTP increases its *cwnd* one MTU (Maximum Transmit Unit, 1500bytes for ethernet) per RTT, regardless of whether or not the acknowledge is delayed or not. Second, SCTP allows the number of bytes outstanding to be less than one MTU greater than the current *cwnd* value, while in TCP, the outstanding bytes are strictly constrained by the *cwnd* size. Thus, compared to TCP, SCTP is allowed to send

one more packet each RTT. Third, when *cwnd* equals *ssthresh*, SCTP regards itself still at the slow start phase, allowing SCTP to recover the *cwnd* faster than TCP after a Time-out. Finally, SCTP use Selective Acknowledgement (SACK) mechanism, similar to SACK TCP [30, 27], to report missing data chunks, however, four rather than three SACKs reporting the same date chunks missing are required in SCTP to trigger a Fast Retransmit, which gives SCTP more tolerance to unordered packet delivery.

The SF-SCTP analytic models presented in this thesis benefit from the works of throughput analytic models of TCP [49, 53, 55, 54, 71] and SCTP [72, 73, 74, 80]. Refs. [49, 53] model only the behavior of TCP's Fast Retransmit mechanism. The model of Ref. [55, 54] captures not only the effect of TCP's Fast Retransmit but also the time-out mechanism. Refs. [55, 54] models the congestion behavior of the TCP-Reno sender under the following assumptions: the RTT is independent of the window size, which implies that the necessary time to send a window of packets to the network is negligible compared to the RTT; and packet losses within a round are correlated while the losses are independent among rounds. That model is shown to be accurate over a range of packet drop rates by the experiments carried on the Internet. The work of Ref. [71] extends Refs. [55, 54] to Diff-Serv environment, which could be a two-drop precedence network [25, 35], or a network with three-drop precedences [36]. Considering the minor differences betweens SCTP and TCP congestion control, Refs. [72, 73, 74, 80] obtain the analytic models for various SCTP implementations by extending TCP's model [55, 54].

Chapter 3

SF-SCTP Design

3.1 Design Motivation and Alternatives

HTTP and Multimedia applications are in great need of the concurrent transmission of several types of data, which may have different QoS requirements. Traditionally, HTTP applications could open multiple TCP connections to provide logical demarcation of data by separating them into different connections according to their types; however, this approach is very inefficient. It is a waste of application resources to start up, maintain and tear down those TCP connections, considering some of them may have very short life periods. Multimedia applications, on the other hand, typically build their own transport layers on top of UDP because UDP minimizes the delay for real-time traffic. However, UDP requires multimedia applications to encapsulate different types of data into pack-

ets, and add flow and congestion control for reliability. These additional requirements significantly increase the complexity of multimedia applications.

SCTP, with its multi-streaming feature, presents applications a new transport layer protocol to choose when transmitting multiple types of data, avoiding the disadvantages of opening multiple connections, and the need to ensure reliable data transmission at the application layer. SCTP streams are independent parallel communication channels between SCTP hosts, providing logical data demarcation within an SCTP association [63]. Several studies show that the use of SCTP benefits HTTP and multimedia applications [10, 50, 57]. Multi-streaming is also useful for applications that multiplex related, yet independent data streams (e.g., voice, text, video) subject to different QoS requirements [23].

Despite the advantages of SCTP to accommodate heterogenous data streams, it is not designed to handle different QoS and is unable to support preferential treatment among its streams. Under the current specification of RFC 2960 for SCTP [69], data chunks from various streams requesting different QoS could be bundled together to form an SCTP packet, which complicates SCTP application's decision to mark an outgoing packet with a particular DSCP value. Also currently in SCTP, all streams share one flow and congestion control at the association level, which may result in false sharing assuming packets marked with different DSCPs experience different drop rates and RTTs from the network. With the congestion information falsely shared, the overall performance

of SCTP will be penalized and the streams with higher priorities are unable to perform better than the other streams.

The original SCTP design also falls short of satisfying the highly dynamic nature of military and disaster recovery networks. For example, in an application transmitting MPEG-4 video streaming over SCTP, the stream carrying intra-coded frames may require a higher level of QoS than the ones carrying predicted frames. Similarly, if one stream carries a voice conversation, while another stream transfers on-going location and situation awareness information, applications would benefit from SCTP placing higher priority on the conversation and from the network offering preferential treatment to this higher-priority stream.

Without necessary modification, SCTP cannot naturally support applications that uses a single SCTP association to carry content differentiated by QoS, e.g., reliable data with no timeliness requirements such as text pages, billing and accounting information, or setup signaling, as well as unreliable data with stringent timeliness constraints, e.g., voice or video [67].

In a Diff-Serv network, flows with different QoS requirements are marked with different Differentiated Services Codepoints (DSCPs). When a flow has a DSCP with high priority, the application requires that this flow be treated in a special way both at the send/receive endpoints and in the network. In this thesis, we consider only the network effect. Each DSCP value represents an underlying service definition and has significant effects on the

transmission of IP datagram [6]:

- Because both hosts and routers can consider the value of the DSCP field of an IP datagram when choosing an appropriate path to get the datagram to its destination, the datagrams with different DSCPs may travel through separate paths with different characteristics.
- Even when the hosts and routers choose the same path for all datagrams, DSCPs can cause datagrams to be handled differently. For example, a host or router might choose to give preferential queuing on network output queues to datagrams that have requested that delay be minimized. Similarly, an overloaded router that is forced to discard packets might still forward packets that have requested high reliability.

Considering the effects of the DSCP marking, the current SCTP is unable to support QoS because user messages with various QoS requirements from different streams could be bundled together and streams are falsely sharing a congestion control at the association level. Different approaches, which could make SCTP capable of supporting QoS, are discussed and compared in this section.

In our design, we group the streams with the same QoS requirements into one subflow, where each subflow inside an SCTP association has its own set of congestion parameters: *cwnd* (congestion window), *ssthresh* (slow start threshold), *SRTT* (Smoothed Round Trip Time), *RTO* (Retransmission time-out), *TSN* (Transmission Sequence Number), etc. Normally, the subflows may have different congestion state at a given time because

the subflows will be handled differently due to either diverse routing or QoS queueing. Our design avoids false sharing and is able to support QoS inside an SCTP association.

Similar to the CM approach [9], one may propose to build a middle layer between SCTP and IP to provide preferential treatment among SCTP streams, without changing the SCTP on-the-wire bits or opening new SCTP associations. This approach, however, is plagued by the problem of false sharing since it is unable to solve the problem of unnecessary Fast Retransmits, nor can it provide an accurate RTT estimate.

To avoid the false sharing among streams of an association, one may consider designing a mechanism to establish *multiple associations* to accommodate streams with different DSCPs. To support different DSCPs in a communication between two endpoints, the user application will establish and maintain multiple associations. The upper layer could either add a new association only when it is necessary, or set up all associations if the application has the knowledge of what service classes the association will support. The user layer also has the choice as to when to close a useless association in order to save sender and receiver resources.

The benefit of the multi-association scheme is that it does not require the change of the SCTP on-the-wire bits. However, this scheme is unable to adapt to the dynamic requirements of the SCTP applications, which could change the DSCP of a stream in the middle of a connection. Also, breaking a single SCTP association into multiple associations may cause possible spiral of increasingly-aggressive congestion control behavior that leads to

increasing packet drop rates, which in turn will have a negative impact on the network performance. Compared with the multiple-association scheme, our proposed SF-SCTP has the following benefits: (1) it can avoid the resource-wasting setup and teardown procedures for multiple associations; (2) it can provide load balancing between paths and subflows with further extension; (3) it includes fractional congestion control that alleviates the over-aggressiveness of SF-SCTP when there is limited bandwidth available; (4) it facilitates the management of a single association for scheduling transmissions or load balancing among streams from the application layer's point of view.

3.2 Modification of SCTP for Multiple Subflows

Samtani et al. [59] initially proposed that, to overcome the problem of false sharing, only the streams requesting the same QoS could share the congestion state. Using this proposal as a starting point, we split the streams inside an association into multiple subflows and each subflow is consisted of streams with the same QoS requirement. To make it easy for an SCTP application to determine an outgoing packet's QoS byte, only messages from the streams belonging to the same subflow could be bundled to form an SCTP packet. Each subflow is independent and has its own transmission and receiving queue to avoid possible interactions with other subflows.

To avoid false sharing, modifications of the flow and congestion control parts of the current SCTP specifications are necessary. When packets from separate subflows were marked

with different DSCPs, they may use different paths or QoS queues and arrive out of order at the receiver endpoint since different paths or separate processing queues are associated with different RTTs and packet loss rates. Out of order arrival of packets will cause the SCTP receiver to initiate immediate Selective Acknowledgement (SACK) transmission to report the TSN gaps to the sender. Even though the packets may not be lost, when four SACKs reporting the same TSN gap are received, the SCTP sender has to respond with an unnecessary Fast Retransmit. If the retransmitted packet was not received in time or lost again, the sender will have no choice except to initiate a time-out recovery. Those unnecessary Fast Retransmits and time-outs will significantly reduce SCTP's performance if the flow and congestion control of current SCTP is not modified.

To overcome these problems, starting from the approach of Ref. [59] , we modify SCTP in the way that each subflow implements its own flow and congestion control [73, 74, 80, 75, 79, 78, 76]. A separate set of congestion parameters are defined for each subflow: $cwnd_i$, $ssthresh_i$, $SRTT_i$, RTO_i, FSN_i (subflow Transmission Sequence Number), etc., where i means that this parameter belongs to the i^{th} subflow. FSN has replaced the TSN, which is originally designed for the whole association, to ensure per-subflow transmission reliability.

The introduction of subflow and individual flow and congestion control modifications require us to change the formats of INIT, INIT_ACK, ECHO, ECHO_ACK, SACK control chunks and the SCTP data chunk. A new SCTP data chunk (Fig. 3.1) is created by

adding two new parameters to the standard SCTP data chunk. The first parameter is a one-byte subflow identifier (FID), which identifies the subflow that this data chunk belongs to. The second parameter is a 3-byte subflow Sequence Number (FSN), which is a monotonically increasing sequence number for the data chunks sent within an subflow. Since the reliability of flow control will be achieved by using the FSN, the function of the TSN is to correlate between data chunks received from different subflows. Thus we choose to change its name from Transmission Sequence Number to Association Sequence Number (ASN). ASN is reserved for the purpose of backward compatibility and possible future extensions to support cooperation among subflows.

Similar to the data chunk, the new SACK format (Fig. 3.2) also includes the FID and FSN parameters. The modification of the SACK format is necessary since, if not modified, it could lead to inaccurate calculation of RTT for subflows, which will consequently lead to unnecessary time-outs. After the modification, the receiver generates SACKs in per subflow basis and includes only the state information belonging to the subflow specified by FID. The state information that SACK conveys to the sender contains 1) the advance of ASN and FSN which acknowledges the arrivals of new packets, 2) the FSN gaps or duplicate FSNs which may indicate the loss of packets, and 3) the available Advertised Flow Receiver Window Size (a_fwnd) when this SACK was generated.

For the sake of backward compatibility, a new optional parameter can be added to the INIT and INIT-ACK chunk, namely, the preferential subflow-capable parameter shown

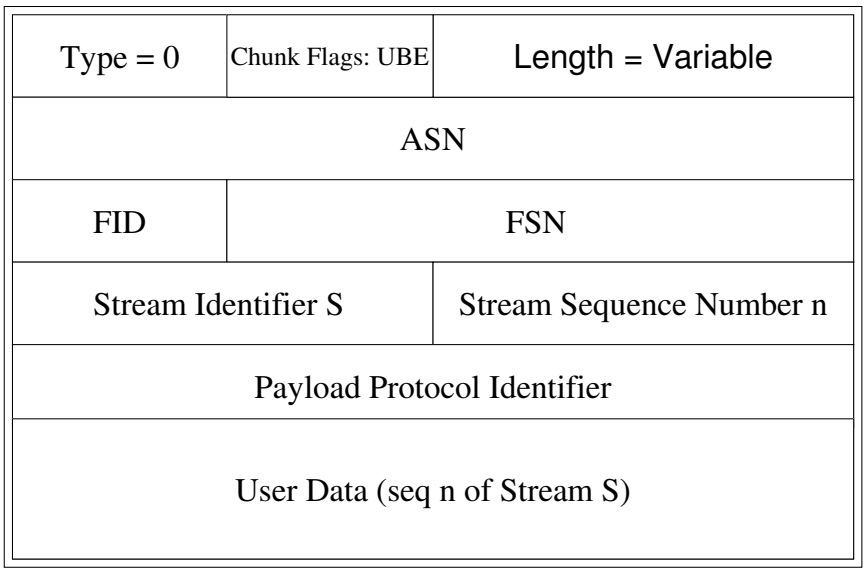


Figure 3.1: Modified SCTP Data Chunk.

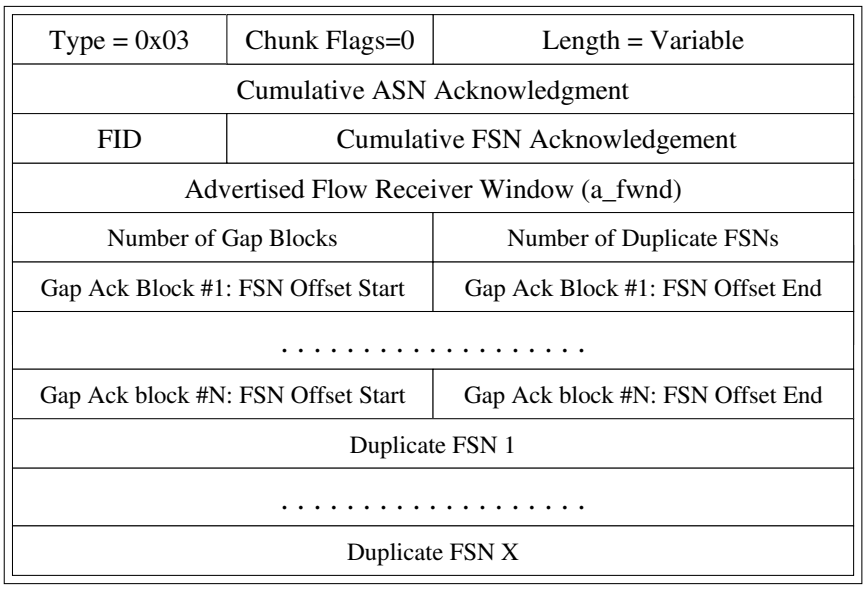


Figure 3.2: Modifications to the data and acknowledgment chunks.

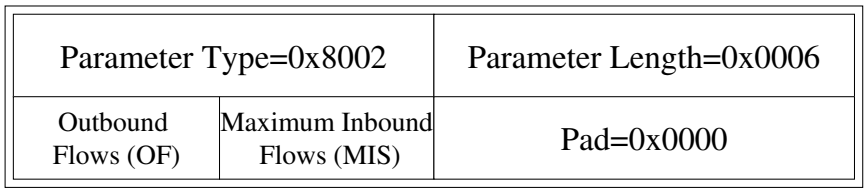


Figure 3.3: Subflow capable parameter.

in Fig. 3.3. The new parameter contains only one byte of data, which notifies the communication peer that this maximum number of subflows can be supported. During the association setup phase, the subflow-capable parameter is exchanged between the communication peer. If one of the peer cannot support even one subflow, or the INIT-ACK chunk does not return subflow-capable parameter (which indicate that the receiver is unable to support SF-SCTP), the sender will start the communication as an original SCTP association.

During the association setup, another two new optional parameters are defined to be exchanged in Cookie Echo and Cookie ACK. The first one identifies the initial Flow Receiver Window Size to be allocated for each subflow. The second one indicates the initial FSN to be used by the sender for each subflow. Figs. 3.4 and 3.5 show the initial Advertised Flow Window Size and the initial FSN for each subflow, respectively. The order of these two parameters in the Cookie Echo and Cookie ACK should be from the lowest to the highest FID, i.e., FID_0 to FID_NSF-1.

In addition, the following modifications to the current SCTP are included in our design:

Parameter Type	Parameter Length=0x0008
FID	Advertised Flow Windows Size (a_fwnd)

Figure 3.4: Initial advertised flow receiver window size.

Parameter Type	Parameter Length=0x0008
FID	Initial Flow Sequence Number (FSN)

Figure 3.5: Initial flow sequence number.

- To satisfy applications that dynamically change a stream's QoS, our SF-SCTP can migrate a stream to a different subflow if the DSCP marking of the stream is changed. SF-SCTP is also able to add/delete a subflow in the case where a new type of service arises or an existing one disappears.
- To save control information, SF-SCTP combines subflows with the similar congestion state parameters.
- Because the Cookie Echo/ACK chunks are exchanged before subflows are established, the modified SCTP will not allow any data chunks to be transmitted inside the Cookie Echo/ACK chunks. The only exception may be the data chunks with the same DSCP as the Cookie Echo/ACK chunks.

3.3 Design Architectural of SF-SCTP

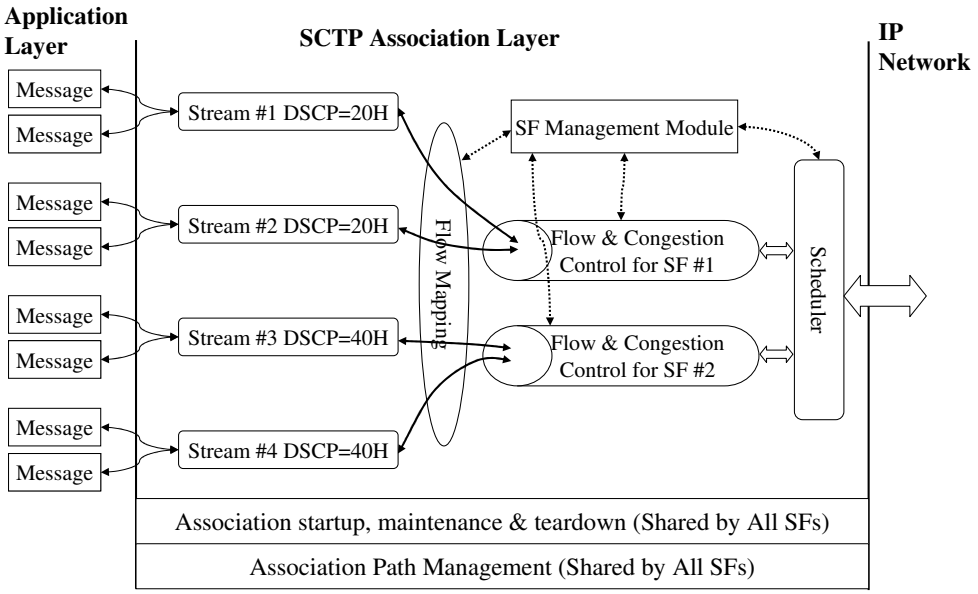


Figure 3.6: Architectural of SF-SCTP Design (SF = subflow).

Fig. 3.6 is an instance of the SF-SCTP implementation and provides the architectural overview for our design. The SF-SCTP association in Fig. 3.6 has four streams of messages. Streams 1 and 2 require the same QoS with a DSCP value of 20H, while streams 3 and 4 require the service DSCP of 40H. Based the DSCP markings, the flow mapping module of SF-SCTP assigns streams 1 and 2 into subflow 1 and streams 3 and 4 into subflow 2. The separate flow and congestion control parameters for both subflows are stored in the *Subflow Management Module* (SMM).

SMM is the central collector and processor of subflows' newest flow and congestion con-

trol state information. The independent flow and congestion control for each subflow in SF-SCTP are supported by the separated state information in SMM. In SMM's internal table, a separate set of flow and congestion parameters are maintained and updated timely for each subflow, such as Congestion Window ($cwnd_i$), Advertised Receiver Window Size (a_rwnd_i), Slow Start Threshold ($ssthresh_i$), Smoothed Round Trip Time ($SRTT_i$), Retransmission time-out (RTO_i), Subflow Identifier (FID_i), Subflow Transmission Sequence Number (FSN_i), etc., where i means that this parameter belongs to the i^{th} subflow.

Packets from all subflows are regulated by a scheduler before injecting into the network. We implement the First-Come First-Served (FCFS) scheduling algorithm for our SF-SCTP. In our study, the SCTP communication peers are assumed to have enough resources, such as a large enough buffer and a fast working network interface card to prevent any packet withholding in the scheduler's queue. We also assume that the only constrain for data transmission are resulted from the limited resource of the network. Therefore, it is safe to claim that, in our study, the FCFS scheduler should not have any effect on the throughput of subflows.

The elegance of our SF-SCTP design lies in the inclusion of SMM and scheduler for regulating data transmission. This design provides ample flexibilities to extend the SF-SCTP to support additional functionalities. For example, by using Priority Scheduling or Multilevel Queue Scheduling instead of FCFS in the scheduler and modifying SMM

appropriately, SF-SCTP can provide internal QoS for subflows. Also proper cooperation among subflows can be realized by enhancing the SMM and scheduler implementation, which will not be discussed in detail here but could be the possible areas of future research.

3.4 Fractional Congestion Control in SF-SCTP

Initially [73, 74], we design SF-SCTP in such a way that each subflow implements the same congestion control as in the original SCTP (RFC 2960). As a result, compared to the original SCTP, which logically has only one flow, SF-SCTP with multiple subflows combined as a whole has a much faster growing congestion window and is more resistant to packet losses.

Compared to the original SCTP, SF-SCTP has a much faster Congestion Window (*cwnd*) growth rate and a greater resistance to packet losses [32, 33]. During the congestion avoidance period, the *cwnd* of each subflow increases by $1 * MTU$ per RTT if there is no packet loss, which implies that the overall *cwnd* of the whole association will increase by $n * MTU$ per RTT (n refers to the number of subflows). When there is a packet loss detected by an SCTP association, the *cwnd* of the original SCTP is reduced to half after a Fast Retransmit. However, for SF-SCTP, its overall *cwnd* will only reduce to $(2n-1)/2n$ of its previous value since only one subflow is affected by this packet loss. SF-SCTP benefits from the fact that it can absorb packet losses to partial subflows, while allowing the unaffected subflows to continue normal operation. However, in a Drop-Tail network,

this benefit of SF-SCTP's greater resistance to packet losses is largely forfeited due to the global synchronization problem of Drop-Tail queue [29].

We introduce fractional congestion control [32, 33] to our SF-SCTP design to reduce its aggressiveness during congestion avoidance period. Under the rule of fractional congestion control, the congestion window of each subflow increases only $1*MTU$ per Φ RTTs, where Φ is the subflow's window growth parameter directly related to n and adjustable to make SF-SCTP fair to the original SCTP. The larger Φ is, the slower a subflow's window growth rate becomes. To ensure fairness, Φ must be greater than n so that the overall congestion window of SF-SCTP increases by less than $1 * MTU$ per RTT to offset SF-SCTP's greater resistance to packet losses. Similar to the work in Ref. [32, 33], fractional congestion control in our work affects only the congestion avoidance period of a subflow and does not apply to a subflow's slow start. An analytic model for SF-SCTP with fractional congestion control which can determine the value of Φ is discussed in chapter 4.

3.5 Endpoint Behavior for SF-SCTP

Fig. 3.7 illustrates the outbound message passage in the SF-SCTP. Messages from the user application are first checked with the PMTU (i.e, Path MTU) and are segmented if they are larger than PMTU. Then, the message data chunks with Stream ID (SID) and Stream Sequence Number (SSN) are placed into the flow decision queue (appeared as Flow Mapping Module in Fig. 3.7), whose function is to separate user messages (streams) into

different subflows according to their QoS requirements. After a data chunk is mapped into an subflow, the data chunk header fields of FID, FSN and ASN are added. Data chunks in the same subflow transmission queue can be bundled together to form an SCTP packet. Without losing generality, in this study, control chunks can be bundled with data chunks to form an SCTP packet.

Packets from all subflows are regulated by a scheduler before getting injected into the network. We implement a First-Come First-Served (FCFS) Scheduling algorithm. It is assumed in this work that the only constraint for data transmission result from the limited resource of the network. The SCTP end hosts are expected to have enough resources such as a large buffer and a fast working network interface card to prevent any packet from being held up in the scheduler's queue. Following those assumptions, it is safe to claim that this FCFS scheduler should not have any noticeable effect on the SF-SCTP's throughput.

At the receiver endpoint, the inbound message passage is shown in Fig. 3.8. The received SCTP packets are first unbundled to separate SCTP control chunks from data chunks. Next, the data chunks are placed into different subflow-receiving buffers according to the FID in the data chunk header. If data chunks are found segmented, they will stay at the subflow receiving buffer until reassembled. After that, if there are no missing data chunks or gaps detected among the received data chunks, they will be moved to user application layer. Otherwise, the affected data chunks from the streams which have gaps in the SSN are buffered until the missing data are delivered, while data chunks from other streams

can still be delivered to the upper layer applications.

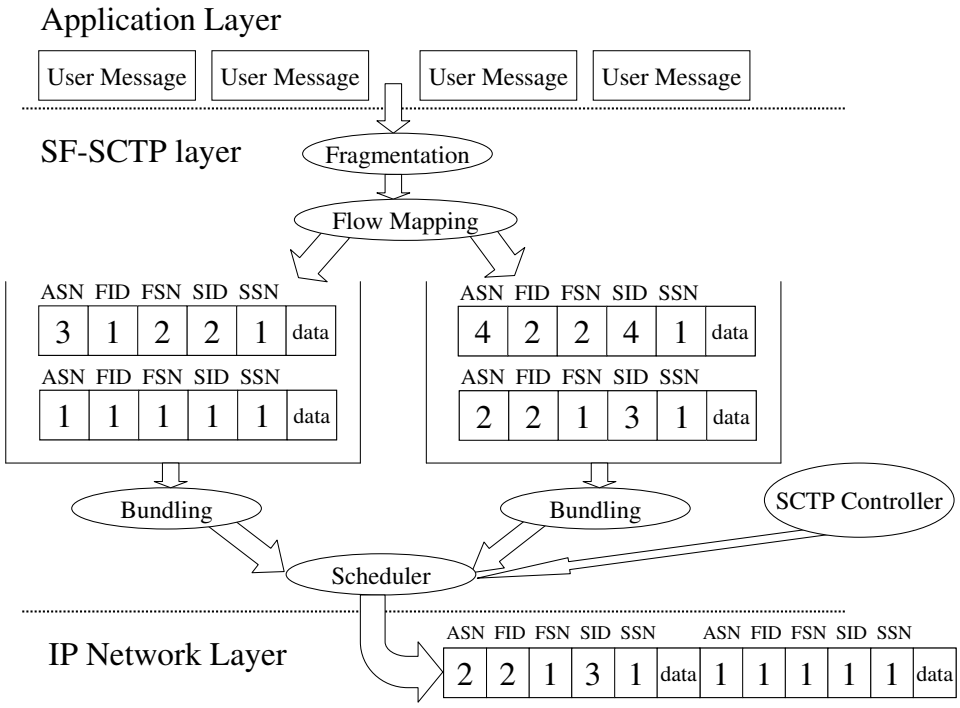


Figure 3.7: Message outbound procedure at the sender.

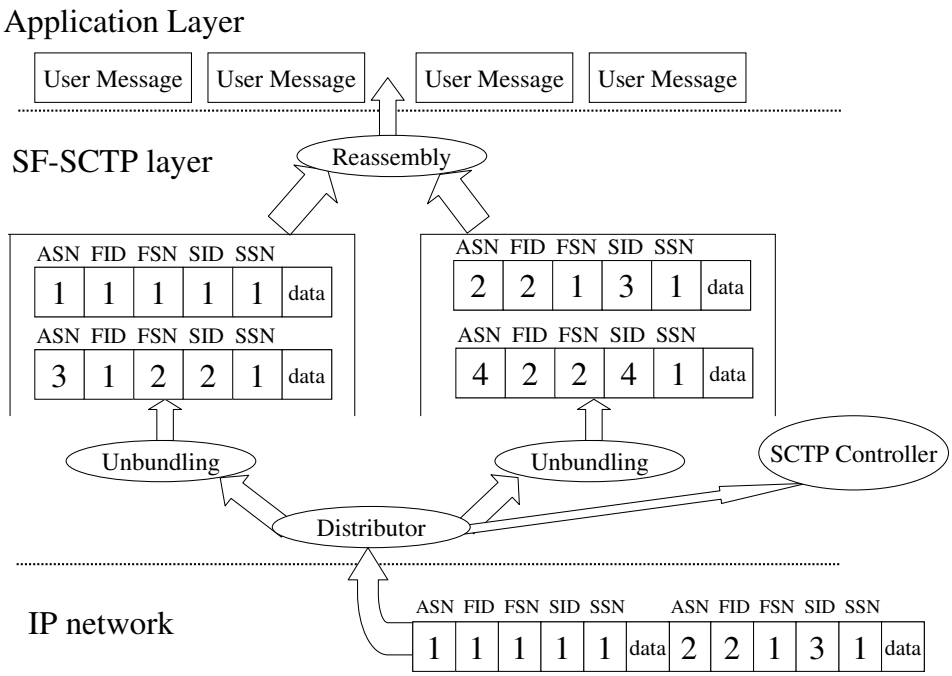


Figure 3.8: Message inbound procedure at the receiver.

Chapter 4

Analytic Model for SF-SCTP

In this chapter, we derive SCTP's analytic models in a simplified Diff-Serv network where each DSCP is mapped to a unique packet drop rate. In this Diff-Serv network, the flows with different DSCPs shall experience different packet drop rates after reaching steady state since the network treats each flow with different priorities. It is also assumed in this simplified Diff-Serv network that all the flows between a sender and receiver shall travel through the same physical path so that they will have the same RTT. The analytic models described in this section benefit but differ from the existing TCP models in Refs. [55, 71]. Our models (1) reflect the improvements that SCTP brings to the congestion control mechanism of TCP-Reno; (2) can account for FCC because they are more generic than the TCP models; (3) can model the SF-SCTP with all subflows sharing congestion control. The last feature is significant in that it demonstrates the need for a designated congestion

control mechanism for each subflow.

Specifically, we present the analytic models for the following cases of SCTP:

Original SCTP (Case 1): The current SCTP specified in RFC 2960 which has no subflow modification and is unaware of QoS such that the outgoing traffic is of type BE.

SF-SCTP-shared (Case 2): SCTP with subflow modification but all subflows share one congestion control at the association level; a Weighted Round-Robin (WRR) scheduler is used to apportion the available bandwidth among the subflows.

SF-SCTP-noFCC (Case 3): SF-SCTP where each subflow implements its own congestion control of the original SCTP style; fractional congestion control is not implemented.

SF-SCTP-FCC (Case 4): SF-SCTP where each subflow implements its own fractional congestion control.

When SF-SCTP has only one subflow with outgoing traffic of type BE, it is equivalent to the original SCTP (Case 1) in terms of throughput. Also when the subflows from SCTP of Case 3 and 4 have the same *cwnd* growth rate of one MTU per RTT ($\Phi = 1$ for Case 4 SCTP), Case 3 SCTP becomes a special instance of Case 4 SCTP. Therefore, the throughput model for SF-SCTP-FCC should be derived first since, from which, Case 1 and Case 3 SCTP's throughput expressions can be obtained. The analytic model for SF-SCTP-shared are derived in the separate Section 4.2.

We use another section to study the performance of SF-SCTP under a single bottleneck

network as depicted in Fig. 4.3. For this single bottleneck network where the queue delay may not negligible, we derived a deterministic lower and upper model for the throughput of SF-SCTP.

The accuracy of an SCTP throughput analytic model depends on how well it captures the congestion control behavior of SCTP. For example, in the original SCTP, an association increases its congestion window exponentially during the slow start period, or linearly (one MTU per RTT) during the congestion avoidance period. On the detection of packet loss, the congestion window of original SCTP is reduced to half for a Fast Retransmit event, or to one MTU for a time-out event. When the packet losses are exclusively responded by Fast Retransmits, the congestion behavior of original SCTP can be simplified in a way as shown in Fig. 4.1(a), which captures only the congestion avoidance periods and Fast Retransmit events of original SCTP. The time-out events, which bring the original SCTP back to slow start phase from congestion avoidance phase, are negligible since they are assumed to be rare. After the simplification, the lifetime of an original SCTP association can be considered as series of Fast Retransmit Periods, each of which is period between the time when the last Fast Retransmit ended and the time when the next Fast Retransmit starts.

Fig. 4.1(a) shows the j^{th} Fast Retransmit Period of the original SCTP. The *cwnd* value of this particular period starts with a value of $\frac{W_{j-1}}{2}$, which is half of the *cwnd* value at the end of the $(j-1)^{th}$ Fast Retransmit Period, and then increases step by step to W_j after

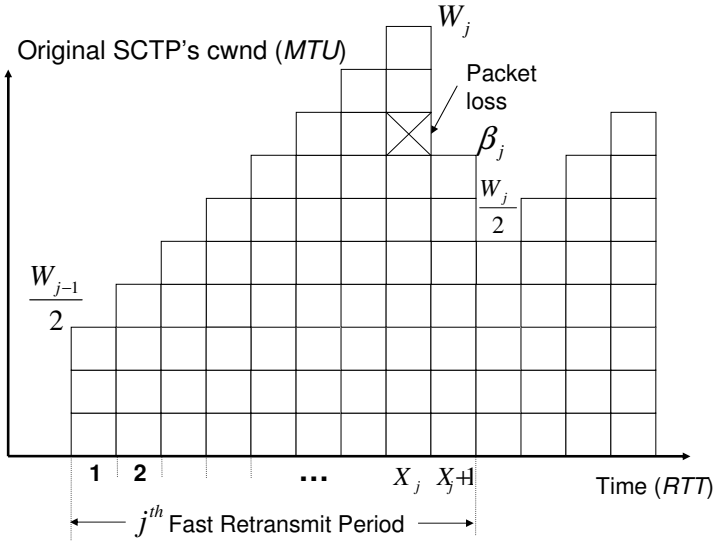
X_j number of RTTs. A packet loss in the X_j RTT marks the coming end of the j^{th} Fast Retransmit Period, which will last for another RTT, the $X_j + 1$ RTT, since it takes the sender one more RTT to detect the packet loss. A Fast Retransmit event is triggered to recover the lost packet and the congestion window is reduced to half of W_j. The variable, β_j, is used to represent the number of bytes sent out in the last RTT of $X_j + 1$.

Similar to Fig. 4.1(a), the simplified congestion control behavior of the i^{th} subflow (SF_i) in SF-SCTP-FCC has been shown in Fig. 4.1(b). Because of fractional congestion control, the *cwnd* growth rate of SF_i is slower than the original SCTP and only increases by one MTU per Φ RTTs. In Fig. 4.1(b), $\Phi = 2$ is chosen and hence $cwnd_i$ of SF_i is shown to increase by one MTU in two RTTs. When a packet belonging to SF_i is lost and then recovered by Fast Retransmit, $cwnd_i$ of SF_i will reduce to half as the original SCTP does. A superscript i has been added to the variables, W_{j-1}, W_j, X_j and β_j in Fig. 4.1(b) to convey that these variables belong to SF_i.

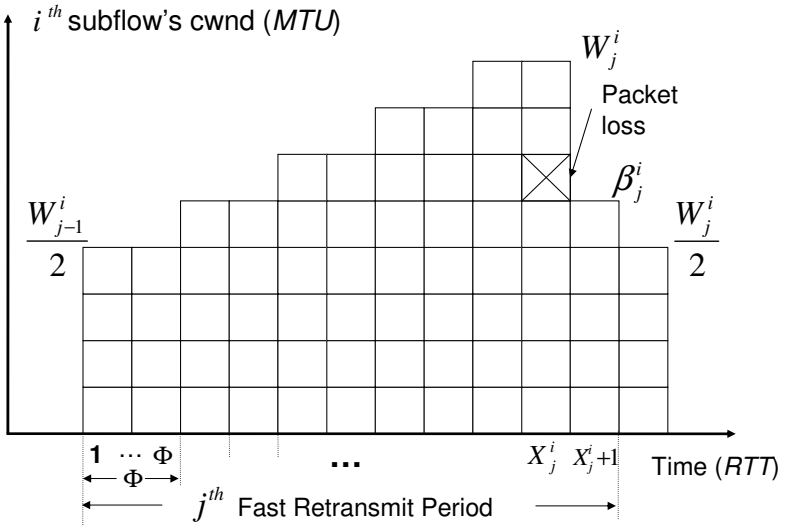
As presented in Refs. [55, 71], after a flow reaches steady state, for a given p (probability that an outgoing packet being dropped), the flow throughput $B(p)$ (bytes/sec) can be expressed as

$$\frac{B(p)}{K} = \frac{E[N]}{E[A]} \tag{4.1}$$

where K is the packet size, $E[A]$ is the mean time duration for a Fast Retransmit Period, $E[N]$ is the mean for the number of packets sent during $E[A]$. Here a flow refers to either an original SCTP association, or a subflow in SF-SCTP.



(a) Original SCTP.



(b) One subflow of SF-SCTP (with FCC).

Figure 4.1: Congestion window behavior of the original SCTP and one subflow in SF-SCTP-FCC

Eq. (4.1) captures the behavior of congestion avoidance and Fast Retransmit events only. A more general throughput equation, which takes time-out events into consideration, has been given in Refs. [55, 71] as

$$\begin{aligned} \frac{B(p)}{K} &= \frac{E[N] + Q * E[R]}{E[A] + Q * E[T^{TO}]} \qquad \text{for general case} \\ \frac{B(p)}{K} &\approx \frac{E[N]}{E[A] + Q * E[T^{TO}]} \qquad \text{for } E[N] \gg E[R],\ Q \ll 1 \end{aligned} \tag{4.2}$$

where Q is the probability that a packet loss is responded by a time-out rather than a Fast Retransmit, $E[R]$ is the number of packets resent during time-out, and $E[T^{TO}]$ is the mean for the time-out duration.

In this work, we consider only the effect of Fast Retransmits. Therefore, instead of Eq. (4.2), Eq. (4.1) has been used to derive the throughput expressions for the four cases of SCTP in Section 4.1 and 4.2.

4.1 Analytic Models for SF-SCTP and Original SCTP

In this section, we will first obtained the throughput expression for one subflow in SF-SCTP-FCC according to the simplified congestion window behavior shown in Fig. 4.1(b). Using the subflow throughput expression as basis, the throughput models for SF-SCTP-FCC, SF-SCTP-noFCC and the original SCTP are derived.

From the fractional congestion window behavior in Fig. 4.1(b), for SF_i, we have

$$W_j^i = \frac{W_{j-1}^i}{2} + \frac{M}{\Phi} X_j^i \tag{4.3}$$

where M is MTU, Φ is the variable that controls subflow's *cwnd* growth rate, W_j^i is the *cwnd* size in bytes at the end of j^{th} Fast Retransmit Period, which is the period between two adjacent Fast Retransmit events, and X_j^i is the RTT where the first packet drop happens in the j^{th} Fast Retransmit Period.

After SF_i reaches steady condition, from Eq. (4.3), the steady state mean of $cwnd_i$ is

$$E[W^i] = \frac{2M}{\Phi} E[X^i] \tag{4.4}$$

Unlike TCP, which strictly requires that the number of outstanding data bytes, represented by D_o, be less than or equal to the current *cwnd* value, SCTP allows D_o to be at most one packet size greater than the value of *cwnd*. In TCP, the transmission of a packet is prohibited if such a transmission would make D_o greater than the current *cwnd* value. However, in SCTP, a packet can be transmitted as long as D_o does not exceed *cwnd*. When applied in our SF-SCTP design, in each RTT, SF_i can send out a total of $(\lfloor \frac{w}{K_i} \rfloor + 1) \times K_i$ bytes, which could be approximately averaged as $w + \frac{K_i}{2}$, where w is the current value of $cwnd_i$ and K_i is the packet size of SF_i.

As illustrated in Fig. 4.1(b) for $\Phi = 2$, the *cwnd* increases by one MTU (M) per Φ RTTs. After $(h \times \Phi)$ RTTs, the *cwnd* in the j^{th} Fast Retransmit Period is incremented to $\frac{W_{j-1}^i}{2} + hM$, and will stay at this level for Φ RTTs. During the segment of those Φ RTTs, a total of approximately $(\frac{W_{j-1}^i}{2} + hM + \frac{K_i}{2}) \times \Phi$ bytes are sent. Excluding the $(X_j^i + 1)$-th RTT, there are a total of X_j^i/Φ of such transmission segments. Therefore, the number of

packets sent out in the j^{th} Fast Retransmit Period, N_j^i, can be expressed as

$$N_j^i = \Big(\sum_{h=0}^{X_j^i/\Phi - 1} \{ (\frac{W_{j-1}^i}{2} + hM + \frac{K_i}{2}) \times \Phi \} + \beta_j^i \Big) / K_i \tag{4.5}$$

where β_j^i is the number of bytes transmitted in the $(X_j^i + 1)$-th RTT. Depending on the position of the lost packet, from Fig. 4.1(b) we can tell that β_j^i has a random value between 0 and W_j^i. On the average, $E[\beta_j^i]$ is approximately equal to $\frac{W_j^i}{2}$. By assuming that $\{W^i\}$ and $\{X^i\}$ are independent Markov processes, we have $E[W_{j-1}^i \times X_j^i] = E[W_{j-1}^i] \times E[X_j^i] = E[W^i] \times E[X^i] = \frac{\Phi E[W^i]^2}{2M}$ since $E[W^i] = \frac{2M}{\Phi} E[X^i]$ according to Eq. (4.4). Taking the mean of Eq. (4.5) and substituting $E[W_{j-1}^i \times X_j^i]$ with $\frac{\Phi E[W^i]^2}{2M}$, we have the mean for N_j^i as

$$E[N^i] = \Big(\frac{3\Phi}{8M} E^2[W^i] + \frac{K_i\Phi + 2M - M\Phi}{4M} E[W^i] \Big) / K_i \tag{4.6}$$

Let α_j^i represent a random process, which is the number of packets sent in the j^{th} Fast Retransmit Period up to and including the first packet that is lost. The probability that $\alpha_j^i = m$ is equal to the probability that exactly $m-1$ packets are successfully transmitted before a loss occurs. Therefore, we have

$$P[\alpha_j^i = m] = (1 - p_i)^{m-1} p_i \tag{4.7}$$

Thus, the mean of α^i is

$$E[\alpha^i] = \sum_{m=1}^{\infty} m \times (1-p_i)^{m-1} p_i = \frac{1}{p_i} \tag{4.8}$$

To find the solution for Eq. (4.6), we must establish the relationship between the packet drop rate p_i and the mean number of transmitted packets in the a Fast Retransmit Period $E[N^i]$. $E[N^i]$ consists of two components: (1) packets sent out up to and including the first lost packet, which is represented by α^i, (2) packets transmitted after the first lost packet and before the sender is informed the packet loss(es). After a packet drop, it takes the sender another RTT to detect the loss. During this period, approximately $E[W]/K_i$ more packets are sent out, including packets in the same window as the dropped packet (the X_j^i RTT in Fig. 4.1(b)). Therefore we have

$$E\left[N^i\right] = E[\alpha^i] + E[W]/K_i = \frac{1}{p_i} + E[W]/K_i \tag{4.9}$$

From Eqs. (4.6) and (4.9), we have the solution for $E\left[W^i\right]$ of SF_i as

$$\frac{E[W_i]}{M} = \frac{2 + \Phi - \eta_i\Phi + \sqrt{(2+\Phi-\eta_i\Phi)^2 + \frac{24\Phi\eta_i}{p_i}}}{3\Phi} \tag{4.10}$$

$$\approx \sqrt{\frac{8\eta_i}{3\Phi p_i}} \qquad \text{for small } p_i \tag{4.11}$$

where $\eta_i = \frac{K_i}{M}$, K_i is the packet size for SF_i and M is the MTU size.

Given Eqs. (4.9), (4.10) and $E[A^i] = RTT_i \times (E[X^i] + 1)$, Eq. (4.1), which is now the

throughput for SF_i in SF-SCTP-FCC (Case 4), can be solved as

$$\frac{B_i(p_i)}{M} = \frac{6\eta_i + 2p_i\Big(2+\Phi-\eta_i\Phi+\sqrt{(2+\Phi-\eta_i\Phi)^2+\frac{24\Phi\eta_i}{p_i}}\Big)}{p_i\Big(8+\Phi-\eta_i\Phi+\sqrt{(2+\Phi-\eta_i\Phi)^2+\frac{24\Phi\eta_i}{p_i}}\Big)RTT_i} \tag{4.12}$$

$$\approx \frac{1}{RTT_i}\sqrt{\frac{3\eta_i}{2\Phi p_i}} \qquad \text{for small } p_i \tag{4.13}$$

When $\Phi = 1$, Eq. (4.13) is the same as the throughput rate found for TCP model derived in Refs. [28, 55] when no delay-ACK is implemented.

The Case 4 SCTP's throughput B_{Case4} for the entire SF-SCTP association is the sum of all subflows' throughput:

$$\frac{B_{Case4}(\Phi)}{M} = \sum_{i=1}^{n}\frac{B_i(p_i)}{M} \approx \sum_{i=1}^{n}\frac{1}{RTT_i}\sqrt{\frac{3\eta_i}{2\Phi p_i}} \tag{4.14}$$

For $\Phi = 1$, Eq. (4.14) gives the throughput for Case 3 SCTP:

$$\frac{B_{Case3}}{M} = \frac{B_{Case4}(\Phi=1)}{M} \approx \sum_{i=1}^{n}\frac{1}{RTT_i}\sqrt{\frac{3\eta_i}{2p_i}} \tag{4.15}$$

When all subflows has the same packet size $\eta_i = \eta$, the same packet drop rate $p_i = p$, and the same Round Trip Time $RTT_i = RTT$, Eqs. (4.14) and (4.15) become

$$\frac{B_{Case4}}{M} \approx \frac{n}{RTT}\sqrt{\frac{3\eta}{2\Phi p}} \tag{4.16}$$

$$\frac{B_{Case3}}{M} \approx \frac{n}{RTT}\sqrt{\frac{3\eta}{2p}} \tag{4.17}$$

When there is only one subflow and $\Phi = 1$, from Eq. (4.14), we get the throughput expression for the original SCTP as:

$$\frac{B_{Case1}}{M} \approx \frac{1}{RTT_o}\sqrt{\frac{3\eta_o}{2p_o}} \tag{4.18}$$

where η_o, RTT_o and p_o are the packet size over MTU, RTT and packet drop rate for the original SCTP, respectively.

Before we verify our analytic models for SCTP of Cases 1, 3 and 4, let us briefly review some of assumptions that we make in our models:

Assumption 1 *Packet losses are recovered exclusively by Fast Retransmits.*

Assumption 2 *Before a Fast Retransmit, $cwnd_i$ shall not be smaller than $4 * MTU$ so that the new $cwnd_i$ after a Fast Retransmit would be always half the old $cwnd_i$, i.e., $cwnd_i := \max(2 * MTU, 0.5 * cwnd_i)$ (Fig. 4).*

This assumption has also been implicitly made in Refs. [28, 55, 71, 72]. Since the *cwnd* growth parameter Φ is positively related to the number of subflows in order for SF-SCTP-FCC to be fair towards the original SCTP, when the number of subflows in SF-SCTP-FCC increases, from Eq (4.10), the mean of $cwnd_i$ becomes smaller. Therefore, this assumption may no longer hold and the analytic models may underestimate the throughput since *cwnd* size is underestimated.

Assumption 3 *For simplicity, we ignore the fact that SF_i employs a slow start after a Fast Retransmit since $cwnd_i$ is set to $ssthresh_i$ (Fig. 4).*

When the occurrence of time-outs become noticeable, our analytic model overestimates the throughput. A better estimate of the throughput can be obtained by Eq. (4.2) since it takes time-outs into consideration[1]. However, we would like to point out that, when packet drop rate is high and time-outs happen frequently, even Eq. (4.2) overestimates the actual throughput.

Compared with congestion avoidance period, this slow start period, which lasts about one RTT, appears to be short and thus negligible. However, because the *cwnd* size increases exponentially in slow start phase, the slow start after each Fast Retransmit event actually serves as a boost and give SCTP (or SF_i in SF-SCTP) an important lift to recover its *cwnd* (or $cwnd_i$). When there are large number of subflows in SF-SCTP, the aggregate effect of slow start phenomenon can no longer be ignored in calculating the throughput. Since our analytic models do not capture slow start period, they may underestimate the actual throughput.

In general, the ns-2 simulation experiments verify that when packet drop rate is high, analytic models overestimate the throughput, as expected by Assumption 1. As implied by Assumption 2 above, we observe that, as the number of subflows increase, underestimating the throughput by our models increases. Also, ns-2 experiments show that when packet drop rate is low and packet losses are recovered exclusively by Fast Retransmit, our

[1]Given the throughput model that covers Fast Retransmits, Ref. [55] has a detailed derivation of Eq. (4.2).

analytic models underestimate the actual throughput, as discussed in Assumption 3. Detailed interpretation of ns-2 experiments are presented in Chapter 5. Chapter 6 concludes the thesis and provides future research directions for this work.

4.2 Analytic Model for SF-SCTP-shared

To illustrate the deteriorating effect of false sharing on SCTP throughput, SF-SCTP-shared (Case 2) where all subflows share one congestion control is included in this study. In this section, we derive the throughput upper limit model for SF-SCTP-shared based on the assumption that the Diff-Serv network treats the packets coming from SF-SCTP-shared only with different drop rates according to their marked DSCPs, and that the packets shall not arrive at the receiver out of order. This assumption excludes the negative effects on SF-SCTP-shared's throughput due to the unnecessary Fast Retransmits or time-outs resulting from the out of order arrivals of packets at the receiver.

To clarify the comparison between SF-SCTP-shared and SF-SCTP (Case 3 and 4), we consider an SF-SCTP-shared association, which has the same number of subflows as the SF-SCTP in Section 4.1. Also, we use the same packet drop rate p_i for SF_i in SF-SCTP-shared as in SF-SCTP. Besides, all subflows in SF-SCTP-shared have the same packet size ($K_{sh} = \eta_{sh} * M$). In SF-SCTP-shared, a WRR scheduler is employed to apportion the throughput among subflows, where λ_i represents the percentage of the bandwidth assigned to SF_i. Assuming that λ_i and p_i are known, the average packet drop rate of

SF-SCTP-shared can be expressed as

$$\bar{p}_{sh} = \sum_{i=1}^{n} \lambda_i p_i \tag{4.19}$$

To derive the throughput expression for SF-SCTP-shared, let us start with the following expression which is similar to Eq. (4.9)[2] in Section 4.1:

$$E[N_{sh}] = E[\alpha_{sh}] + E[W_{sh}]/K_{sh} \tag{4.20}$$

where α_{sh} is the random process representing the number of packets sent in a Fast Retransmit Period of SF-SCTP-shared up to and including the first packet lost. The variable W_{sh} represents the *cwnd* size of SF-SCTP-shared in bytes.

In the following steps, we start to derive the expression for mean of α_{sh}, $E[\alpha_{sh}]$, from the simplest situation where all the subflows share the bandwidth equally, that is $\lambda_i = \frac{1}{n}$ for SF_i. In this situation, using a round robin scheduler, the probability that $(m-1)$ packets are successfully transmitted while the m^{th} one is dropped is

$$P[\alpha_{sh} = m] = \sum_{i=1}^{n} \frac{1}{n} \Big(\prod_{s=1}^{n} (1-p_s)^{\lfloor \frac{(m-1)}{n} \rfloor} \Big) \Big(\prod_{r=1}^{(m-1) \bmod n} (1-p_x) \Big) p_i \tag{4.21}$$

where i corresponds to the SF_i that drops the m^{th} packet, and x is defined as

$$x = \begin{cases} (n+i-r) \bmod n & \text{if} \quad x \neq 0 \\ n & \text{otherwise} \end{cases}$$

[2]This is the SCTP version of Eq. (2) in Ref. [55]

The mean of α_{sh} is

$$\begin{aligned} E[\alpha_{sh}] &= \sum_{m=1}^{\infty} mP[\alpha_{sh}=m] = \sum_{m=1}^{\infty} m \sum_{i=1}^{n} \frac{1}{n}\Big(\prod_{s=1}^{n}(1-p_s)^{\lfloor \frac{(m-1)}{n} \rfloor}\Big)\Big(\prod_{r=1}^{(m-1) \bmod n}(1-p_x)\Big)p_i \\ &= \sum_{m=1}^{\infty} m \prod_{s=1}^{n}(1-p_s)^{\lfloor \frac{(m-1)}{n} \rfloor}\Big(\sum_{i=1}^{n}\frac{1}{n}\prod_{r=1}^{(m-1) \bmod n}(1-p_x)p_i\Big) \end{aligned} \tag{4.22}$$

To simplify Eq. (4.22), let us define $m = nl + h$ for $l = 0, \ldots, \infty$, and $h = 1, \ldots, n$, and $Y(m)$ as a periodic function with the period of n:

$$Y(m) = \sum_{i=1}^{n}\frac{1}{n}\Big(\prod_{r=1}^{(m-1) \bmod n}(1-p_x)\Big)p_i = \sum_{i=1}^{n}\frac{1}{n}\Big(\prod_{r=1}^{h-1}(1-p_x)\Big)p_i = Y(h) \approx \bar{p}_{sh} \quad \text{for small } p_i \tag{4.23}$$

Given Eq. (4.23), Eq. (4.22) can be simplified as

$$\begin{aligned} E[\alpha_{sh}] &= \sum_{l=0}^{\infty}\sum_{h=1}^{\infty}(nl+h)\prod_{s=1}^{n}(1-p_s)^l\, Y(h) = \sum_{h=1}^{\infty}\sum_{l=0}^{\infty}Y(h)\Big(nl\prod_{s=1}^{n}(1-p_s)^l + h\prod_{s=1}^{n}(1-p_s)^l\Big) \\ &= \sum_{h=1}^{n}Y(h)\Big(\frac{nC}{(1-C)^2}+\frac{h}{1-C}\Big) \end{aligned} \tag{4.24}$$

where C is defined as

$$C = \prod_{s=1}^{n}(1-p_s) \approx 1 - \sum_{s=1}^{n}p_s = 1 - n\bar{p}_{sh} \quad \text{for small } p_s \tag{4.25}$$

From Eqs. (4.23), (4.24), and (4.25), we obtain

$$E[\alpha_{sh}] = \sum_{h=1}^{n}Y(h)\Big(\frac{1-n\bar{p}_{sh}}{n\bar{p}_{sh}}+\frac{h}{n\bar{p}_{sh}}\Big) \approx \sum_{h=1}^{n}Y(h)\frac{1}{n\bar{p}_{sh}} \approx \frac{1}{\bar{p}_{sh}} \tag{4.26}$$

Comparing Eq. (4.26) with Eq. (4.8), for small packet drop rates, we can use the equivalent

packet drop rate $\bar{p}_{sh}$ to compute the throughput for SF-SCTP-shared as:

$$\frac{B_{Case2}}{M} = \frac{B_{Case1}(\eta_{sh})}{M} \approx \frac{1}{RTT_{sh}}\sqrt{\frac{3\eta_{sh}}{2\bar{p}_{sh}}} \qquad \text{for small value of } \bar{p}_{sh} \tag{4.27}$$

Eq. (4.27) is derived based on the condition of $\lambda_i = \frac{1}{n}$. It can be easily proved that, as long as the RTTs for all subflows in Case 2 SCTP are the same, the same expression as shown in Eq. (4.27) can be obtained even when the subflows do not get equal share of the throughput. Therefore, Eq. (4.27) presents the throughput model for Case 2 SCTP where all subflows has the same RTTs. If the RTTs for the subflows in Case 2 SCTP are not the same, unnecessary Fast Retransmits and time-outs, due to unordered packet arrivals at the receiver, may cause the actual throughput to be much lower than the analytic result given by Eq. (4.27).

4.3 Comparison of the Four Cases of SCTP

Comparing Eq. (4.16) with Eq. (4.17), we find that, for the same packet drop rate p_i, packet size K_i and Round Trip Time RTT_i, we have the throughput relationship between SF-SCTP-noFCC (Case 3) and SF-SCTP-FCC (Case 4) as

$$B_{Case3} = \sqrt{\Phi} B_{Case4} \tag{4.28}$$

If $\Phi = 1$, SF-SCTP-FCC has the same throughput as SF-SCTP-noFCC.

If SF-SCTP (Case 3 and 4) has only one subflow and its packet drop rate, packet size, and RTT are the same as those for the original SCTP (Case 1), SF-SCTP and the original SCTP has the same throughput.

Comparing the original SCTP with SF-SCTP-FCC using Eqs. (4.14) and (4.18), we find that, when $\Phi = n^2$, SF-SCTP-FCC achieves roughly the same throughput as the original SCTP, for equal η, RTT and p values. This result implies that, when SF-SCTP increases its combined congestion window by $1/n * MTU$ per RTT, it reduces its aggressiveness and becomes fair to the original SCTP. The comparison between the original SCTP and SF-SCTP-FCC are made under two assumptions. We first assume that packet losses are exclusively recovered by Fast Retransmit. When packet drop rate becomes high enough, time-outs are no longer rare events. After a time-out, a subflow in SF-SCTP begins slow start without affecting other subflows' congestion state, while for the original SCTP, the whole association must begin with slow start. The SF-SCTP's ability to absorb packet losses in a subset of subflows makes it more tolerable to losses and more aggressive than the original SCTP. We also assume that subflow's *cwnd* shall not be less than $4 * MTU$ before a Fast Retransmit. When there are a lot of subflows in SF-SCTP-FCC, its *cwnd* must increase very slowly in order to be fair with the original SCTP. However, the slower growth rate a subflow has, the more likely its congestion window may be less than $4*MTU$ before a Fast Retransmit. High packet drop rates may also increase the possibility that a subflow's *cwnd* is less than $4 * MTU$ before a Fast Retransmit.

From the discussion of these assumptions, we can point out that SF-SCTP-FCC has a better chance to achieve fairness towards the original SCTP in a network with low packet drop rates (normally high speed network) than a network with high packet drop rates. However, even if the assumptions above do not hold, we can still safely claim that fractional congestion control alleviates the aggressiveness of SF-SCTP. The fairness between SF-SCTP-FCC and the original SCTP are further discussed in Section 5.4.

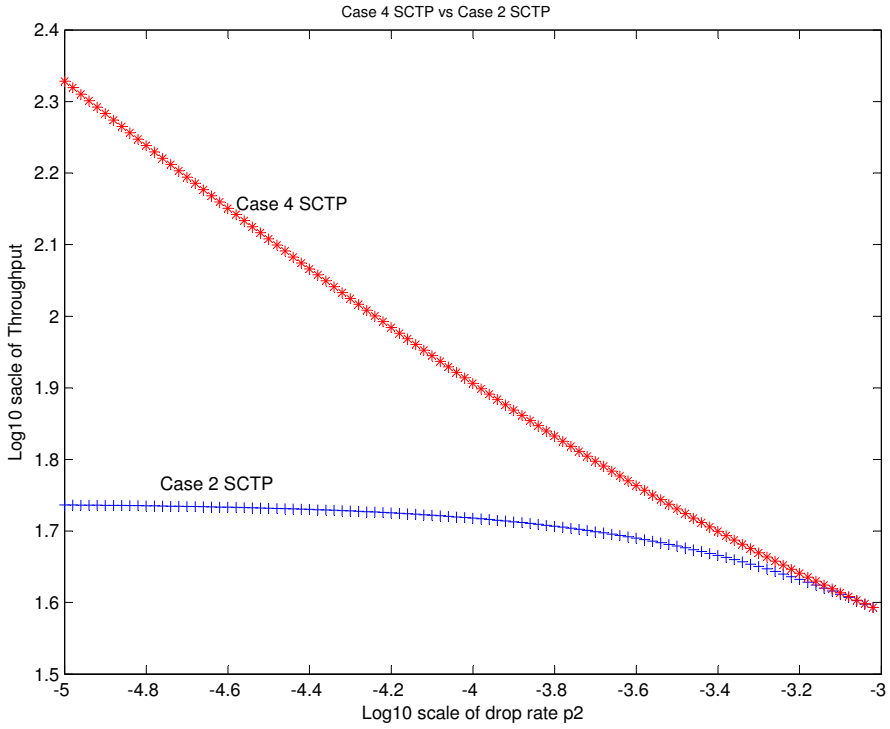


Figure 4.2: Comparison of Case 4 SCTP with Case 2 SCTP.

Fig. 4.2 is the throughput comparison of SF-SCTP-FCC with SF-SCTP-shared (Case 2) using Eqs. (4.14) and (4.27). We define two subflows, SF_1 and SF_2 in both cases of SCTP, where $p_1 = 10^{-3}$, p_2 varies from 10^{-5} to 10^{-3}, $RTT_1 = RTT_2 = 1\,\text{sec}$, $\Phi = 4$ and

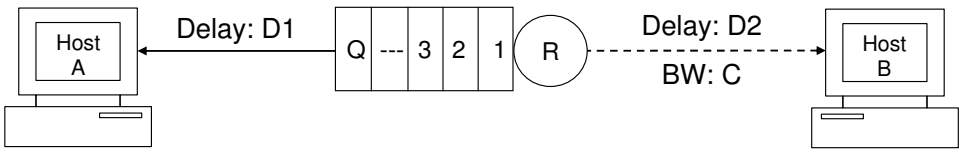


Figure 4.3: The topology of a network with a single bottleneck link

$\eta_1 = \eta_2 = 1$. Fig. 4.2 shows that, when all subflows have the same RTTs and packet drop rates ($p_1 = p_2 = 10^{-3}$), false sharing is no longer a problem and therefore, SF-SCTP-FCC achieves the same throughput as SF-SCTP-shared. However, when $p_1 = 10^{-3}$ and $p_2 = 10^{-5}$, SF-SCTP-FCC obtains almost 3 times more throughput than SF-SCTP-shared because the SF_2 (higher priority) in Case 2 SCTP was penalized due to the falsely shared information with SF_1 (lower priority).

4.4 Single Bottleneck Network

In this section, we will study the performance of SF-SCTP under a single bottleneck network as depicted in Fig. 4.3 using analytic models. The bottleneck network consists of two SCTP hosts (A and B), one router (R), and two links. The link between host A and router R has a link delay of D_1 and has almost an infinite bandwidth when compared with the bottleneck link. The bottleneck link exists between router R and host B with a bandwidth of C and a delay of D_2. The router implements the Drop-Tail buffering mechanism with a queue size of Q. Between host A and B, there are multiple SCTP flows running concurrently, competing with each other for the limited bottleneck bandwidth.

For derivation purpose, we define a number of variables. We define the round trip link delay as the basic RTT, represented by R_L and $R_L = 2(D_1 + D_2)$. The maximum queue delay caused by the fulled occupied queue can be calculated as $D_Q = Q * K/C$, where Q is the queue size in packets, K is the packet size in bytes, and C is the bottleneck bandwidth in number of bytes. The maximum RTT when queue reaches full capacity is represented by R_Q and $R_Q = R_L + D_Q$. We define W_L as the product of propagation delay and bottleneck bandwidth ($W_L = C * R_L$). W_L represents the maximum number of bytes which could be hold by the links between host A and B. We define W_Q as $W_Q = C * R_Q = W_L + Q * K$, which is the maximum number of bytes both the link and the buffer can hold. The overall *cwnd* of the SF-SCTP flows, which control the total number of bytes outstanding in the communication channel, cannot exceed W_Q. Otherwise, the bottleneck queue will start dropping packets and a subset subflows of the SF-SCTP will be affected.

For a single bottleneck network where the queue delay may not negligible, a deterministic lower and upper model for the throughput and packet loss rate of SF-SCTP are derived. The differences between the deterministic models in this section and the stochastic model in previous sections lie in the following aspects:

- **Applicability:** The deterministic models can only be applied in a single bottleneck network where drop-tail buffering mechanism is used and all packet losses are due to congestion. The basic RTT, bottleneck link bandwidth and queue size have to be known in order to use the deterministic models. Unlike the deterministic models, the

stochastic model can be used in any type of network where queue delay is ignorable. Also, the packet losses in the stochastic model can either due to congestion or non-congestion. However, the RTTs and packet loss rates of all subflows in SF-SCTP must be known prior before applying the stochastic model.

- **Accuracy:** The stochastic models are confirmed to be accurate when packet loss rate is smaller than 10^{-2} [55]. The deterministic models, however, only give the throughput and packet loss rate boundaries for SF-SCTP with multiple subflows. When the SF-SCTP has only one subflow, the lower and upper deterministic models converge and accurate predictions can be obtained.

- **Functionality:** With deterministic models, one can get the throughput and packet loss rate boundaries for the SF-SCTP. For the stochastic model, one can only obtain the throughput since packet loss rate is the precondition. For SF-SCTP with only one subflow, the deterministic models can accurately capture the instantaneous behavior of the *cwnd*, while the stochastic model can only get the average *cwnd* size.

4.4.1 Queue Occupancy and *cwnd* Behavior

According the transmission rate of SF-SCTP, at any time, the bottleneck queue is in one of the following three states: Queue Buildup (QBU), Queue Draining (QDN), or Queue Steady (QSD) [56]. A summary of those three states and the state transition are presented

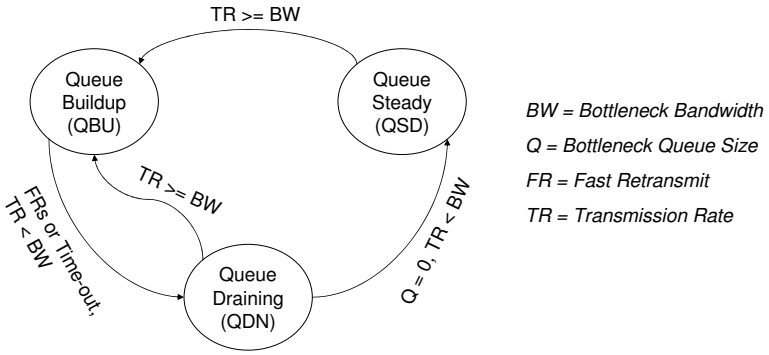


Figure 4.4: Based on transmission rate, the bottleneck queue can be currently filling (QBU), draining (QDN) or being steady (QSD).

in the following[3]:

- *Queue Buildup (QBU) state:* In this period, the transmission rate of SF-SCTP remains almost constant at a rate slightly above the bottleneck bandwidth. Since the packet arrival rate at the bottleneck queue, which equals the SF-SCTP's transmission rate, is higher than the packet departure rate, which equals the bottleneck bandwidth, packets waiting at the queue for departure are accumulating. As the bottleneck queue occupancy increases, the RTT for the SF-SCTP also increases. At the same time, the *cwnd* of SF-SCTP keeps increasing before packet drops happen. The synchronous increasing of the *cwnd* and RTT explains why the transmission rate remains roughly constant in this period because the effect of increasing *cwnd* is equally offset by the increasing of RTT. The queue remains in the QBU state and will not make its transition to the QDN state until it is full and

[3]The detail discussions of the QBU, QDN, and QSD states together with the state transitions can be found in [77]

starts to drop packets and until SF-SCTP responds by cutting its transmission rate to a level lower than the bottleneck bandwidth.

- *Queue Draining (QDN) state:* The transmission rate of SF-SCTP is less than the bottleneck bandwidth such that the queue is in the process of being depleted. The reduction of the bottleneck queue occupancy has the effect of decreasing the RTT. Since transmission rate is directly proportional to the *cwnd* over RTT, the abrupt decrease of RTT explains the sharp increase of the transmission rate in the QDN period. Both the *cwnd* and the transmission rate of SF-SCTP begin their recovery during the QDN state. When the queue becomes completely depleted, if the transmission rate is still less than the bottleneck bandwidth, the queue will transit from the QDN to the QSD state. If the transmission rate is recovered to a level equal to or greater than the bottleneck bandwidth before the queue is empty, the queue will move from the QDN back to the QBU state.

- *Queue Steady (QSD) state:* The queue occupancy in this period remains as almost empty since the packet arrival rate is less than the departure rate at the bottleneck queue. When the *cwnd* is recovered to a certain value such that the transmission rate of SF-SCTP is equal to the bottleneck bandwidth, the queue will transit to the QBU state.

The duration of the QDN period is positively related to the value of Q, C, R_L and Φ, while negatively related to n, the number of subflows in SF-SCTP. The duration of the QDN period is positively related to Q, while negatively related to the C and n because of the alleviated global synchronization problem. The length of the QSD period is positively

related to C and R_L, while negatively related to Q and n.

The state diagram of Fig. 4.4 shows the transitions of those three states. A *queue cycle* is defined such that it starts from the beginning of the QBU state and ends the next time when the state is back into the QBU state again. Depending on the queue size and the number of subflows in SF-SCTP, a queue cycle can either be QBU+QDN or QBU+QDN+QSD. If the queue cycle does not include the QSD state, it means the queue will not be completely emptied when SF-SCTP is affected by packet losses.

4.4.2 Deterministic Models

In this section, we derive the analytic model for SF-SCTP with n subflows. In SF-SCTP, each subflow its *cwnd* by one MTU per Φ RTTs and decrease its *cwnd* to half after a Fast Retransmit. For the SF-SCTP, in average, its overall *cwnd* increases at a rate of nM/Φ bytes per RTT. When the overall *cwnd* increases to a level above W_Q, the drop-tail queue starts to drop some packet(s) because of queue overflow. Given that k subflows in the SF-SCTP are affected, the SF-SCTP reduces its *cwnd* to $(2n-k)/(2n)$ of its previous highest value W_Q, assuming that each subflow has equal share in the overall *cwnd*. For simplicity, we approximately $W_Q \approx W_Q$. The models presented in this section are deterministic because we assume that all queue cycles, also named the Fast Retransmission Periods in the stochastic model, are identical during the lifetime of the SF-SCTP. Also different than the previous stochastic model, the models are fluid models since we assume the *cwnd*

value of SF-SCTP is continuous, while in reality, it is discrete.

When packet losses happen, the worst situation for SF-SCTP is that all subflows inside are affected, while the best situation is that only one subflow is affected. When FCC is not implemented, all subflows in the SF-SCTP are likely to be affected by packet losses due to the global synchronization propriety of drop-tail queue. With FCC, the global synchronization problem are alleviated and packet losses may only affected a subset of the SF-SCTP. Based on the worst and best situation, the throughput lower and upper limit model can be derived, respectively, while in the throughput upper limit model, the overall *cwnd* is between $(2n-1)W_Q/(2n)$ to W_Q.

4.4.3 Throughput Lower Limit for SF-SCTP

In the throughput lower limit model, the overall *cwnd* of the SF-SCTP ranges from $W_Q/2$ to W_Q. If the minimum overall *cwnd* value $W_Q/2$ is greater than the product of bottleneck bandwidth and link delay W_L, the queue cycle only includes the QBU and QDN states and the bottleneck queue will always have packets waiting inside for departure. Otherwise, if $W_Q/2 \leq W_L$, the queue cycle will include the QSD state and the queue will be depleted after Fast Retransmit events.

When a queue cycle consists of only the QBU and QDN states, the instantaneous value of the RTT, represented by $r(t)$, is the sum of the link delay and queue delay:

$$r(t) = R_L + \frac{w(t) - W_L}{C} = \frac{W_L}{C} + \frac{w(t) - W_L}{C} = \frac{w(t)}{C}, \tag{4.29}$$

where $w(t)$ is the instantaneous *cwnd* value, and $w(t) - W_L$ is the number of bytes waiting in the queue.

Because of the packets accumulating at the bottleneck queue, the increase of *cwnd* will result in the increase of RTT. With the increasing of RTT, the growth rate of overall *cwnd* becomes slower in the congestion avoidance period. Assume at time t_1, $w(t_1) = w_1$ and $r(t_1) = r_1$. After a RTT, which is approximately r_1, the *cwnd* will increased by nM/Φ to $w(t_2) = w_2 = w_1 + nM/\Phi$. Thus, the derivative of the *cwnd* can be expressed as

$$\frac{dw}{dt} = \frac{w_2 - w_1}{t_2 - t_1} = \frac{nM/\Phi}{r(t)} = \frac{nMC}{\Phi w(t)}. \tag{4.30}$$

Given $w(0) = W_Q/2$, the above equation can be solved as

$$w(t) = \sqrt{\frac{W_Q^2}{4} + 2nMCt/\Phi}. \tag{4.31}$$

Also as we know, at the end of the queue cycle, $w(T_{low1}) = W_Q$, where T_{low1} represent the length of a queue cycle. Thus, we can solve the the during of the queue cycle as

$$T_{low1} = \frac{3\Phi W_Q^2}{8nMC}. \tag{4.32}$$

As we know, the instantaneous transmission rate of SF-SCTP is $w(t)/r(t)$. Hence, the integration $\int_0^{T_{low1}} \frac{w(t)}{r(t)} dt$ represents the total number of bytes transmitted out during a queue cycle. Therefore, the throughput of the SCTP can be calculated as

$$B_{low1} = \frac{\int_0^{T_{low1}} \frac{w(t)}{r(t)} dt}{T_{low1}} = \frac{\int_0^{T_{low1}} C dt}{T_{low1}} = C. \tag{4.33}$$

At the end of the queue cycle, since all subflows are affected by packet losses, there are a total of n Fast Retransmits in the SF-SCTP. Therefore, the SF-SCTP's Fast Retransmit rate, which is an effective indication of the packet loss rate, can be obtained as

$$P_{low1} = \frac{n/T_{low1}}{B_{low1}/K} = \frac{8n^2KM}{3\Phi W_Q^2}. \tag{4.34}$$

The above Eqs. (4.29), (4.30), (4.31), (4.32), (4.33) and (4.34) are derived under the circumstance that a queue cycle consists of only QBU and QDN states.

When a queue cycle includes the QSD state, $\frac{W_Q}{2}$ is smaller than W_L. While $w(t) \in [\frac{W_Q}{2}, W_L]$, the queue is empty and the RTT is fixed at R_L. Therefore, the instantaneous value of the RTT for the small queue case can be expressed as

$$r(t) = \begin{cases} R_L & \text{when} \quad \frac{W_Q}{2} \leq w \leq W_L \\ \frac{w(t)}{C} & \text{when} \quad W_L < w \leq W_Q \end{cases} \tag{4.35}$$

The overall *cwnd* of SF-SCTP will increase linearly during the QSD period. During the QBU period, the *cwnd* increases similarly as in the large queue delay case. Therefore, the instantaneous *cwnd* value for the small queue case is

$$w(t) = \begin{cases} \frac{W_Q}{2} + \frac{nMC}{\Phi W_L}t & \text{when} \quad 0 \leq t \leq \tau \\ \sqrt{W_L^2 + 2nMC(t-\tau)/\Phi} & \text{when} \quad \tau < t < T_{low2} \end{cases} \tag{4.36}$$

where T_{low2} is the length of the queue cycle for the small queue case, τ is the duration that the queue stay at the QSD period and $\tau = \frac{\Phi(2W_L^2 - W_Q W_L)}{2nMC}$ and $w(\tau) = W_L$.

Similarly as in the large queue case, we get the queue cycle length, throughput, and Fast Retransmit rate of SF-SCTP in the small queue case as:

$$T_{low2} = \Phi \frac{W_Q^2 - W_Q W_L + W_L^2}{2nMC} \tag{4.37}$$

$$B_{low2} = \frac{\int_0^{T_{low2}} \frac{w(t)}{r(t)} dt}{T_{low2}} = \frac{\int_0^{\tau} \frac{w(t)}{R_L} dt + \int_{\tau}^{T_{low2}} C dt}{T_{low2}} = \frac{3W_Q^2 C}{4W_Q^2 - 4W_Q W_L + 4W_L^2} \tag{4.38}$$

The packet loss rate of the SCTP, which is also the network drop rate, can be calculated as

$$P_{low2} = \frac{n/T_{low2}}{B_{low2}/K} = \frac{8n^2 KM}{3\Phi W_Q^2} \tag{4.39}$$

Compare Eq. (4.34) with Eq. (4.39), we find both equations have the same form. Thus, for the throughput lower model, we have an uniform Fast Retransmit rate as

$$P_{low} = P_{low1} = P_{low2} = \frac{8n^2 KM}{3\Phi W_Q^2} \tag{4.40}$$

4.4.4 Throughput Upper Limit for SF-SCTP

In the upper limit model, we assume that, after a Fast Retransmit, the *cwnd* of the SF-SCTP flows reduce to $\frac{2n-1}{2n}$ of its previous value. Only one flows is affected each time the Drop-Tail queue drops packet(s). Similar as in the lower limit model, we separate the derivation in this section to the large queue case where $\frac{2n-1}{2n} W_Q \geq W_L$ and small queue case where $\frac{2n-1}{2n} W_Q < W_L$.

For large queue case where queue cycle consists of only QBU and QDN states, the instantaneous RTT and *cwnd* formula are the same as Eqs. (4.29) and (4.31). However, the boundary condition for $w(t = 0)$ are different than the lower limit model and now $w(0) = \frac{(2n-1)W_Q}{2n}$. Given this condition, the Eq. (4.31) can be solved for the upper limit model as:

$$w(t) = \sqrt{w(0)^2 + \frac{2nMCt}{\Phi}} = \sqrt{\frac{(2n-1)^2}{4n^2}W_Q^2 + \frac{2nMCt}{\Phi}}. \tag{4.41}$$

Since $w(T_{up1}) = W_Q$, we have

$$T_{up1} = \frac{(4n-1)\Phi W_Q^2}{8n^3MC}. \tag{4.42}$$

The SF-SCTP throughput is

$$B_{up1} = \frac{\int_0^{T_{up1}} \frac{w(t)}{r(t)}dt}{T_{up1}} = \frac{\int_0^{T_{up1}} Cdt}{T_{up1}} = C. \tag{4.43}$$

When the drop-tail queue drop packet(s), only one subflow in the SF-SCTP is affected and needs to response with a Fast Retransmit. Thus, the Fast Retransmit rate of the SF-SCTP, which is an effective indicator of the network drop rate, is obtained as

$$P_{up1} = \frac{1/T_{up1}}{B_{up1}/K} = \frac{8n^3KM}{(4n-1)\Phi W_Q^2}. \tag{4.44}$$

For the small queue case where $\frac{2n-1}{2n}W_Q < W_L$, the instantaneous value of the RTT for the small queue case can be expressed as

$$r(t) = \begin{cases} R_L & \text{when} \quad \frac{(2n-1)W_Q}{2n} \le w \le W_L \\ \frac{w(t)}{C} & \text{when} \quad W_L < w \le W_Q \end{cases}. \tag{4.45}$$

The instantaneous *cwnd* value for the small queue case is

$$w(t) = \begin{cases} \frac{(2n-1)W_Q}{2n} + \frac{nMC}{\Phi W_L}t & \text{when} \quad 0 \le t \le \tau \\ \sqrt{W_L^2 + 2nMC(t-\tau)/\Phi} & \text{when} \quad \tau < t < T_{up2} \end{cases} . \tag{4.46}$$

where τ is the duration that the queue stay at the QSD period and $\tau = \frac{\Phi(2nW_L^2-(2n-1)W_QW_L)}{2n^2MC}$ and $w(\tau) = W_L$.

Substitute the boundary condition $w(T) = W_Q$ into Eq. (4.46), we have

$$T_{up2} = \Phi\frac{nW_Q^2 + nW_L^2 - (2n-1)W_QW_L}{2n^2MC}. \tag{4.47}$$

The SF-SCTP throughput is

$$B_{up2} = \frac{\int_0^{T_{up2}} \frac{w(t)}{r(t)}dt}{T_{up2}} = \frac{\int_0^{\tau} \frac{w(t)}{R_L}dt + \int_\tau^{T_{up2}} Cdt}{T_{up2}} = \frac{(4n-1)W_Q^2C}{4n^2W_Q^2 + 4n^2W_L^2 - 4n(2n-1)W_QW_L}. \tag{4.48}$$

The Fast Retransmit rate of the SF-SCTP, which indicates the network drop rate, can be calculated as

$$P_{up2} = \frac{1/T_{up}}{B_{up2}/K} = \frac{8n^3KM}{(4n-1)\Phi W_Q^2}. \tag{4.49}$$

Similar as in the throughput lower model, we find Eq. (4.44) has the same form as Eq. (4.49). Therefore, for the throughput upper model, we also have an uniform Fast Retransmit rate expressed as

$$P_{up} = P_{up1} = P_{up2} = \frac{8n^3KM}{(4n-1)\Phi W_Q^2}. \tag{4.50}$$

We want to point out here that the name of lower and upper model is with regarded to the throughput. For Fast Retransmit rate, we always have $P_{low} \geq P_{up}$.

4.4.5 Deterministic Model for Single SCTP

When number of subflows in SF-SCTP equals one, the throughput lower and upper limit models converges and we get the exact analytic expressions for single SCTP association as:

$$w(t) = \sqrt{\frac{W_Q^2}{4} + 2MCt} \quad \text{large queue case where } W_Q \geq 2W_L. \tag{4.51}$$

$$w(t) = \begin{cases} \frac{W_Q}{2} + \frac{MC}{W_L}t & if\ w(t) \leq W_L \\ \sqrt{W_L^2 + 2MC(t-\tau)} & if\ W_L < w(t) \end{cases} \quad \text{small queue case where } W_Q < 2W_L. \tag{4.52}$$

$$T_{sctp} = \begin{cases} \frac{3W_Q^2}{8MC} & \text{large queue case where } W_Q \geq 2W_L \\ \frac{W_Q^2+W_L^2-W_QW_L}{2MC} & \text{small queue case where } W_Q < 2W_L \end{cases}. \tag{4.53}$$

$$B_{sctp} = \begin{cases} C & \text{large queue case where } W_Q \geq 2W_L \\ \frac{3W_Q^2C}{4W_Q^2+4W_L^2-4W_QW_L} & \text{small queue case where } W_Q < 2W_L \end{cases}. \tag{4.54}$$

$$P_{sctp} = \frac{8KM}{3W_Q^2}. \tag{4.55}$$

Chapter 5

Simulation Experiments

In this Chapter, we report the results of the simulation experiments conducted using ns-2 [70] with the integrated SCTP extensions from University of Delaware [20, 15] and Diff-Serv Model from Nortel Networks [11]. We implement our SF-SCTP by extending the ns-2 SCTP module defined by RFC 2960.

Extensive simulations have been performed to study the behavior of SF-SCTP in a Single Bottleneck Network, a simplified Diff-Serv network and a Wide Area Network[1].

Through simulation experiment, we first investigate the capability of SF-SCTP to support QoS provided by the underlying networks. Two set of experiments are being conducted in Section 5.1. Give a single bottleneck who support QoS through virtual queues, we

[1]Throughout this thesis, we assume the SCTP receiving buffer is large enough and should not place any constrains on the performance of SCTP.

compare the packet drop rate of Original SCTP, SF-SCTP-shared and SF-SCTP-noFCC. Given a simplified Diff-Server network which can provide different packet drop rates or RTTs, experiments have been conducted to verify SF-SCTP's superior capability for supporting QoS. In Section 5.2, ten different experiments have been carried out to study SF-SCTP's behavior in a network with a single bottleneck link. By varying the bottleneck queueing mechanism, queue size, link bandwidth and link delay, we gain better understanding how SF-SCTP exploits the limited bandwidth. Experiments VIII to XI have been performed to investigate how SF-SCTP adapts itself when the bottleneck network has different types of background traffic. In Section 5.3, we performed experiments to investigate how SF-SCTP adapts itself when the bottleneck network has different types of background traffic. Finally, in Section 5.4, we study the effect of Fractional Congestion Control on the congestion control behavior of SF-SCTP.

The obtained results from various simulation experiments have also been used to validate the accuracy of analytic models of the original SCTP, SF-SCTP, and SF-SCTP-shared. Since the SCTP used in our simulation experiments do not include a Fast Recovery mechanism algorithm such as New-Reno SCTP [66], multiple losses in a window of data may result in several Fast Retransmits. As a result, in rare cases, we observed the occurrence of consecutive Fast Retransmits followed by a time-out, which severely reduced the SCTP's throughput and caused it to deviate from the analytic models' expected throughput. When the packet drop rate becomes higher, we find that the analytic models

overestimate the throughput [55].

When deriving the analytic models, we assumed that only one loss indication (Fast Retransmit or time-out) should be counted even if there may be multiple packet losses in one round. In the experiments, we approximate the loss indication rate as

$$\text{loss indication rate } = \frac{\text{sum of the number of Fast Retransmits and time-outs}}{\text{total number of packets transmitted}} \quad (5.1)$$

The loss indication rate from Eq. (5.1) should used instead of the packet drop rate in the analytic models from Section 4.1.

To obtain an accurate value for the throughput from the analytic models, a good approximation for the RTT is required. However, this approximation is difficult since it is hard to calculate the average queue delay (queue length changes dynamically during an experiment). This difficulty is also reported in other simulation experiments [55]. To minimize the effect of this inaccuracy, we either use a relatively small queue size routers and monitor the average queue size to take the average queue delay into account, or use a wide area network where queue delay can be ignored.

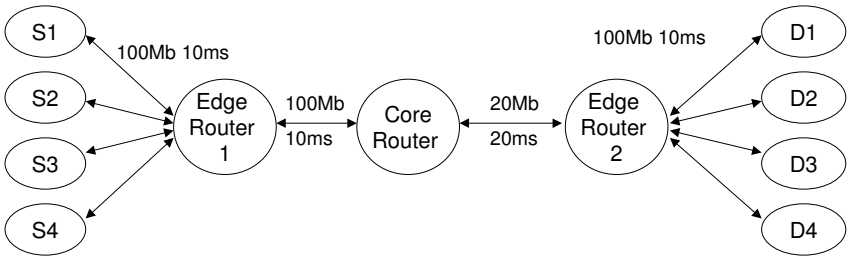


Figure 5.1: Single bottleneck network used to investigate SF-SCTP's ability to support QoS.

5.1 Capability of SF-SCTP to Support QoS

5.1.1 Single Bottleneck Network

5.1.1.1 Experiment Setup

To interpret the results intuitively, we use a simple network topology shown in Fig. 5.1, which has one bottleneck link between the core router and edge router 2.

Table 5.1: Traffic source for nodes S1 through S4.

	traffic type	DSCP	flow(s)
S1	Original SCTP (Case 1)	0	1 BE
S2	SF-SCTP-shared (Case 2)	0 or 1	1/2 BE + 1/2 non-BE
S3	SF-SCTP-noFCC (Case 3)	0 or 1	1 BE + 1 non-BE
S4	background traffic	0 or 1	24 BE + 4 non-BE

Table 5.2: RED queue parameters.

	min thresh	max thresh	max drop rate
Virtual Queue 0 (BE)	$0.2 \times Q \times (1-a)$	$0.8 \times Q \times (1-a)$	0.05
Virtual Queue 1 (non-BE)	$0.2 \times Q \times a$	$0.8 \times Q \times a$	0.05

The traffic types generated by the source nodes S1 through S4 are given in Table 5.1: (1) BE traffic (packets marked with DSCP code 0); and (2) non-BE traffic (packets marked DSCP code 1). Our simplified Diff-Serv network in the experiment provides different packet drop rates for the BE and non-BE packets. The concept of flow in our experiments means a group of outgoing packets sharing one set of congestion control parameters. According to this definition, the SF-SCTP-shared with two subflows can only be considered as one flow because the subflows are sharing one congestion control at the association level. And since these two subflows, one marked with DSCP code 0 and the other with DSCP code 1, are regulated by a round-robin scheduler inside SF-SCTP-shared, we could consider the SF-SCTP-shared association as the composition of 1/2 BE flow and 1/2 non-BE flow. For Case 3 SCTP, since the two subflows implement their own congestion control, the SF-SCTP-noFCC association could be treated as one BE flow and one non-BE flow. Delayed-SACK is not employed for all SCTP associations because during the simulation we found in that the analytic models for SCTP are more accurate when the SCTP receiver immediately acknowledges each arriving packet.

In the core router, only one physical queue is maintained. However, this physical queue is divided into two virtual queues (numbered as 0 and 1). All the BE packets will go through virtual queue 0 while the non-BE packets will use virtual queue 1. The reason we implement two virtual queues instead of two physical queues at the core router is that we would like to maintain a First-In First-Out (FIFO) principle for packets passing the core router. In this implementation, the packets coming from the two subflows of Case 2 SCTP will be delivered in order at the receiver to avoid unnecessary Fast Retransmits.

Both virtual queues at the core router are RED queues whose parameters are given in Table 5.2, where Q is the physical queue size (40 packets), and a is the fraction of the physical queue slots allocated to the virtual queue 1. In the experiments, when the virtual queue size changes as a is varied from 10% to 90%, the packet drop rates for BE and non-BE flows will be increasing and decreasing, respectively.

5.1.1.2 Discussion of Simulation Results

The following observations are based on the data from Figs. 5.2, 5.3(a) and 5.3(b):

- The simulation results confirm the accuracy of the throughput formulas derived in Section 4.1. The high accuracy is achieved when the loss indication rate from Eq. (5.1) is used instead of the packet drop rate.

- Although SF-SCTP-shared has 1/2 non-BE subflow, which is treated with higher priority by the Diff-Serv network, it achieves roughly the same throughput as the original

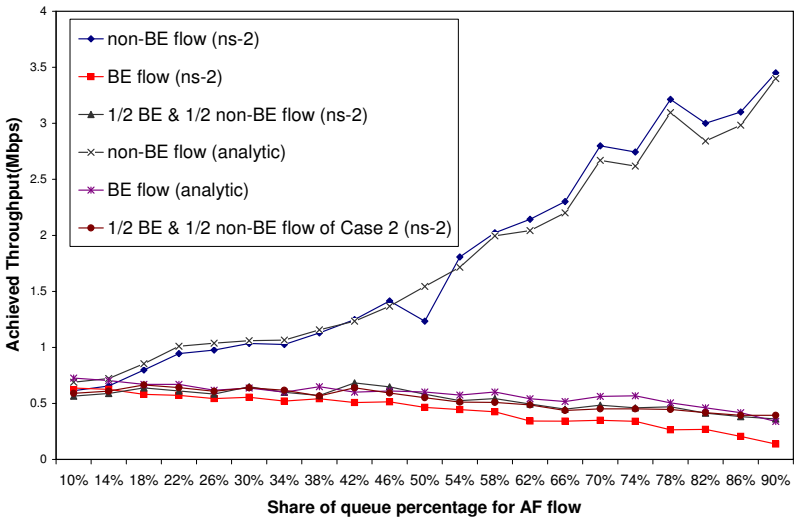
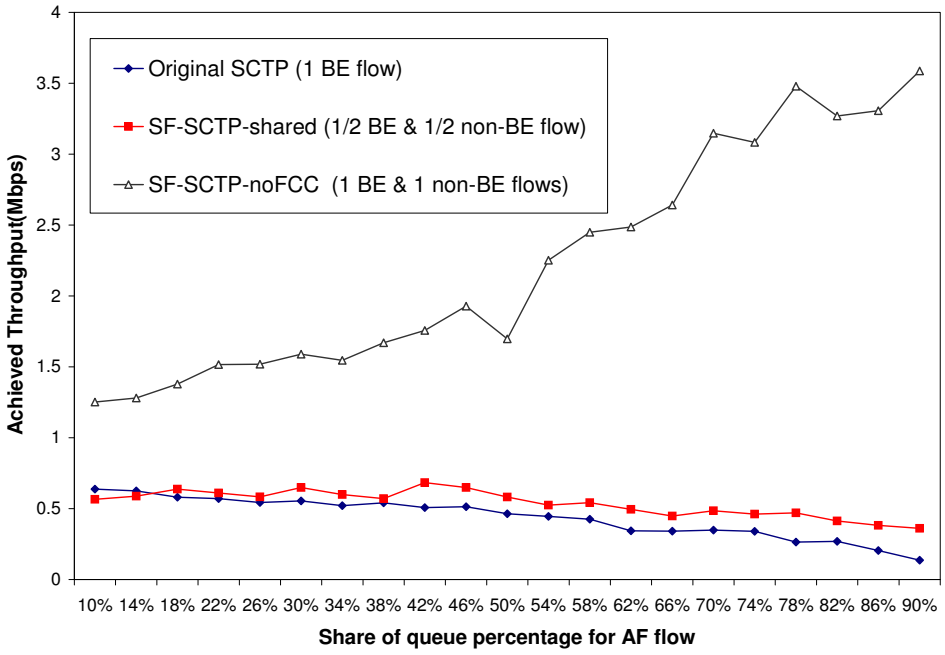


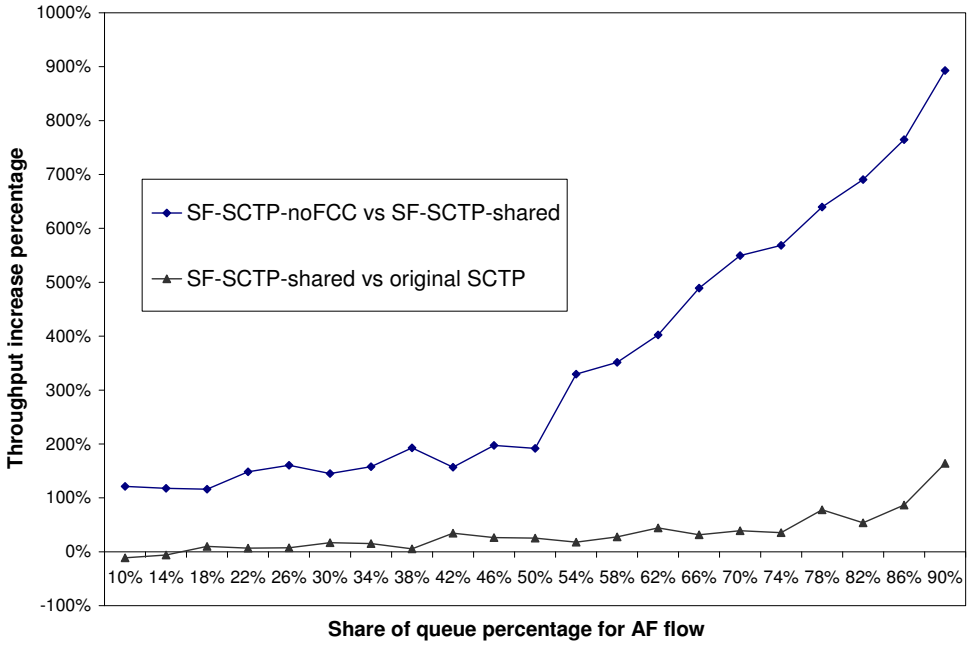
Figure 5.2: Experiment I: Throughput achieved by BE and non-BE flows for different setups.

SCTP that opens only a BE flow. The false sharing degrades the throughput of non-BE subflows in SF-SCTP-shared so severely that the benefits of DSCP marking are negligible.

- The non-BE subflow performs much better in SF-SCTP-noFCC than in SF-SCTP-shared because the performance of non-BE subflow in SF-SCTP-shared is dragged down by BE subflows due to falsely shared congestion information. When the queue size for non-BE flows is approaching 90%, we observe that the throughput of SF-SCTP-noFCC is almost 8 times higher than for SF-SCTP-shared. This is because the SF-SCTP-noFCC is able to take full advantage of the Diff-Serv network by avoiding the problem of false sharing.



(a) Throughput.



(b) Throughput increase.

Figure 5.3: Experiment I: Throughput comparison for different SCTP implementations.

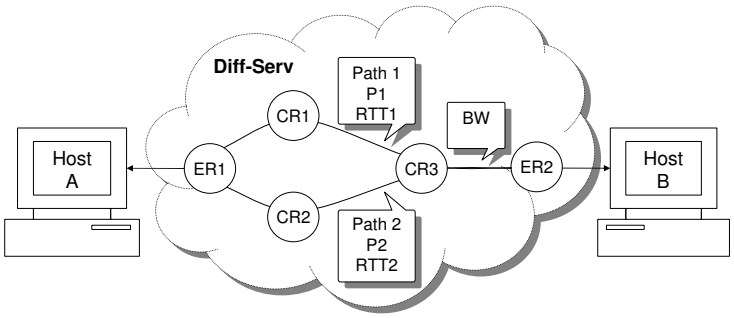


Figure 5.4: Simulation topology to investigate SF-SCTP's ability to support QoS

When the bandwidth share of the non-BE flows is 50%, false sharing is no longer a problem and the SF-SCTP-noFCC still achieves almost twice as much throughput as the original SCTP or SF-SCTP-shared because SF-SCTP-noFCC implements two sets of congestion control parameters while for the original SCTP and SF-SCTP-shared there is only one set. This makes the SF-SCTP-noFCC more aggressive in taking more bandwidth than the original SCTP and SF-SCTP-shared.

5.1.2 Simplified Diff-Serve Network

5.1.2.1 Simulation Setup

With the differentiated services provided by the Diff-Serv network shown in Fig. 5.25, two simulation experiments have been performed to study SF-SCTP's capability to support QoS among its streams. The simulation topology in Fig. 5.25 consists of Host A (sender), Host B (receiver), and a simplified Diff-Serv network between them. There may be one or multiple SCTP association(s) running concurrently between Hosts A and B. For simplicity, no delayed-SACK is used for all SCTP associations. The ER1 (Edge Router 1) in the Diff-

Serv network is able to map traffic coming from Host A to two different paths according to the marked DSCPs. Packets traveling through the two paths will experience different packet drop rates (namely, p_1 for path 1, p_2 for path 2), and RTTs (RTT_1 for path 1, RTT_2 for path 2). In our simulation setting, path 2 in Fig. 5.25 always has a higher priority than path 1 with regards to packet drop rate or RTT.

The Diff-Serv network is configured as a high speed network and has almost infinite resources available such that adding or removing an SCTP association shall not affect the two paths' packet drop rates and RTTs. It has been observed in our simulations that all packet losses in this network result in Fast Retransmits. Since every lost packet is recovered by a Fast Retransmit, we define the notion of Fast Retransmit rate that, from the SCTP sender's point of view, is an effective indication of the packet drop rate it experienced. Fast Retransmit rate is obtained as

$$\text{Fast Retransmit rate} = \frac{\text{Number of Fast Retranmits during the simulation}}{\text{Throughput} \times \text{Simulation duration}} \tag{5.2}$$

The Diff-Serv network configurations for the two simulation experiments are as follows:

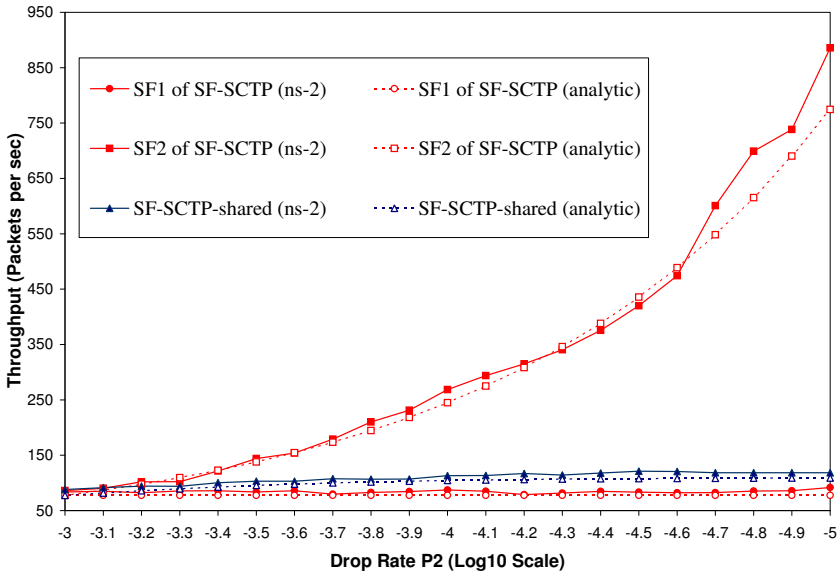
- *Experiment II (effect of different packet drop rates):* The two paths between Hosts A and B have the same RTTs ($RTT_1 = RTT_2 = 0.5\,\text{sec}$), but different packet drop rates. For each simulation, path 1 has a fixed packet drop rate $p_1 = 10^{-3}$, while p_2 of path 2 varies from 10^{-5} to 10^{-3}. Three instances of SCTP associations, original SCTP, SF-SCTP and SF-SCTP-shared, are established between Hosts A and B. The original SCTP, unaware of QoS, is been mapped to path 1, which has a higher packet drop rate of p_1. Both

the SF-SCTP and SF-SCTP-shared associations contain two subflows, SF_1 is assigned to path 1 and SF_2 to path 2. For SF-SCTP-shared, where the subflows share flow and congestion control, the WRR scheduler apportions each subflow with 50% of the achieved throughput. Since both paths have the same RTTs, packets coming from the two subflows of SF-SCTP-shared are delivered in order at the SCTP receiver (Host B) even though they travel through different paths. This ordered delivery effectively avoids unnecessary Fast Retransmits.

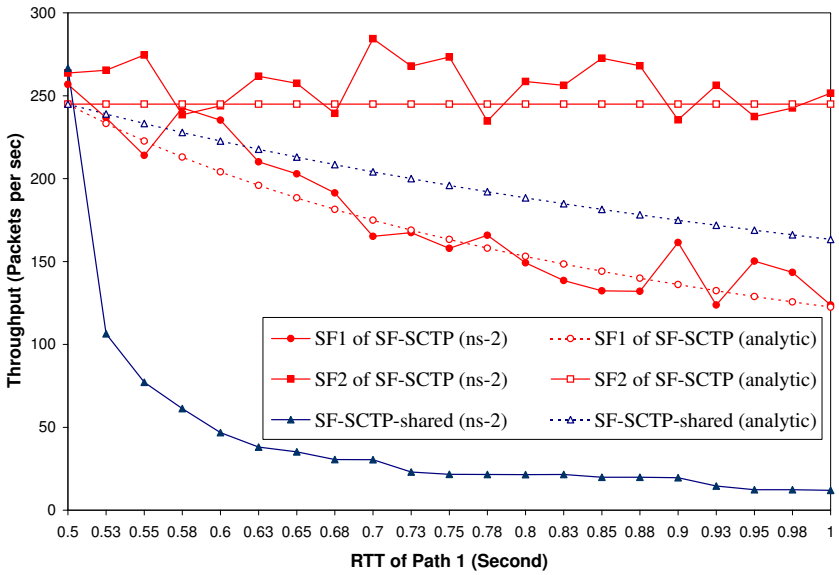
- *Experiment III (effect of different RTTs):* The packet drop rates for both paths are fixed at $p_1 = p_2 = 10^{-4}$. The differentiated services of paths 1 and 2 are provided by the different RTTs, RTT_2 is fixed at 0.5 sec, while RTT_1 is varying between 0.5 sec to 1 sec. The other network settings are the same as Experiment II.

5.1.2.2 Discussion of Simulation Results

Experiments II and III are designed to study the throughput differences of original SCTP, SF-SCTP and SF-SCTP-shared under different packet drop rates or RTTs. The simulation results and the corresponding analytic expectations for those results are presented in Figs. 5.5, 5.6 and 5.7. Fig. 5.5 illustrates the flow throughput differences of SF-SCTP-shared with SF_1 and SF_2 of SF-SCTP. Fig. 5.6 compares the association throughput, while Fig. 5.7 compares the Fast Retransmit rates of the original SCTP, SF-SCTP and SF-SCTP-shared.

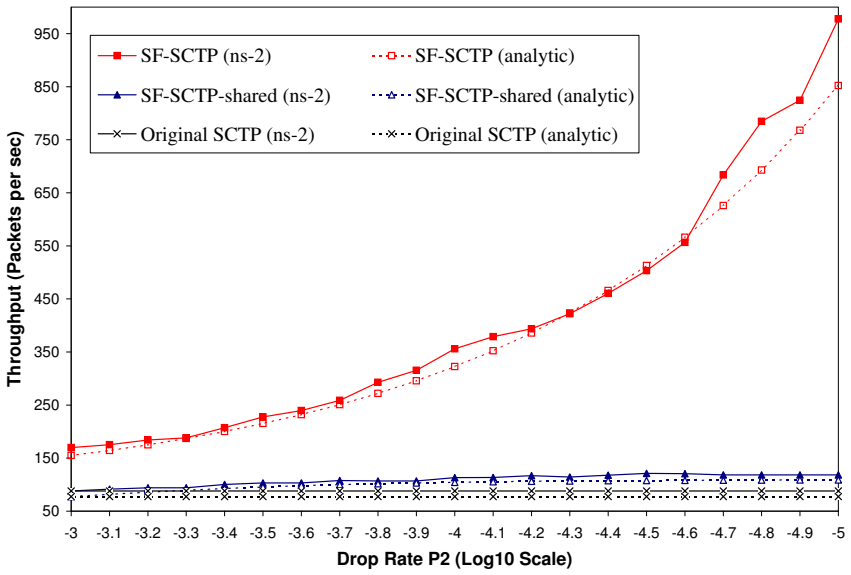


(a) Experiment II.

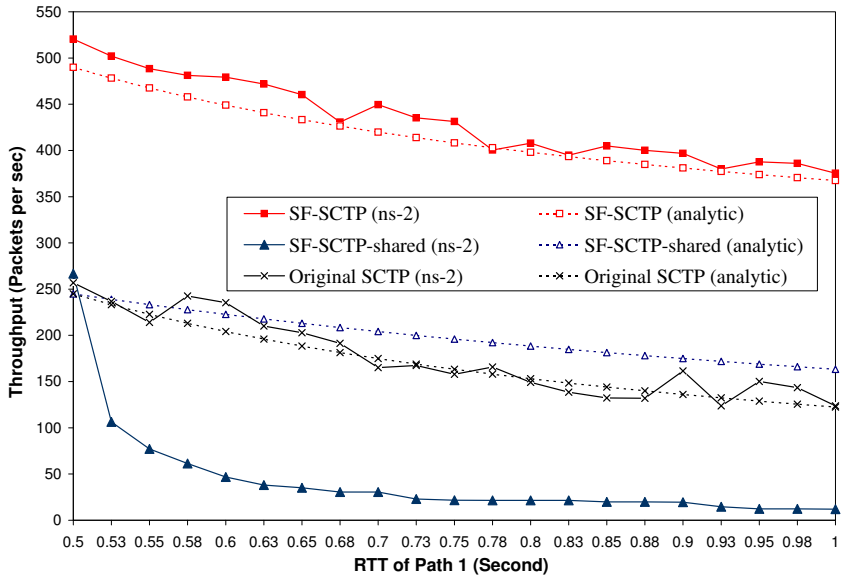


(b) Experiment III.

Figure 5.5: Flow throughput for SF-SCTP-shared and SF_1, SF_2 of SF-SCTP.

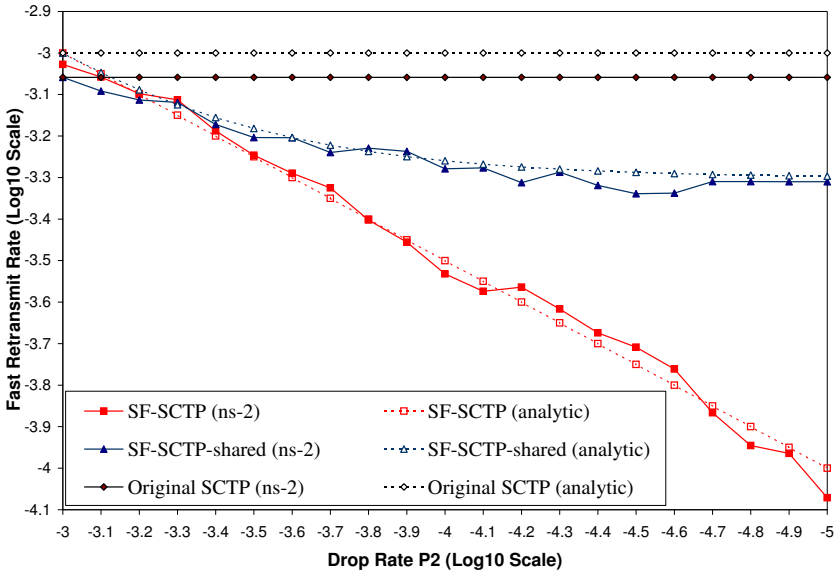


(a) Experiment II.

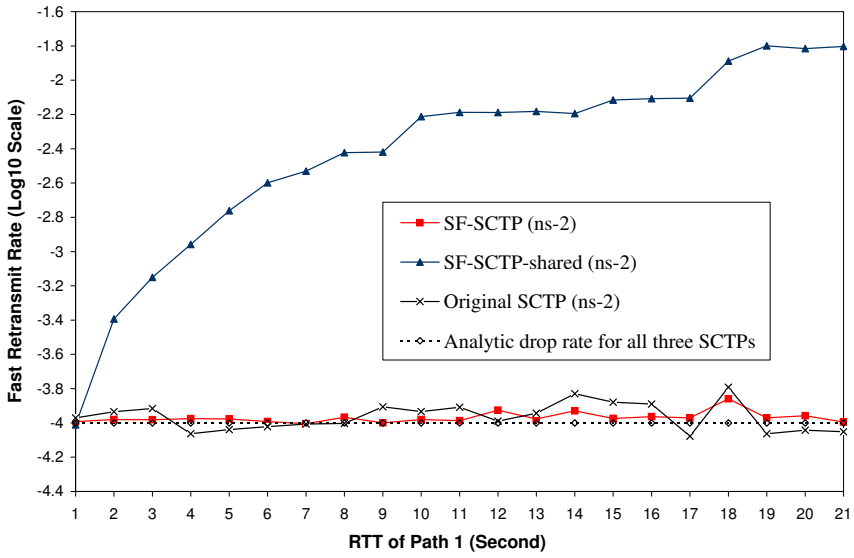


(b) Experiment III.

Figure 5.6: Association throughput for the original SCTP, SF-SCTP and SF-SCTP-shared.



(a) Experiment II.



(b) Experiment III.

Figure 5.7: Fast Retransmit rate comparison for the original SCTP, SF-SCTP and SF-SCTP-shared.

The results in Fig. 5.5 confirm the capability of SF-SCTP to support preferential treatment among its subflows. Since all subflows in SF-SCTP have their own flow and congestion control, each subflow can adapt to the dynamic change of network characteristics without affecting the others. When the two paths have the same drop rates and RTTs, we observe from Figs. 5.5(a) and 5.27(a) that SF_1 and SF_2 of SF-SCTP achieve approximately the same throughput. As the Diff-Serv network starts to provide differentiated services among the two paths by either reducing the packet drop rate of path 2 or increasing the RTT of path 1, SF_2 begins to outperform SF_1. From Fig. 5.5(a), we can tell that the throughput of SF_2 increases while the throughput of SF_1 remains relatively constant when the drop rate of path 2 reduces. SF_2 gets roughly 10 times the throughput as SF_1 when $p_2 = 10^{-2} * p_1 = 10^{-5}$. From Fig. 5.27(a), we observe that, as the RTT of path 1 increases, the throughput of SF_1 decreases while the throughput of SF_2 remains almost unchanged. SF_2 obtains about twice the throughput of SF_1 when $RTT_1 = 2 * RTT_2 = 1\,\text{sec}$.

The comparison of SF-SCTP with SF-SCTP-shared in Figs. 5.5, 5.6 and 5.7 tells us SF-SCTP's ability to support QoS partially lies in the fact that it avoids false sharing by implementing separate flow and congestion control for each subflow. From Fig. 5.7(a) of Experiment II, where both paths have the same RTTs, we see that SF-SCTP-shared's Fast Retransmit Rate is close to the average packet drop rates of SF_1 and SF_2 of SF-SCTP. This observation confirms our analytic models, which imply that, for SF-SCTP-shared,

we could use the subflows' weighted average packet drop rate as the equivalent packet drop rate when all subflows in SF-SCTP-shared have the same RTTs. In Experiment III where the two paths have different RTTs, we see that SF-SCTP-shared's Fast Retransmit Rate is much higher than the subflows' weighted average packet drop rate (Fig. 5.7(b)). The unequal RTTs and the falsely shared congestion control in SF-SCTP-shared lead to the unnecessary occurrences of Fast Retransmits. This severe degree of false sharing could cause SF-SCTP-shared's throughput to be much lower than the analytic expectation, as shown in Figs. 5.27(a) and 5.27(b).

Except for SF-SCTP-shared, of which we can only predict the throughput upper limit, we observe in Figs. 5.5, 5.6, and 5.7 that our analytic models accurately reflect the congestion behavior of the original SCTP and SF-SCTP. We derive the analytic models based on a few idealized assumptions [73, 74, 80]. Also our models do not capture SCTP's time-outs or consecutive Fast Retransmits events. The experiments conducted under the simplified Diff-Serv network, where no occurrences of time-out or consecutive Fast Retransmits have been observed, confirm the accuracy of our analytic models and justify the assumptions made during the derivation procedure.

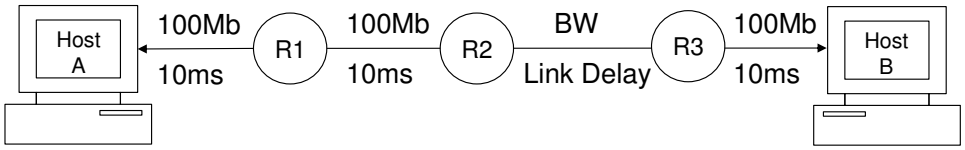


Figure 5.8: Simulation topology to investigate SF-SCTP behavior in a single bottleneck network.

5.2 SF-SCTP Behavior in Single Bottleneck Network

5.2.1 Simulation Setup and Results

All the experiments reported in this section involve the network with a single bottleneck link as depicted in Fig. 5.8. This bottleneck network consists of two SCTP hosts (A and B), three routers (R1, R2 and R3), and a number of links. Except for the bottleneck link, the bandwidth and delay of the other links are shown in Fig. 5.8. The bottleneck link between R2 and R3 is used to constrain the network capacity and is the place where all packet drops occur. The simulation network does not provide any type of differentiated services. Packets from any subflow of SF-SCTP, all marked with a DSCP value of zero, are being treated equally in competing for the limited bandwidth of the bottleneck link. In this and the next sections, we conduct nine experiments to examine how SF-SCTP exploits available resources under various network conditions. All simulation experiments were run for 1200 seconds; we collected statistics after 400 seconds in order to avoid data from the transient state.

In this section, we will discuss five simulation experiments with slightly different network

Table 5.3: Network configurations for Experiment IV to VIII (Exp. = Experiment, pkts = packets).

	# of SFs	QueueType	QueueSize(pkts)	LinkBW(Mbps)	LinkDelay(ms)
Exp. IV	1,2,3...12	Drop-Tail	100	10	20
Exp. V	1 or 4	Drop-Tail	10,20,30...150	10	20
Exp. VI	1 or 4	Drop-Tail	100	1,2,3...30	20
Exp. VII	1 or 4	Drop-Tail	100	10	10,15,20...70
Exp. VIII	1,2,3...12	RED	100	10	20

settings. The configurations for the bottleneck link bandwidth, link delay, queue mechanism and queue size are illustrated in Table 5.3. For each experiment, the number of subflows of the SF-SCTP association, running between Hosts A and B, are also given in Table 5.3. In Experiment IV, RED queue management rather than Drop-Tail is implemented between R2 and R3. The minimum threshold of the RED queue is 25 packets, the maximum threshold is 75 packets, while the maximum drop rate is set as 5%.

To examine the behavior of SF-SCTP under different network configurations, a special instance has been chosen in each simulation experiment. For each chosen instance, we sample SF-SCTP's *cwnd* size, transmission rate, and the bottleneck link queue occupancy at an interval of 0.1 sec. To study how the behavior of SF-SCTP with multiple subflows

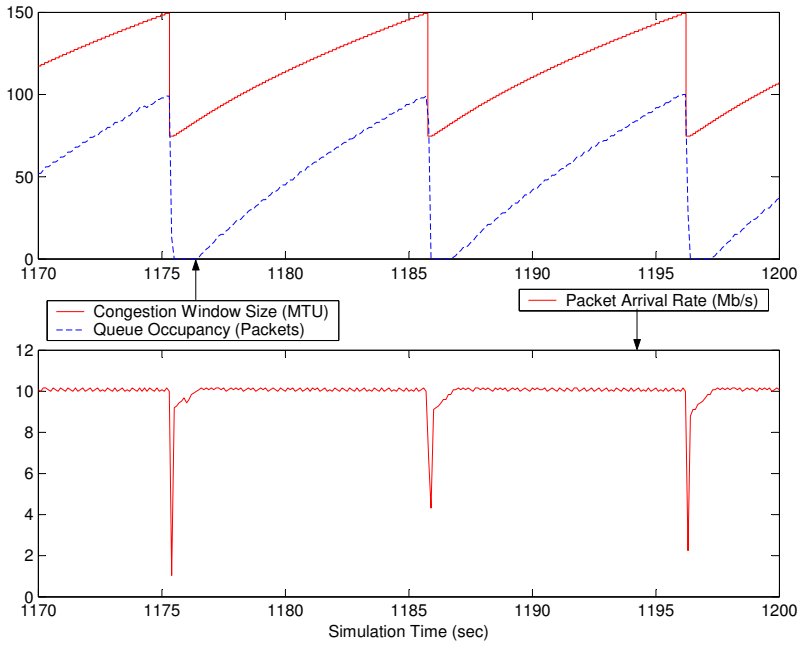
is different from the SF-SCTP with only one subflow, we let the number of subflows in SF-SCTP to be 1 or 4 in each instance, respectively. The results between 1170 sec and 1200 sec for the chosen instances are depicted in Figs. 5.9, 5.10, 5.11, 5.12, and 5.13.

The network utilization and packet drop rates for Experiments IV to VIII are summarized in Figs. 5.14, 5.15, 5.16, 5.17 and 5.18.

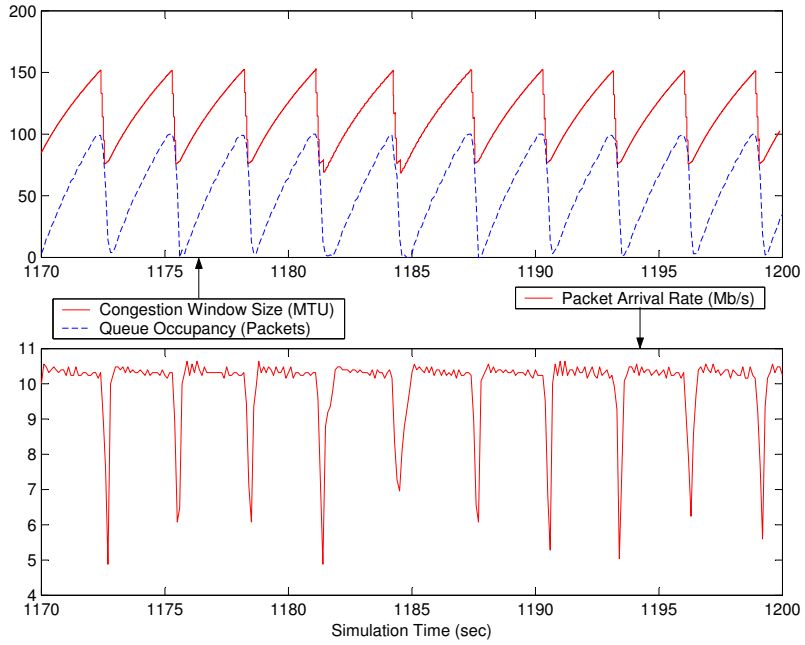
5.2.2 Periodicity of SF-SCTP

From the simulation results for Experiments IV to VIII (Figs. 5.9, 5.10, 5.11, and 5.12), we observe that SF-SCTP's *cwnd*, transmission rate, and the bottleneck queue occupancy demonstrate a strong trend of periodicity. Fig. 5.13 shows the results of Experiment IV, which has the Drop-Tail buffering strategy replaced by the RED mechanism. Since RED queue can randomly drop packets before the queue is full, this introduces a certain amount of randomness into the simulation results. However, as shown in Fig. 5.13, this additional randomness is not strong enough to significantly alter the periodic fluctuations of SF-SCTP.

At any time during an SF-SCTP association, the bottleneck queue is in one of the following states: Queue Buildup (QBU), Queue Draining (QDN), or Queue Steady (QSD) [56]. The state diagram of Fig. 4.4 shows the transitions of those three states. A *queue cycle* is defined such that it starts from the beginning of a QSD state and ends the next time when the state is back into the QSD state again.

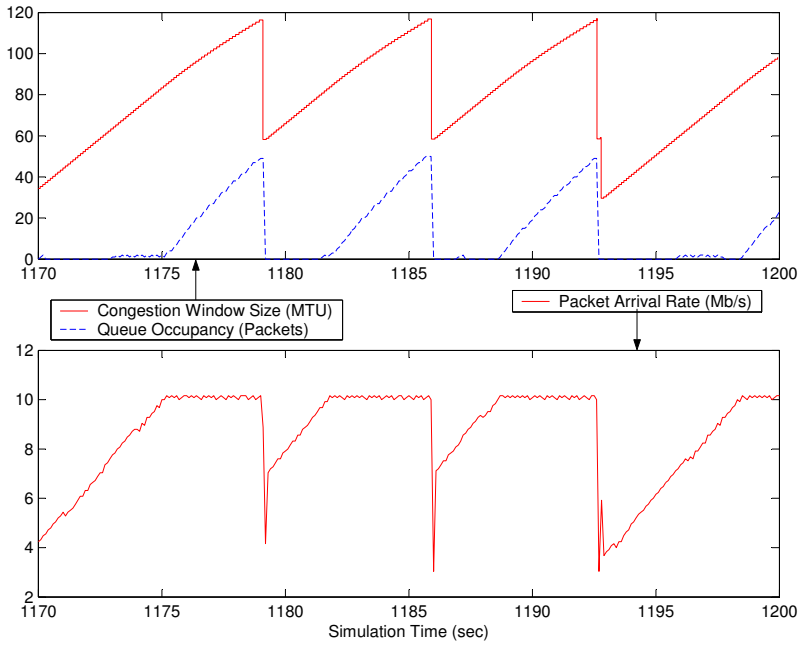


(a) SF-SCTP with one subflow.

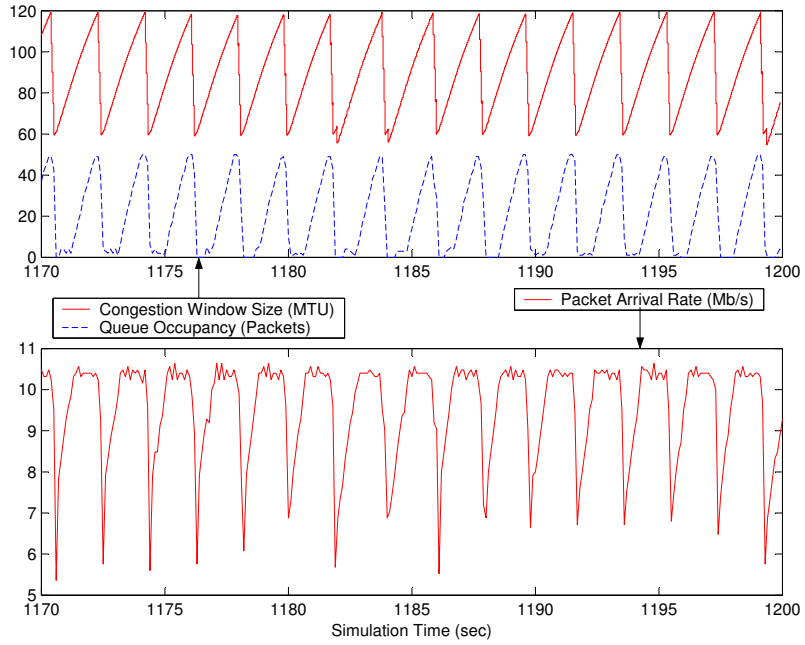


(b) SF-SCTP with four subflows.

Figure 5.9: Experiment IV: SF-SCTP's Congestion Window, transmission rate and bottleneck link's queue occupancy (bottleneck queue size = 100 pkts, bandwidth = 10Mbps, link delay = 20ms).

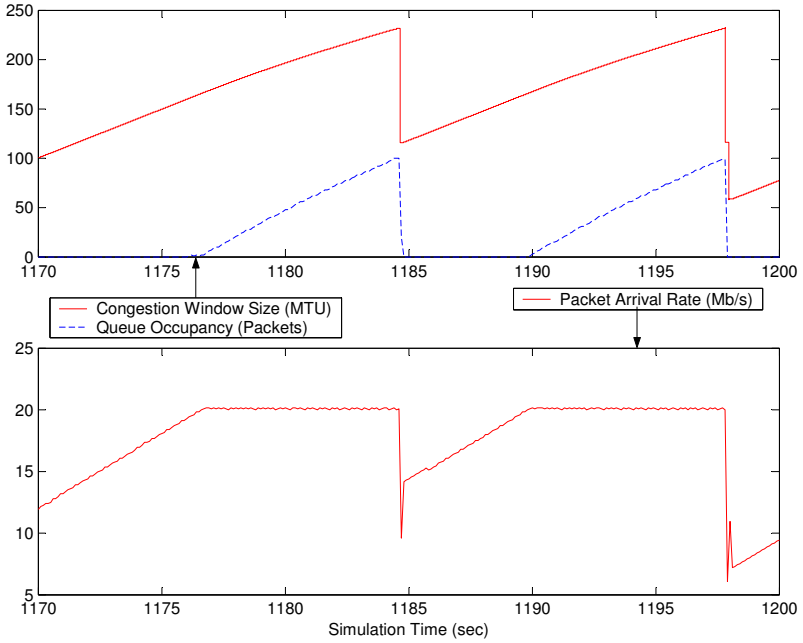


(a) SF-SCTP with one subflow.

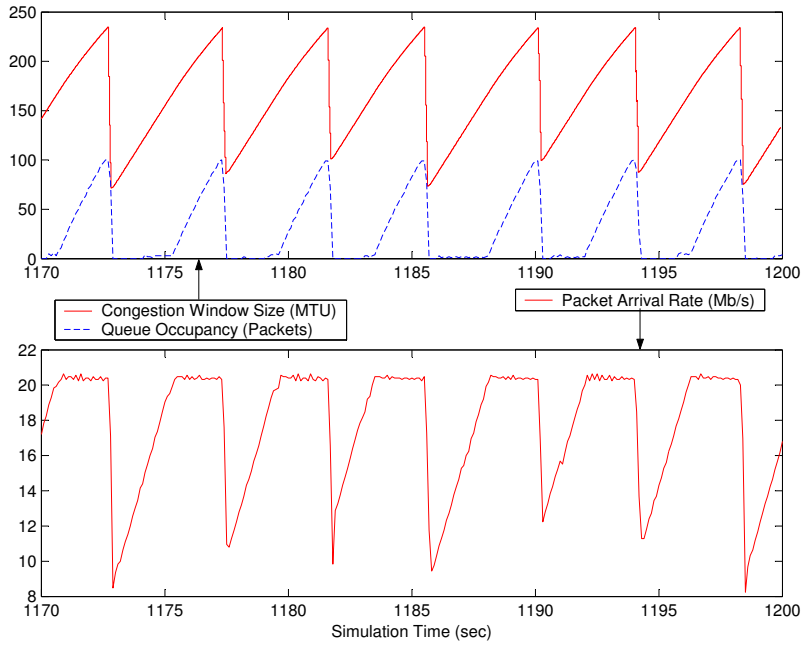


(b) SF-SCTP with four subflows.

Figure 5.10: Experiment V: The queue size is 50 packets (100 packets in Experiment IV).

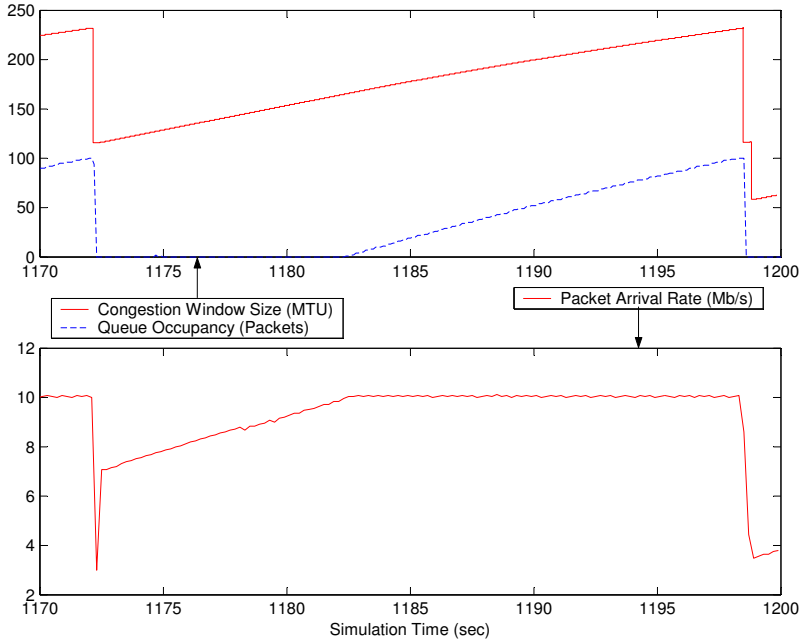


(a) SF-SCTP with one subflow.

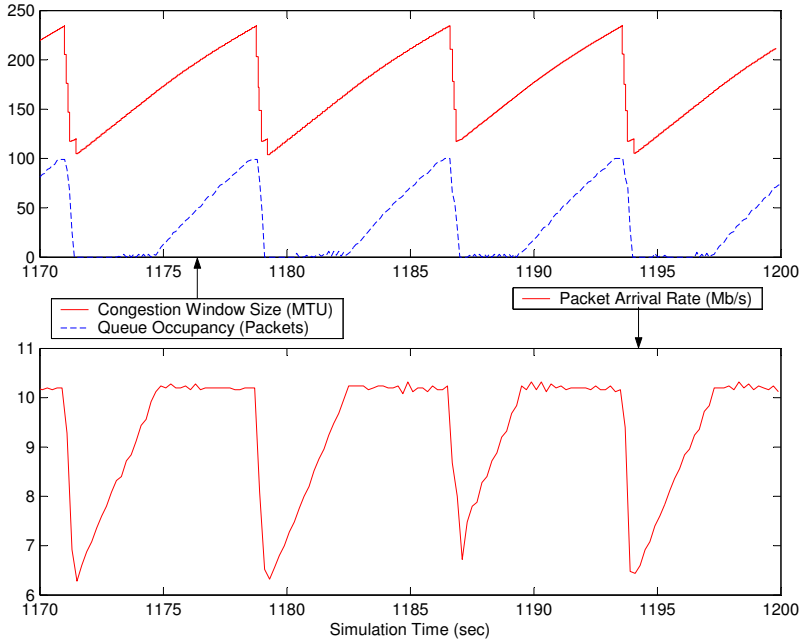


(b) SF-SCTP with four subflows.

Figure 5.11: Experiment VI: The bottleneck bandwidth is 20Mbps (10Mbps in Experiment IV).

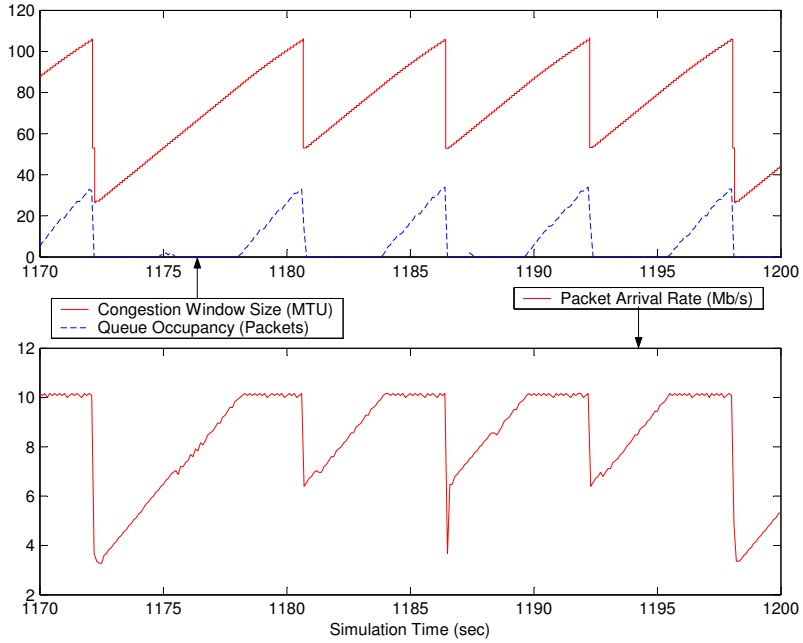


(a) SF-SCTP with one subflow.

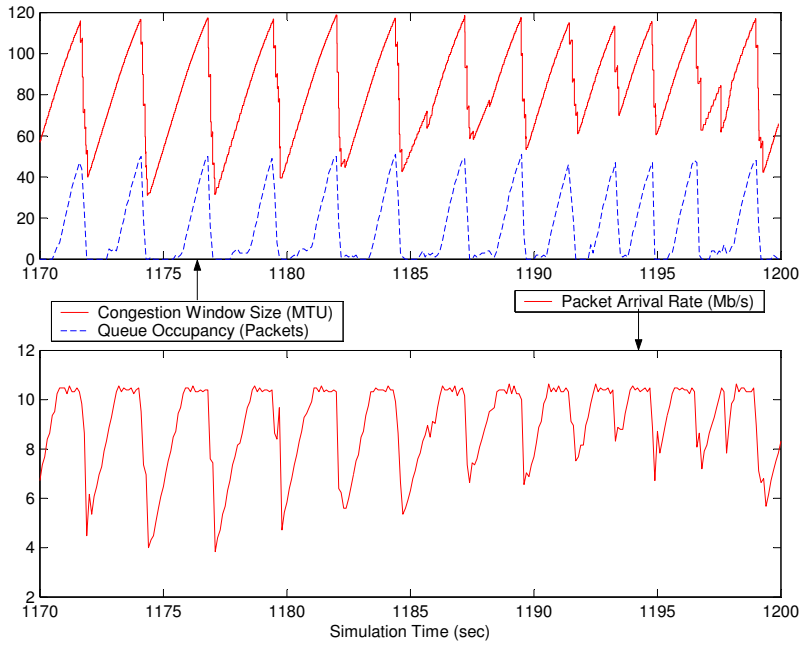


(b) SF-SCTP with four subflows.

Figure 5.12: Experiment VII: The bottleneck link delay is 70ms (20ms in Experiment IV).

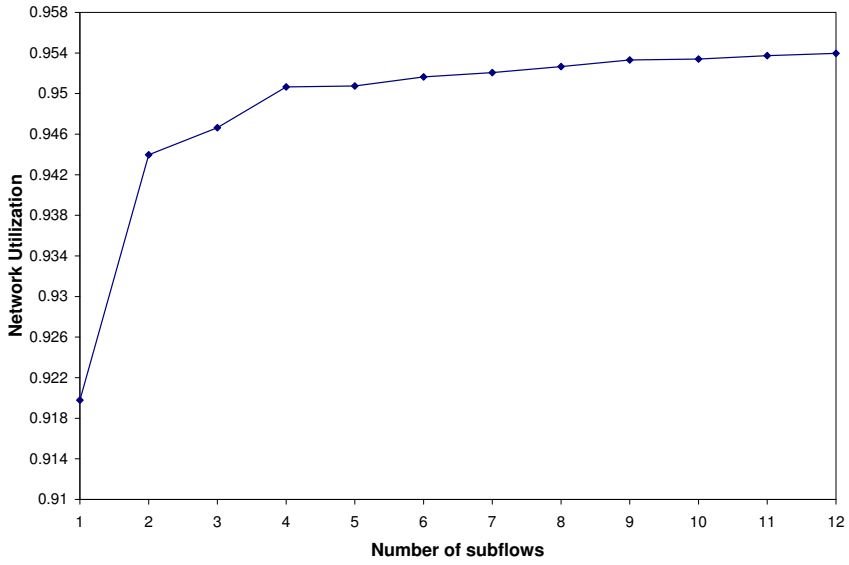


(a) SF-SCTP with one subflow.

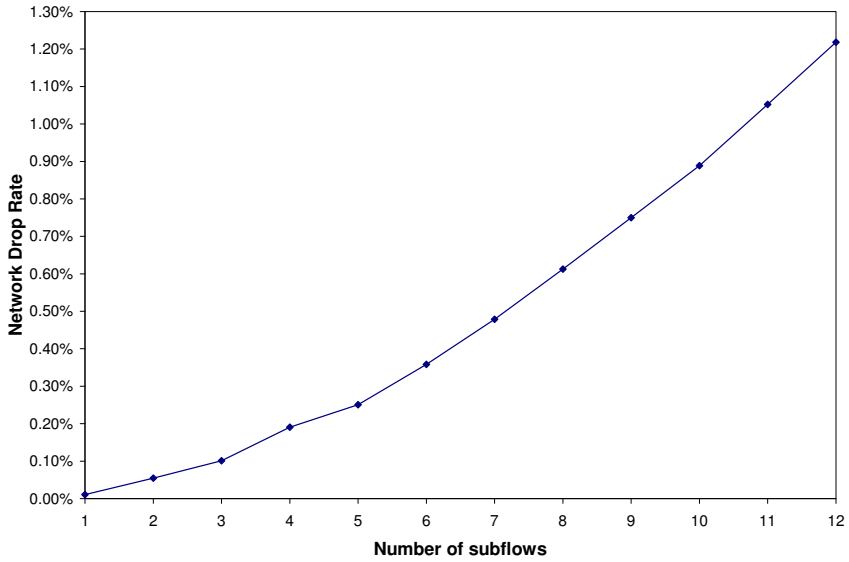


(b) SF-SCTP with four subflows.

Figure 5.13: Experiment VIII: RED queue is used (Drop-Tail queue used in Experiment IV).

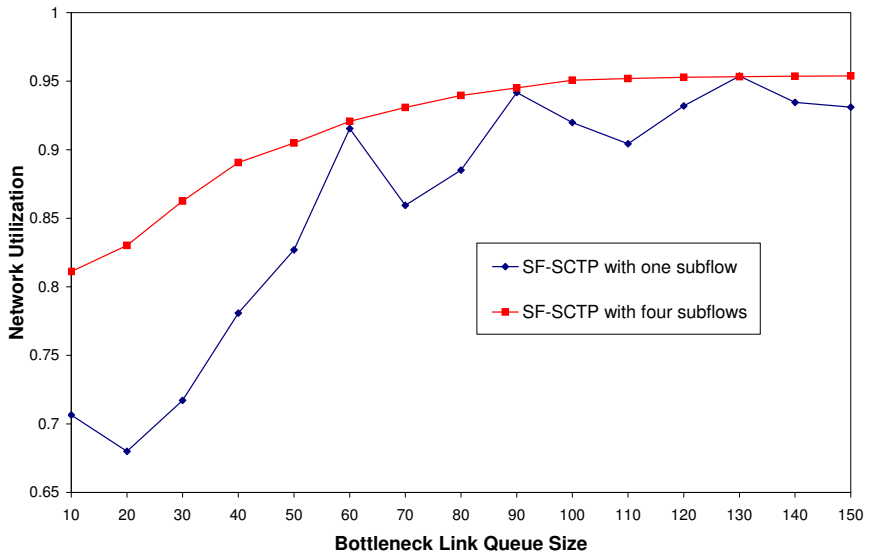


(a) Network utilization.

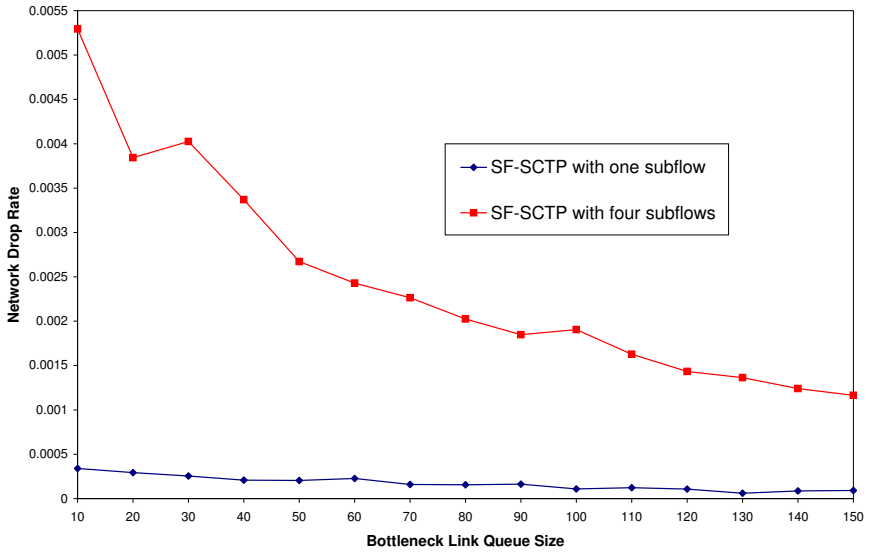


(b) Network drop rate.

Figure 5.14: Experiment IV: Number of subflows changing from 1 to 12.

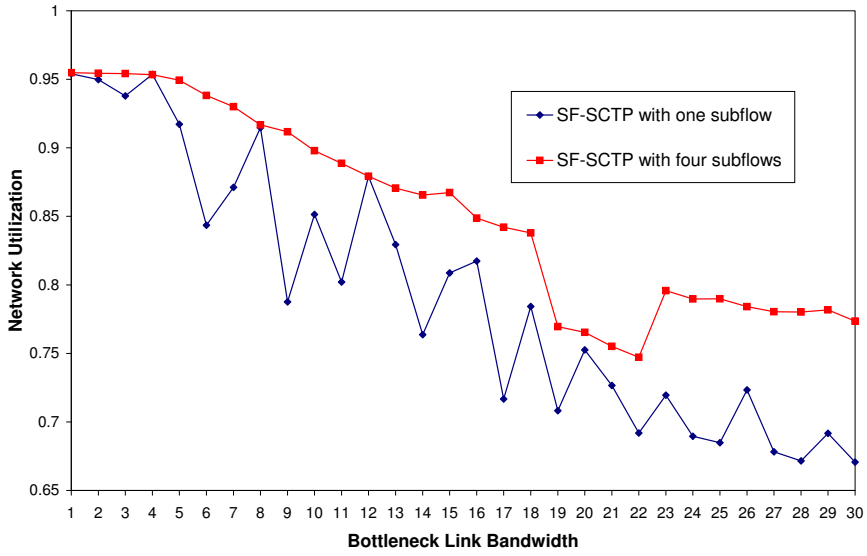


(a) Network utilization.

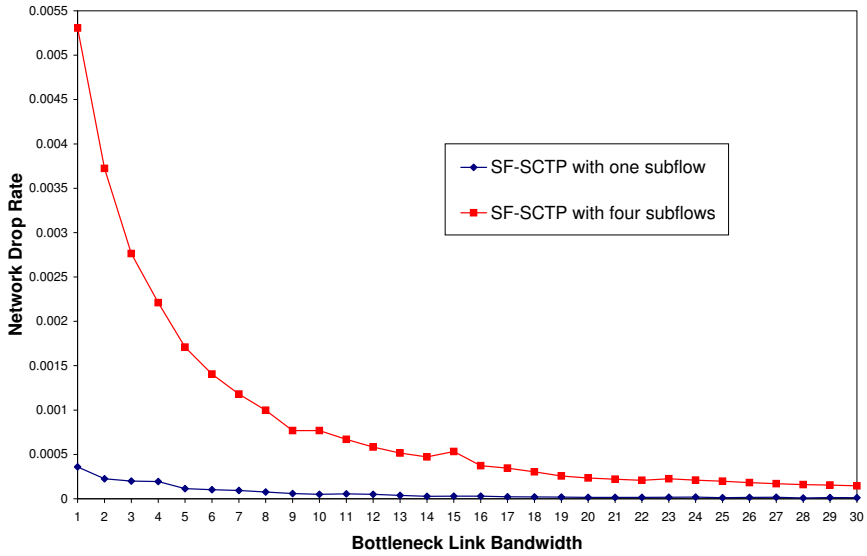


(b) Network drop rate.

Figure 5.15: Experiment V: Bottleneck link queue size changing from 10 to 150.

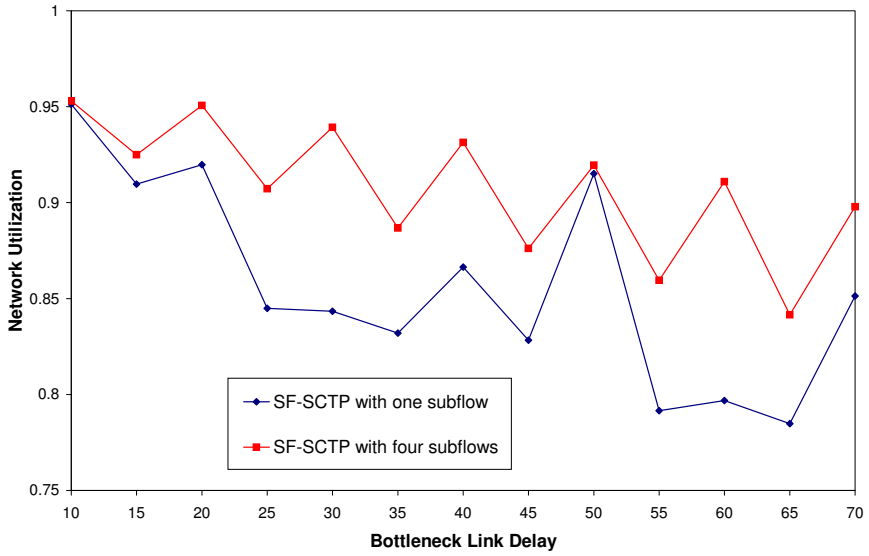


(a) Network utilization.

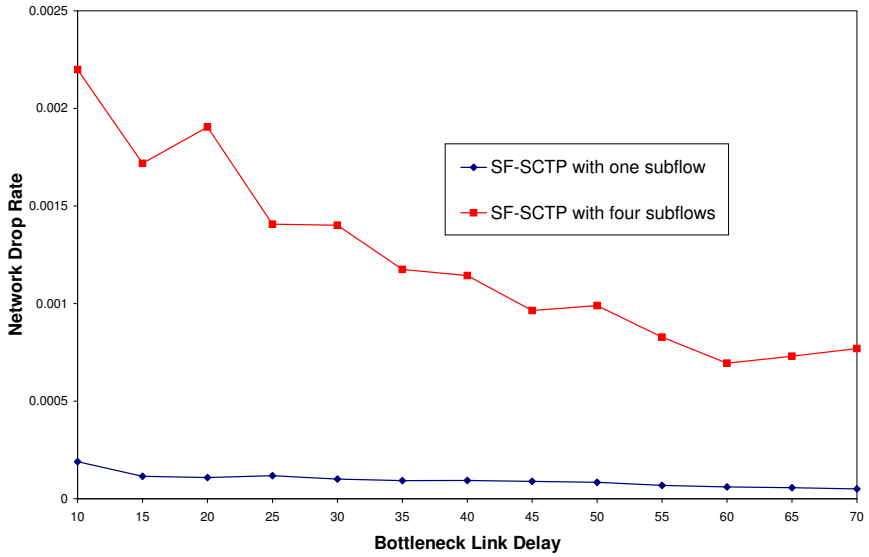


(b) Network drop rate.

Figure 5.16: Experiment VI: Bottleneck link bandwidth changing from 1Mbps to 30Mbps.

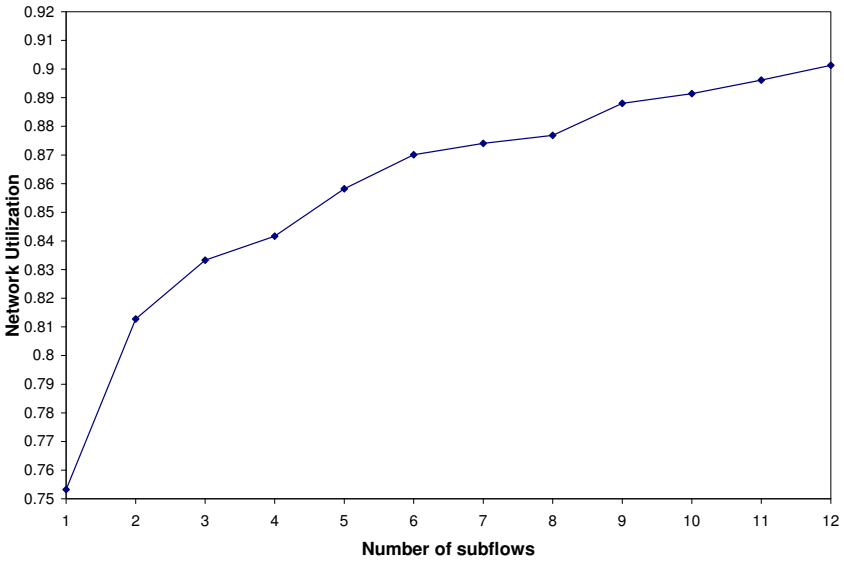


(a) Network utilization.

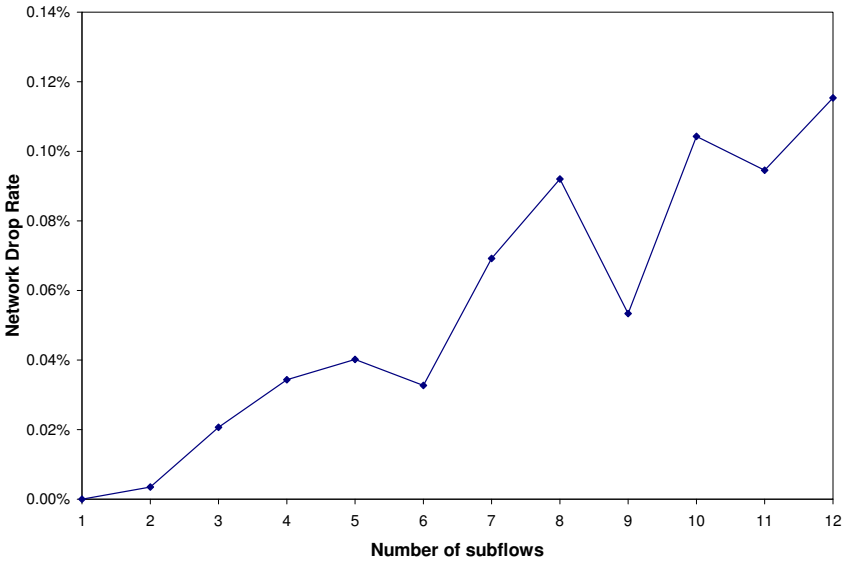


(b) Network drop rate.

Figure 5.17: Experiment VII: Bottleneck link delay changing from 10ms to 70ms.

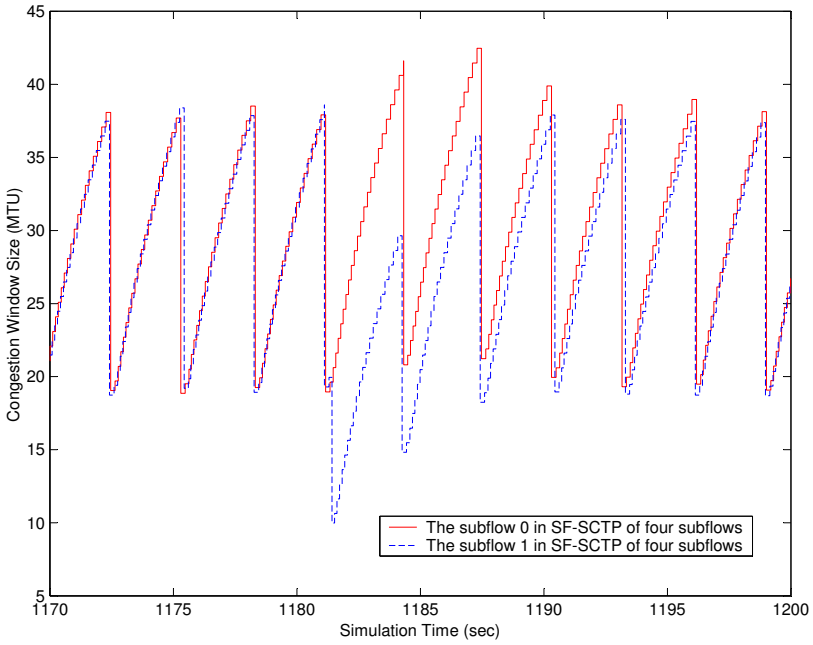


(a) Network utilization.

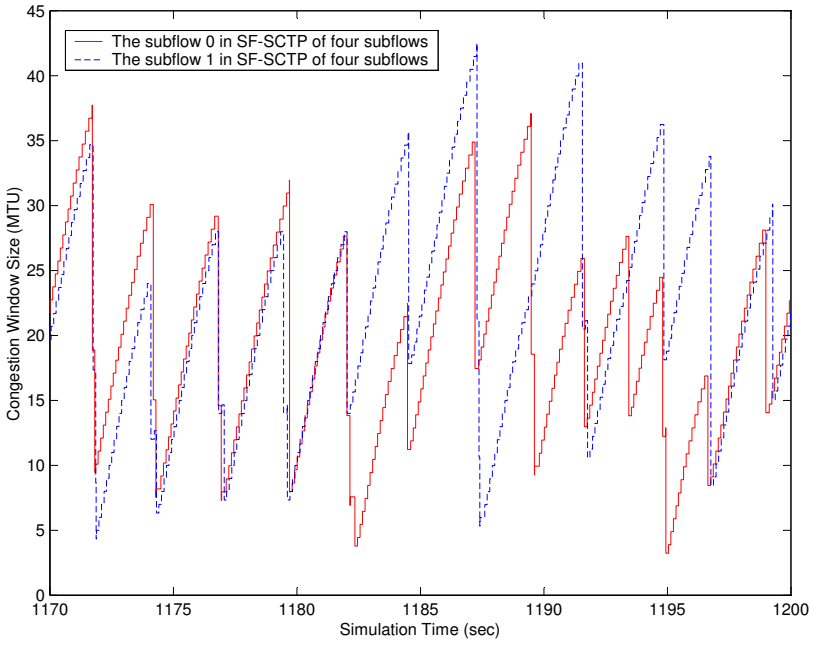


(b) Network drop rate.

Figure 5.18: Experiment VIII: RED is used with number of subflows changing from 1 to 12.



(a) Experiment IV, Drop-Tail queue.



(b) Experiment VIII, RED queue.

Figure 5.19: The individual Congestion Window for subflows 0 and 1 in SF-SCTP.

From Figs. 5.9, 5.10, 5.11, 5.12, and 5.13(a), we do not observe the occurrences of queue state transition from the QDN to the QBU state. In those simulation scenarios, SF-SCTP's transmission rate is always reduced to half or even less upon the detection of packet loss(es). Before the transmission rate recovers to a value above the bottleneck bandwidth, the bottleneck queue has enough time to deplete the remaining packets. The global synchronization [29] problem of Drop-Tail queue largely offsets SF-SCTP's benefit of greater resistance to packet losses when it has multiple subflows. When the Drop-Tail queue starts to drop packets, we observe all subflows of SF-SCTP are affected at about the same time such that the overall transmission rate of SF-SCTP is reduced to half or even less. In Fig. 5.13(b), where RED queue is used and SF-SCTP has four subflows, we observe the transition from the QDN to the QBU state. RED queue can drop a packet earlier before the queue is full. Since those earlier packet drops may only affect a subset of subflows in SF-SCTP, the transmission rate of SF-SCTP may reduce less than half. Thus, the queue may not have enough time to empty its contents before the transmission rate is recovered.

Comparing Figs. 5.9 and 5.10, we find that Fig. 5.10 has a shorter QBU period while a longer QSD period, and altogether a shorter queue cycle. It is reasonable that a shorter bottleneck queue size results in a shortened QBU period because it takes less time to fill up the queue. On the other hand, the reason that Experiment V has a longer QSD period is not so obvious and can only be explained by the queue delay introduced.

As what can be seen from the simulation results for Drop-Tail queue (Figs. 5.9, 5.10, 5.11 and 5.12), the *cwnd* reaches its highest value when the queue is full. Considering that transmission rate is almost fixed at a rate slightly above the bottleneck bandwidth during the QBU period, a longer queue, which can cause more delay (and thus prolong RTT), will have a larger *cwnd* value at the end of the QBU period. From Fig. 5.9 we can see the highest *cwnd* value is 150, while in Fig. 5.10 it is less than 120. After a transitory QDN period that empties the queue, the RTT will roll back to 0.1sec for both Experiments IV and V. Therefore, a higher *cwnd* value at the end of the QBU state definitely means a greater transmission rate at the beginning of the QSD state. The normal starting transmission rate at the beginning of the QSD period in Fig. 5.9 is approximately 9Mpbs, while in Fig. 5.10 it is 7Mbps. In short, a queue of larger size can better preserve SF-SCTP's transmission rate after a Fast Retransmit. This explains that Experiment V has longer QSD period than Experiment IV because it takes more time for the transmission rate in Fig. 5.10 to recover.

The network utilization and drop rate for Experiment V shown in Fig. 5.15 further support our comparison analysis between Figs. 5.9 and 5.10. To improve network utilization, one wants to prolong the QBU period, while shortening the QSD period because SF-SCTP wastes bandwidth in the QSD state. To reduce the packet drop rate, one wants to prolong the average queue cycle. Since each queue cycle is associated with one or more packet drops, the smaller number of queue cycles within a unit time necessarily means a lower

packet drop rate. From Fig. 5.15, we see the clear trend that, as the queue size increases, the network utilization improves; at the same time, packet drop rate is reduced.

Figs. 5.9 and 5.11 show that Experiment VI, which has higher bottleneck bandwidth than Experiment IV, achieves longer QSD periods and queue cycles. Although Experiments IV and VI have the same queue size, the queue in Experiment VI causes less maximum delay since Experiment VI has the higher bandwidth. A smaller queue delay before a Fast Retransmit means a more dramatic change in the transmission rate of SF-SCTP, which prolongs the QSD period. Fig. 5.16 shows that, as the bottleneck bandwidth increases in Experiment VI, both the network utilization and drop rate are reduced.

We find out from Figs. 5.9 and 5.12 that a larger bottleneck link delay, which leads to a higher RTT, can prolong both the QBU and QSD periods. Compared with Fig. 5.9, the *cwnd* in Fig. 5.12 has the smaller growth rate, but the larger peak value. Thus, it takes longer for the *cwnd* in Experiment VII to recover after a Fast Retransmit. Also, after the RTT increases, the queue delay has a smaller percentage share in the RTT. In Experiment VII, when the queue is full, it occupies only 28.6% of the total RTT, while the corresponding number in Experiment IV is 44.4%. If the queue delay has the smaller percentage share in the RTT, it has less capability to preserve the transmission rate. Therefore, it takes longer for the transmission rate in Experiment VII to recover. From Fig. 5.17, we observe that both network utilization and packet drop rate are reduced as the link delay increases because of the less effectiveness of queue delay and the prolonged

queue cycles.

We observe from Figs. 5.9 and 5.13 that RED buffering mechanism reduces the queue utilization when configured as in our simulation; consequently, it reduces the network utilization as shown in Fig. 5.18(a). Due to earlier packet drops, the queue has never had the chance to reach its full utilization of 100pkts (Fig. 5.13). On the other side, RED queue does demonstrate its advantage in reducing network drop rate. While comparing Figs. 5.18(b) and 5.14(b), it can been seen that, for SF-SCTP with 12 subflows, the RED queue drop rate is 10 times lower than that of the Drop-Tail queue.

5.2.3 Effect of Number of Subflows

By comparing Figs. 5.9(a), 5.10(a), 5.11(a), 5.12(a), and 5.13(a), where the SF-SCTP has only one subflow, with Figs. 5.9(b), 5.10(b), 5.11(b), 5.12(b), and 5.13(b), where SF-SCTP has four subflows, we observe the QBU and QSD periods of the former are approximately four times longer than the latter. SF-SCTP with more subflows has a faster overall *cwnd* growth rate such that it takes less time to recover the *cwnd* and transmission rate, and to fully fill up the bottleneck queue.

From Figs. 5.14, 5.15, 5.16, 5.17, and 5.18, we observe that SF-SCTP with multiple subflows increases the network utilization at the cost of a high packet drop rate. Following the discussion of SF-SCTP's periodicity property in Section 5.2.2, we know that, by shortening the QSD period while prolonging the QBU period, the network utilization

could be improved. Unfortunately, when increasing the number of subflows, we find out that both the QSD and QBU periods are shortened in approximately the same scale. This observation informs us that the higher network utilization of SF-SCTP with four subflows does not directly result from the greater *cwnd* growth rate. On the other hand, the shortened QSD and QBU periods mean that, within a certain amount of time, SF-SCTP with four subflows will have more queue cycles, indicating a higher packet drop rate. Therefore, the greater *cwnd* growth rate of SF-SCTP with multiple subflows does not necessarily improve the network utilization but surely brings the negative effect of a high packet drop rate.

A careful comparison of Figs. 5.9(b), 5.10(b), 5.11(b), 5.12(b), 5.13(b) with Figs. 5.9(a), 5.10(a), 5.11(a), 5.12(a), 5.13(a) shows that SF-SCTP with four subflows has a smoother queue occupancy curve than SF-SCTP with one subflow. This is due to the asynchronous congestion behavior of the subflows in SF-SCTP. Since the congestion control of subflows in SF-SCTP is independent, different subflows may undergo different congestion states (i.e., slow start, congestion avoidance) at the same time. However, when subflows travel through the same Drop-Tail queue that becomes full, packets from all subflows are typically dropped, with all subflows responding with Fast Retransmit(s). In this way, the well-known global synchronization effect of the Drop-Tail queue tends to make all subflows synchronized. For example, in Fig. 5.19(a) of Experiment IV, SF_0 and SF_1 of SF-SCTP with four subflows are nearly, but not perfectly, synchronized. This imperfection smooths

the overall *cwnd* behavior of SF-SCTP; it results from the fact that packets are dropped in order and subflows may not respond to the packet drops at exactly the same time. The smoother overall *cwnd* behavior of SF-SCTP with four subflows leads to a smoother queue occupancy curve, which helps improve network utilization.

In Experiment IV, we observe in Fig. 5.13(b) that sometimes the *cwnd* of SF-SCTP with four subflows only reduces to 3/4 of its peak value, which significantly shortens the QSD period and helps improve the network utilization. Since the RED queue mechanism partially alleviates the global synchronization problem, subflows may have a higher degree of asynchronism than under the Drop-Tail queue (Fig. 5.19). We conclude that SF-SCTP with multiple subflows is more effective in improving utilization of a network implementing RED queue rather than Drop-Tail queue. The comparison of Fig. 5.14 with Fig. 5.18 confirms this observation. In Experiment IV with Drop-Tail queue, the network utilization is improved 3.4% from 92.0% to 95.4% when the number of subflows changes from 1 to 12; in Experiment VIII, the network utilization is improved 21.8% from 73.3% to 95.1%. With the number of subflows in SF-SCTP increasing, RED queue can achieve roughly the same network utilization as Drop-Tail queue, while keeping a relatively smaller packet drop rate.

5.3 SF-SCTP Behavior Under Various Background Traffic

5.3.1 Simulation Setup and Results

The same network topology as shown in Fig. 5.8 is used to investigate SF-SCTP's behavior under various types of background traffic. The bottleneck queue employs the Drop-Tail buffering mechanism with a queue size of 100 packets. The bottleneck bandwidth is 10Mbps and the link delay is 20ms. We run one SF-SCTP association with 1 or 4 subflow(s) between Hosts A and B (Fig. 5.8), against four different types of background traffic. For both SF-SCTP and the background traffic, we use a packet size of 1000 bytes. The four types of background traffic are described as follows:

- *Experiment IX–Constant Bit Rate (CBR):* The CBR traffic source is always on and generates packets according to a deterministic rate of 5Mbps. The shape of the CBR traffic, which is a straight line, is given in Fig. 5.20.

- *Experiment X–Square-Wave:* The Square-Wave traffic has 5sec deterministic "on" and "off" recurrent periods. The transmission rate is 8Mbps during the "on" period and zero during the "off" period. The shape of the Square-Wave traffic is in Fig. 5.21.

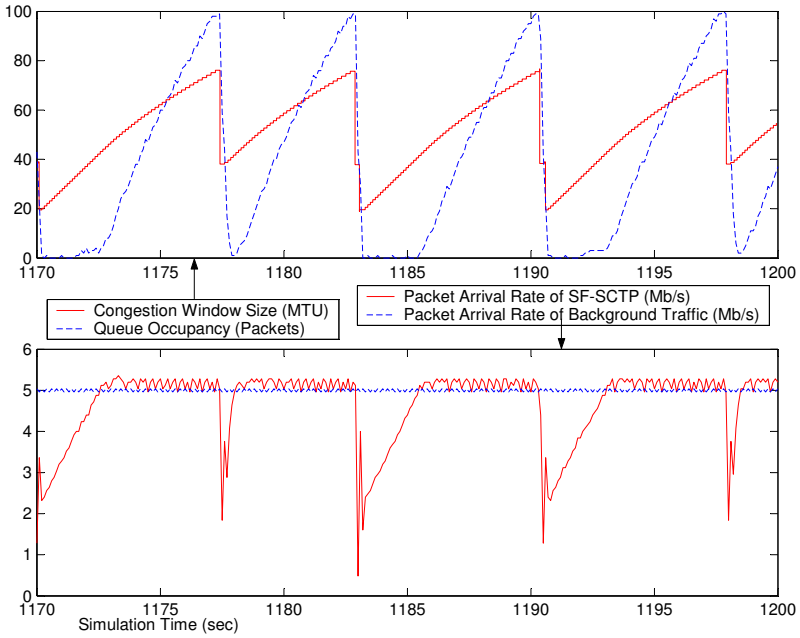
- *Experiment XI–Exponential On/Off:* The duration of the "on" and "off" periods satisfies an exponential distribution. The mean duration of "on" and "off" periods is set as 5sec and the transmission rate as 8Mbps during the "on" period. Two instances of the Exponential On/Off traffic are shown in Figs. 5.22(a) and 5.22(b).

- *Experiment XII–Pareto On/Off:* The Pareto On/Off traffic generates packets according to Pareto On/Off distribution. Packets are sent at a fixed rate of 8Mbps during "on" periods, and no packets are sent during "off" periods. Assuming Web object sizes form a Pareto distribution, the Pareto traffic generator can be used to simulate Web traffic. The average "on/off" periods and the transmission rate of the Pareto traffic are set as the same as the Exponential traffic. In the experiment, we set the Pareto shape parameter as 1.5, which is the ns-2 default value. We are uncertain whether an actual Web object size distribution may fall in our simulation setting or not. Two instances of the Pareto traffic are shown in Figs. 5.23(a) and 5.23(b).

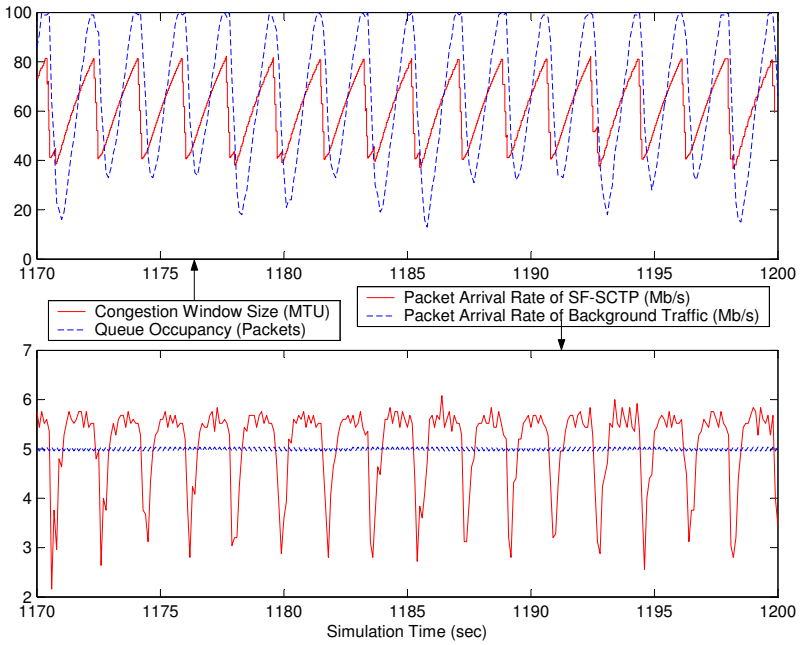
5.3.2 Discussion of Simulation Results

The simulation results for Experiments IX–XII are summarized in Figs. 5.20, 5.21, 5.22, 5.23 and 5.24. Figs. 5.20, 5.21, 5.22 and 5.23 show the *cwnd* behavior of SF-SCTP, the queue occupancy, and the transmission rates for both SF-SCTP and the background traffic. Fig. 5.24 shows the bandwidth share and packet drop rate of SF-SCTP from Experiments IX to XII.

As shown in Fig. 5.20, the CBR traffic, which has an invariable transmission rate of 5Mpbs, does not alter the periodic property of SF-SCTP. When the transmission rate of SF-SCTP is up to a level above 5Mbps, the bottleneck link is under full utilization and SF-SCTP is in direct competition with the CBR traffic. Although the bottleneck queue

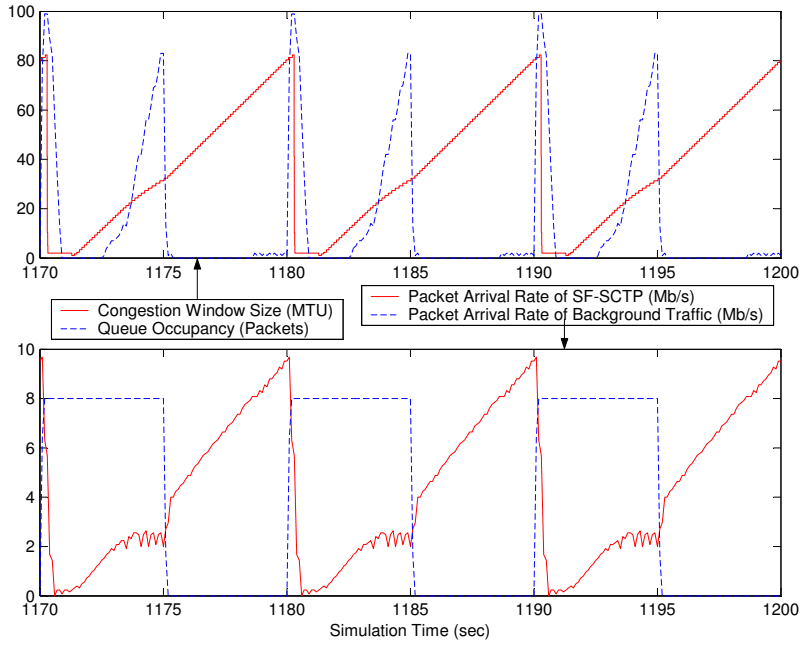


(a) SF-SCTP with one subflow.

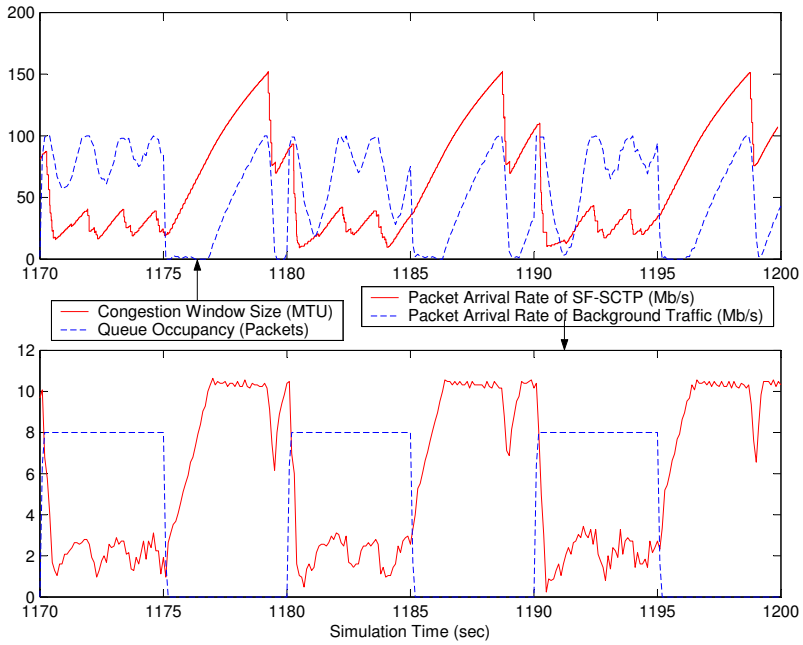


(b) SF-SCTP with four subflows.

Figure 5.20: Experiment IX: SF-SCTP's performance under the CBR traffic.

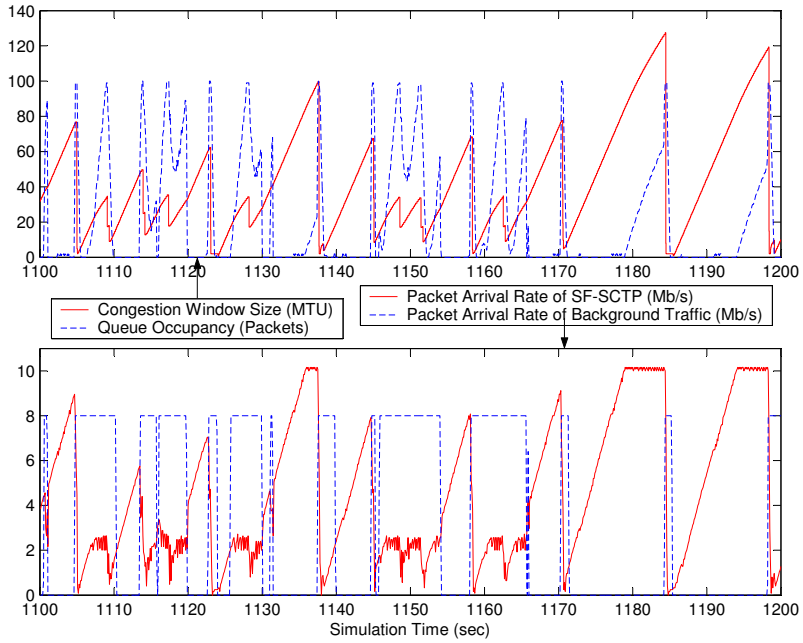


(a) SF-SCTP with one subflow.

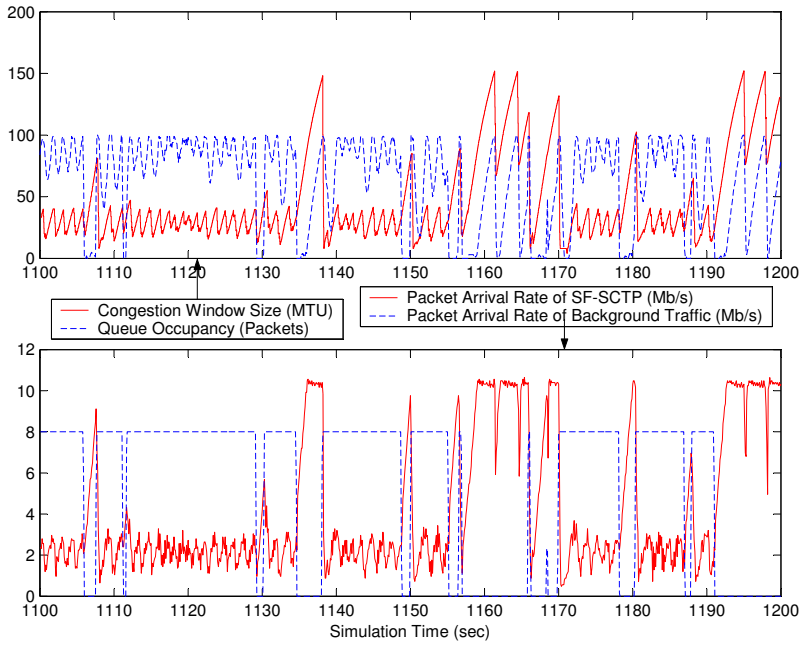


(b) SF-SCTP with four subflows.

Figure 5.21: Experiment X: SF-SCTP's performance under the Square-Wave traffic.

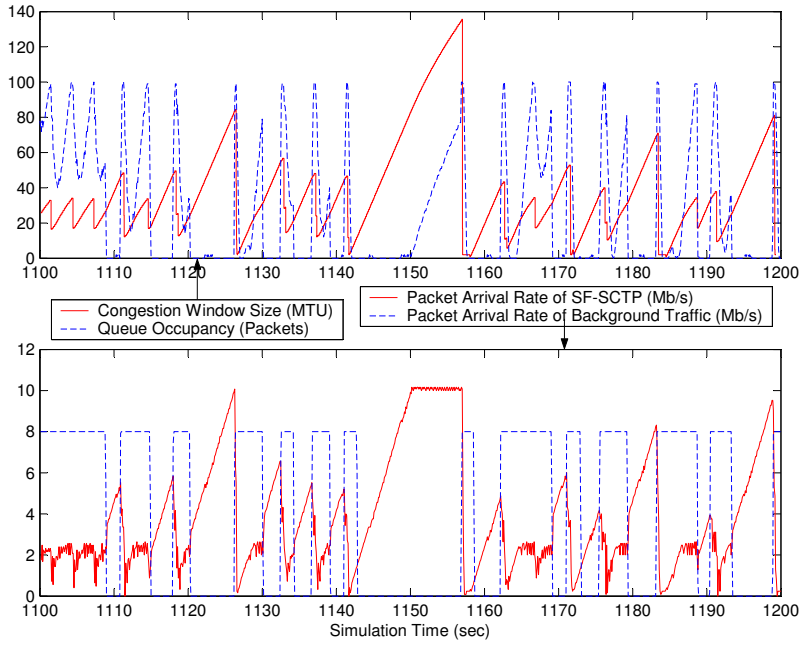


(a) SF-SCTP with one subflow.

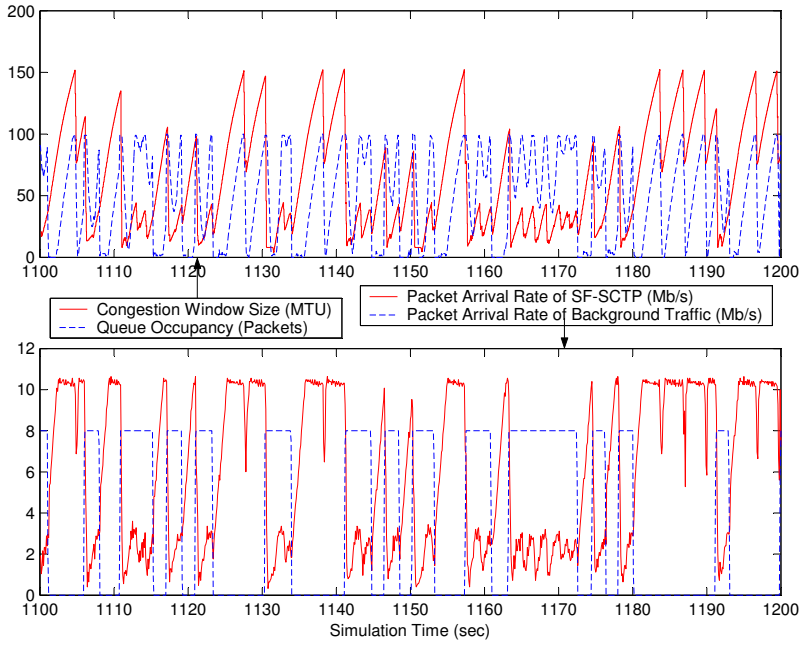


(b) SF-SCTP with four subflows.

Figure 5.22: Experiment XI: SF-SCTP's performance under the Exponential On/Off traffic.

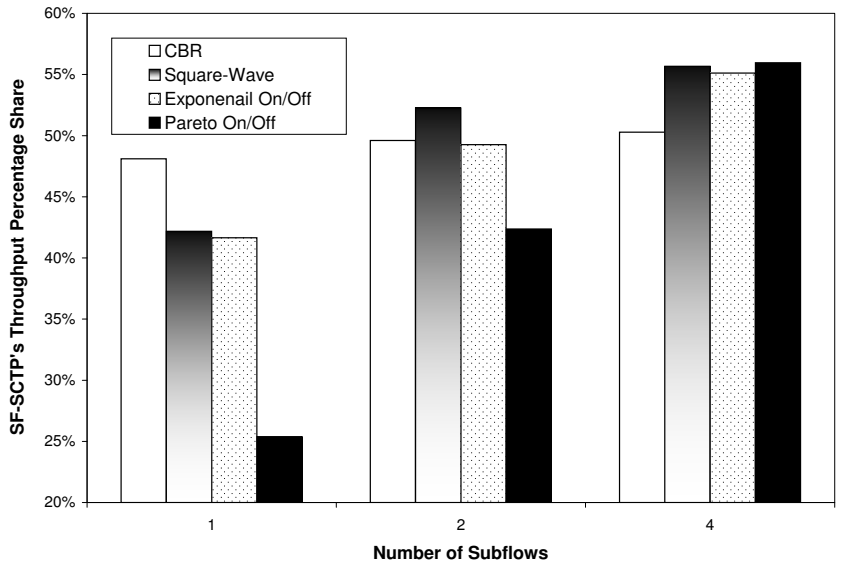


(a) SF-SCTP with one subflow.

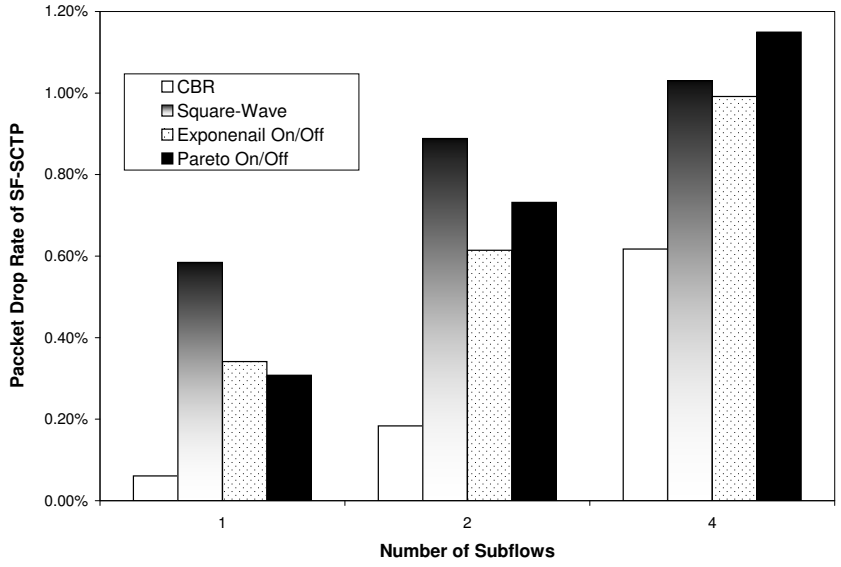


(b) SF-SCTP with four subflows.

Figure 5.23: Experiment XII: SF-SCTP's performance under the Pareto On/Off traffic.



(a) SF-SCTP's bandwidth share.



(b) SF-SCTP's drop rate.

Figure 5.24: The bandwidth share and drop rate of SF-SCTP with 1, 2 or 4 subflow(s) under various background traffic.

starts to build up, the SF-SCTP keeps increasing its *cwnd*, probing for more bandwidth share. This aggressive behavior of SF-SCTP will finally result in the bottleneck queue overflow, which in turn will cause SF-SCTP to reduce its transmission rate.

From Figs. 5.21, 5.22, and 5.23, we observe that SF-SCTP's periodicity is suppressed under the background traffic with variable transmission rates. When a "on" period begins and the background traffic suddenly starts to transmit at the fixed rate of 8Mbps, and if SF-SCTP is transmitting at a rate greater than 2Mbps, the bottleneck queue immediately builds up and becomes overwhelmed. Thereafter, SF-SCTP detects some packet drops and reduces its transmission rate. The abrupt pop-up nature of the background traffic may cause SF-SCTP to take several consecutive Fast Retransmits and finally a time-out event in order to reduce its transmission rate below 2Mbps. The sudden appearance of the background traffic normally indicates the earlier termination of the queue's QSD or QBU state, whichever the phase it belongs to at that time.

When a "on" period ends and the background traffic suddenly stops to transmit any packets, the competing SF-SCTP immediately detects the bandwidth gap and starts to fill that gap by increasing its *cwnd* and transmission rate. During this procedure, the queue may first undergo a short QDN period, then a QSD period, and finally a QBU period, assuming this procedure is not interrupted by another pop-up of the background traffic.

The background traffic in Experiment IX is configured to take 50% of the available band-

width, while in Experiments X, XI and XII, it takes 40%. Due to the packet drops introduced by the competing SF-SCTP association, the background traffic actually consumes less bandwidth than expected. This competition from SF-SCTP becomes tougher when SF-SCTP has more subflows as shown in Figs. 5.20, 5.21, 5.22, and 5.23.

For SF-SCTP of one subflow, we see from Fig. 5.24(a) that it performs best for the CBR traffic and obtains 48.1% of the total bandwidth. The performance of SF-SCTP with one subflow degrades when competing against a background traffic with variable transmission rates. When it comes to the Pareto traffic that exhibits the most dynamic transmission rate change by having the highest density of "on" and "off" period distribution, SF-SCTP with one subflow only seizes 25.4% share of the total bandwidth.

From Figs. 5.21(b), 5.22(b), 5.23(b) with Figs. 5.21(a), 5.22(a), 5.23(a), we observe that SF-SCTP with four subflows can reach a higher peak transmission rate, and that the queue stays much shorter in the QSD state. Both factors contribute to the higher bandwidth share of SF-SCTP with four subflows. When competing against background traffic with dynamic transmission rates, SF-SCTP with more subflows yields greater benefits, being able to adjust itself faster to network changes and even taking advantages of those changes. As shown in Fig. 5.24(a), the SF-SCTP with four subflows captures 55.9% share of bandwidth under the Pareto traffic, even higher than the 50.3% share it captures under the CBR traffic.

Also as shown in Fig. 5.24(b), SF-SCTP with more subflows brings a higher packet drop

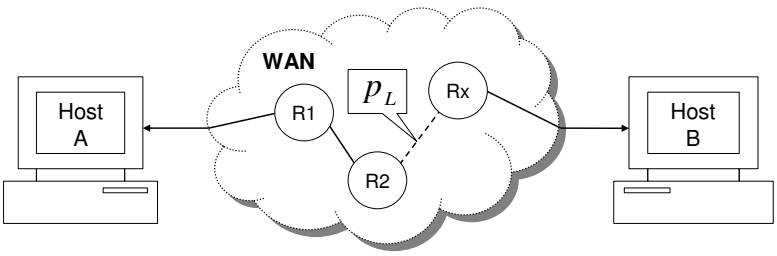


Figure 5.25: The topology of a Wide Area Network.

rate in the network while improving its own bandwidth share. The bandwidth improvement of SF-SCTP partially comes from the sacrifice of the background traffic. SF-SCTP with multiple subflows is more aggressive than the original SCTP or SF-SCTP with one subflow. It also has the negative effect of harming other network traffic. Fractional congestion control can be imported into SF-SCTP to alleviate its over-aggressiveness problem. The applicability of fractional congestion control to the SF-SCTP, which is published in Ref. [80], will be discussed in the next section.

5.4 Effects of Fractional Congestion Control

In this section, we report the simulations results from ns-2 experiments which have been conducted to investigate the effects of FCC on the throughput of SF-SCTP in various network environments. We also study the fairness of SF-SCTP with FCC towards original SCTP.

5.4.1 Wide Area Network

In a Wide Area Network, the effect of query delay, an parameter which is not included in our analytic model, can be ignored. Therefore, the result will move accurately match our analytic model. Fig. 5.25 is the topology used for Experiments XIII, XIV and XV. In our simulation settings, the number of routers (x) in Fig. 5.25 is set as three and the RTT between Hosts A and B is set as 0.5sec. A link error rate of p_L exists between the routers R2 and R3. The SCTP packet size all simulation experiments reported in this section is set as 1500bytes, which equals the Maximum Transmission Unit (MTU) in our simulation network.

In Experiment XIII and XIV, all the links in Fig. 5.25 are configured to have almost infinite bandwidth such that all packet losses in the network are due to the link errors between R2 and R3. Experiment XIII is designed to study the effects of FCC on SF-SCTP with small number of subflows under various packet loss rates environments. In Experiment XIII, we run one single SCTP flow and one SF-SCTP with two subflows between Hosts A and B. For each simulation running, the link error rate p_L varies from 10^{-3} to 10^{-5}. Also for each p_L value, we set the Φ value to be 1, 1/2, 1/4 and 1/8 to find out the effects of different *cwnd* growth rate on the throughput of SF-SCTP. Experiment XIV is used to study the effects of FCC on SF-SCTP with various number of subflows from 1 to 27. The link error rate p_L is set as 10^{-4} in Experiment XIV. The Φ parameter takes the value of 1, n, n^2, or $2n^2$ for each simulation running, where n is the number of

subflows in SF-SCTP. The SF-SCTP and single SCTP experience the same packet drop rate and RTT under the simulation settings of Experiment XIII and XV.

The simulation setting of Experiment XV is similar to Experiment XIV except that the bottleneck link in Experiment XV is set to have a limited bandwidth of 20Mbps. The bottleneck queue in Experiment XV may be overwhelmed and drop packets due to congestion. Due to the effects of FCC, in Experiment XV, the SF-SCTP with FCC will experience a lower packet drop rate than single SCTP.

The throughput results from simulations and the corresponding analytic expectations for Experiments XIII to XV are presented in Figs. 5.26, 5.27 and 5.28. Fig. 5.26 plots the throughput of SF-SCTP and single SCTP as a function of link error rate. There are eight curves in Fig. 5.26, presenting the analytic and simulation throughput of SF-SCTP under different Φ and link error rate values. Figs. 5.27 and 5.28(a) illustrate the relationship between the throughput and number of subflow in SF-SCTP. Fig. 5.27(a) shows the throughput of SF-SCTP when Φ equals 1 and n, while in Fig. 5.27(b), Φ equals n^2 and $2n^2$. Fig. 5.28(b) of Experiment XV shows the SF-SCTP's Fast Retransmit rate, which is an accurate indication of the packet loss rate from both link errors and queue overflow.

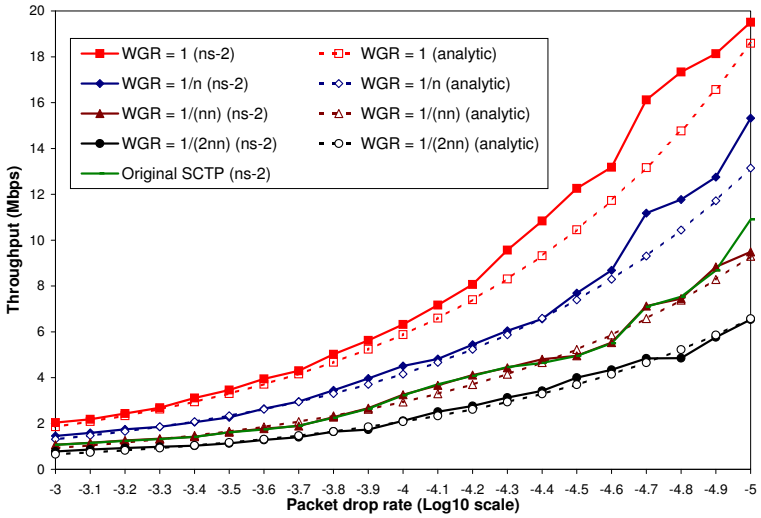
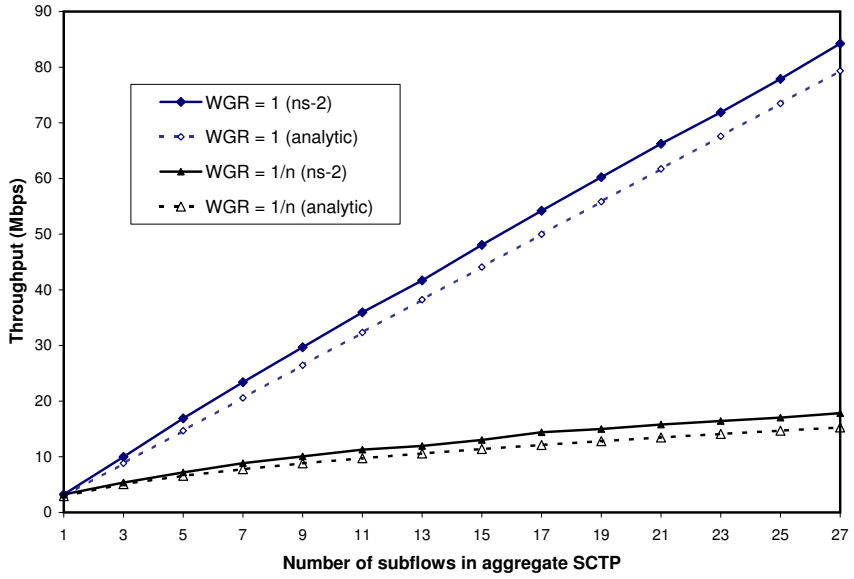


Figure 5.26: Experiment XIII: Throughput of SF-SCTP and single SCTP.

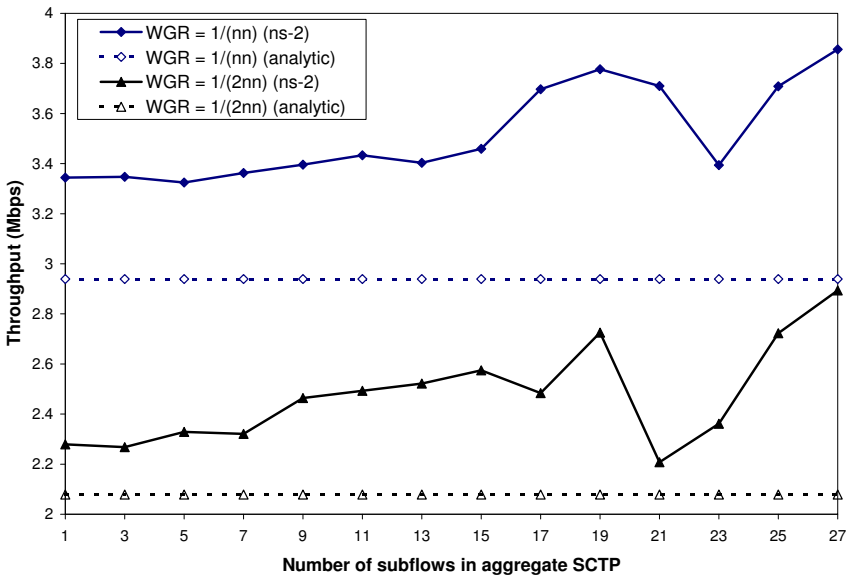
5.4.2 Discussions of FCC in WAN

The simulation scenarios of Experiment XIII and XIV apply to the following situations: (1). A saturated WAN with almost fix packet drop rate. Adding or removing an SCTP flow in this network will not affect the packet drop rate. (2). A wireless network where link errors dominate. The simulation scenarios of Experiment XV apply to the following situations: a wireless network where packet may drop due to either congestion or link error.

The results from Experiment XIII in Fig. 5.26 prove the accuracy of the throughput relationship equation $B_{ag} = \sqrt{\frac{n^2}{\Phi}} B_{sg}$. When $\Phi = 1$, SF-SCTP with two subflows achieves roughly twice throughput as single SCTP. When $\Phi = n = 2$, SF-SCTP achieve roughly $\sqrt{2}$ times throughput as single SCTP. When $\Phi = n^2 = 4$, SF-SCTP achieve almost the

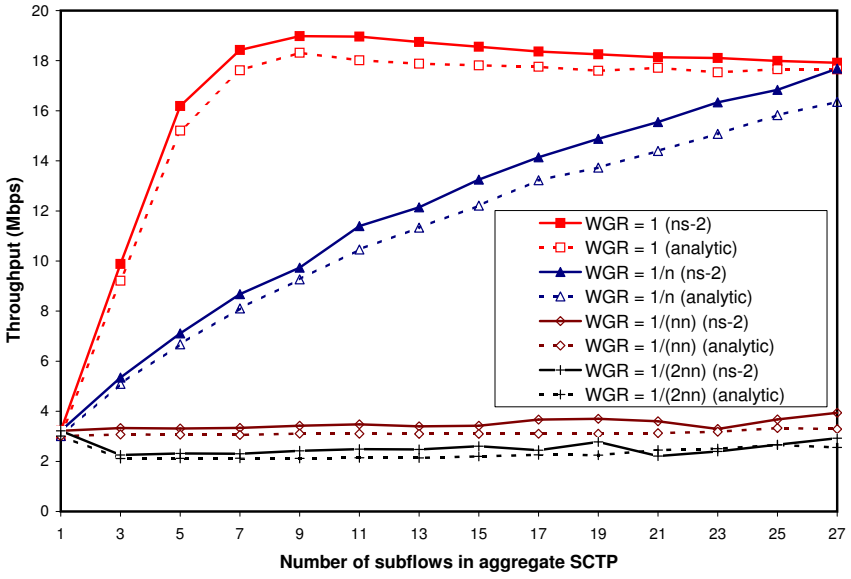


(a) $\Phi = 1$ or n.

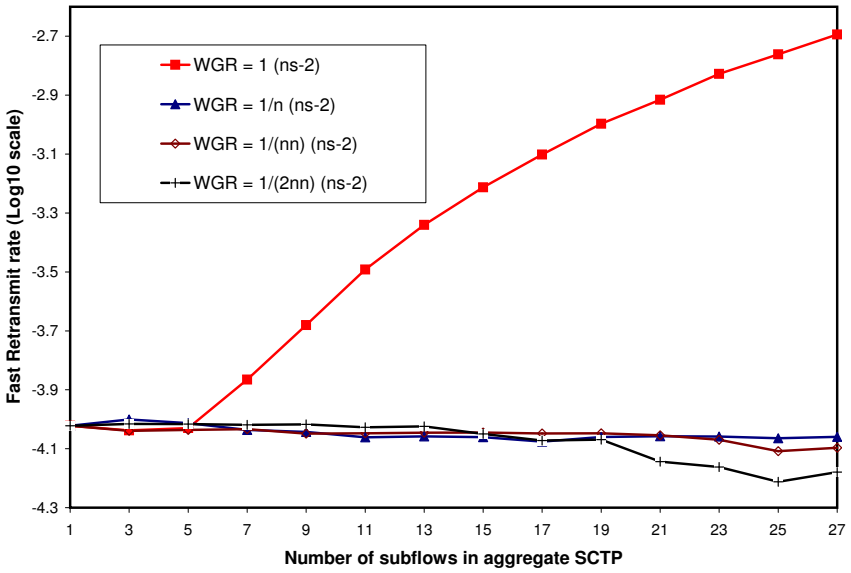


(b) $\Phi = n^2$ or $2n^2$.

Figure 5.27: Experiment XIV: Throughput of SF-SCTP.



(a) Throughput.



(b) Fast Retransmit rate.

Figure 5.28: Experiment XV: Throughput and Fast Retransmit rate of SF-SCTP.

same throughput as single SCTP. When $\Phi = 2n^2$, SF-SCTP achieves less throughput than single SCTP and $B_{ag} = \frac{\sqrt{2}}{2} B_{sg}$.

From Fig. 5.27 of Experiment XIV, we observe that when $\Phi = 1$ or n, the throughput of SF-SCTP increases as the number of subflows increases, while when $\Phi = n^2$ or $2n^2$, we do not observe this clear trend. According to the analytic models, when $\Phi = n^2$ or $2n^2$, the throughput of SF-SCTP should not change. However, as mentioned before, our models do not capture all the congestion control behavior of SF-SCTP. On one hand, as the n increases, SF-SCTP is allowed to send more extra packets out each Round Trip Time. On the other hand, the Congestion Window Grow Rate (WGR) of each subflow becomes slower as n increases and subflows may resort to time-outs more often.

From Fig. 5.28(a) for Experiment XV, for a bottleneck network with both congestion and non-congestion losses, when $\Phi = 1$, SF-SCTP needs only 7 subflows in order to reach 90% of network utilization. When $\Phi = n$, SF-SCTP needs 27 subflows to reach 90% of network utilization. When $\Phi = n^2$ or $2n^2$, the throughput of SF-SCTP is totally constrained by the link errors and slow *cwnd* grow rate.

From Fig. 5.28(b) for Experiment XV, for $\Phi = 1$, congestion losses begin to occur when $n > 5$ and increases very fast with the increasing of n. An interesting discover from Fig. 5.28(b) is that the increasing of n do not necessary help to increase the network utilization. For $n \in [1, 9]$, the increase of n help to overcome the constrain of link errors and thus improve the throughput of SF-SCTP. When $n > 9$, the increasing of n accompanies

the decreasing of throughput and at the same time, the dramatic increasing of packet drop rate.

Conclusions (for a saturated network with almost fixed packet drop rate or a wireless network with : (1) In order to make SF-SCTP fair to single SCTP, we would like to make $\Phi = n^2$. However, by doing this, the effectiveness of opening multiple subflows is lost. (2) For a network with non-congestion losses, we would suggest Φ to take the value of n. In this way, the overall *cwnd* of SF-SCTP increase by one MTU per RTT and the advantages of SF-SCTP over single SCTP is mainly due to its greater resistance to packet losses. Also in this way, the network maybe able to achieve a high utilization at the cost the of low packet drop rate.

5.4.3 Single Bottleneck Network.

In this section, we present the simulation results of two experiments performed under the single bottleneck network topology as shown in Fig. 5.8. Experiment XVI is configured as the large queue where the maximum queue delay exceeds the basic RTT. Since $D_Q > R_L$, the queue will never be empty after the SF-SCTP reaches steady state. Experiment XVII is configured as the small queue case where the maximum queue delay is less than the basic RTT. The network configurations for Experiments XVI and XVII are shown in Table 5.4.

Experiments XVI and XVII have been run for different instances of SF-SCTP with number

Table 5.4: Network configurations for Experiments XVI and XVII (Exp. = Experiment, pkts = packets).

	Bottleneck Bandwidth	Queue Size	Basic RTT	Max. Queue Delay
Exp. XVI	10Mbps	100pkts	100ms	120ms
Exp. XVII	20Mbps	20pkts	100ms	12ms

of subflows varying from 1 to 25. For each simulation running, we obtain the average goodput, Fast Retransmit rate and capture the *cwnd*, queue occupancy, transmission rate of SF-SCTP.

Fig. 5.29 provides a comparison of the *cwnd* behavior from ns-2 simulation results and analytic model for SF-SCTP with one subflow. When plotting Fig. 5.29, we choose the time $t = 0$ as the beginning of a Fast Retransmit period. The real line in Fig. 5.29 is plotted according to the tracing data from ns-2 simulations, while the dotted line is calculated by corresponding analytic equations, specifically, Eq. (4.51) for Experiment XVI and Eq. (4.52) for Experiment XVII. Fig. 5.29 shows that the *cwnd* predicted by the deterministic model almost perfectly matches the real ns-2 simulation results. As shown in Fig. 5.29, the only discrepancy is that the analytic model underestimate the highest value of *cwnd*. Our models approximate the highest *cwnd* value as W_Q ($W_{max} \approx W_Q$), while in reality, it can reach a value slightly above W_Q. This discrepancy makes the Fast Retransmit period in analytic model appears shorter than in reality, which further lead

to a slight overestimation of the packet drop rate.

Fig. 5.30 shows the strong periodicity property of *cwnd*, queue occupancy, and transmission rate of SF-SCTP with only one subflow. The comparison of Experiments XVI and XVII in Fig. 5.30 illustrates the advantages of a large bottleneck link queue size, which effectively shortens or even eliminates the QSD period and help to improve network utilization.

Figs. 5.31 and 5.32 show the *cwnd* behavior, while Figs. 5.33 and 5.34 show the queue occupancy of SF-SCTP with four and twelve subflows, respectively. Each of the above four figures has four subfigures, corresponding to different WGR values of 1, $1/n$, $1/n^2$ and $1/(2n^2)$. In Figs. 5.31 and 5.32, when $WGR = 1$, which is the case FCC is not applied, we always observe the overall *cwnd* of SF-SCTP reduces to half in responding to packet losses due to the global synchronization problem of Drop-Tail queue. For the case FCC is applied ($WGR = 1$, $1/n^2$ or $1/(2n^2)$), when queue is full and drop packets (as shown in Figs. 5.33 and 5.34), most of the time, only a subset flows of the SF-SCTP are affected. FCC alleviate the global synchronization problem of Drop-Tail router and help to provide SF-SCTP a greater resistant to packet losses in Drop-Tail Environments. Also as shown in Figs. 5.31 and 5.32, a small WGR value necessarily means a smoother *cwnd* behavior.

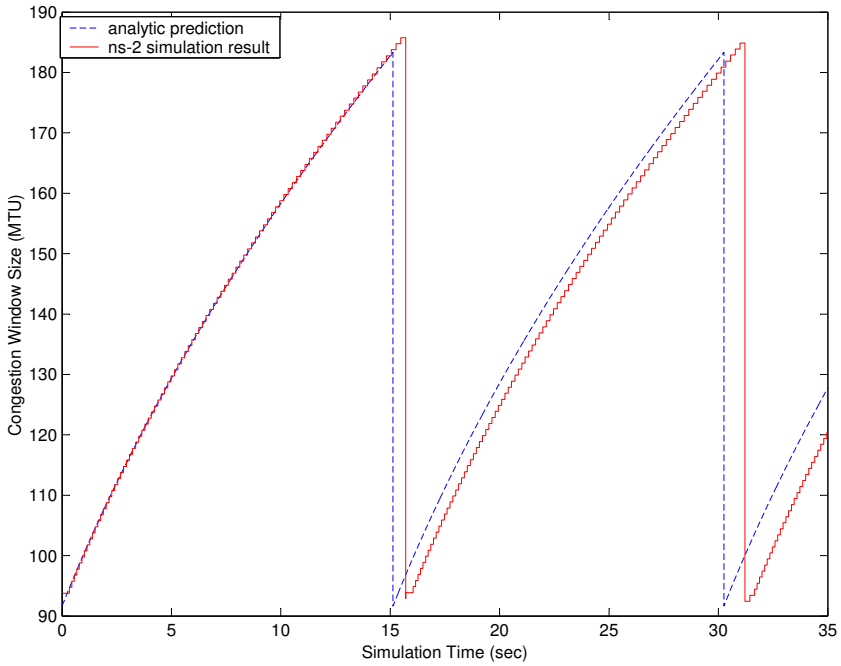
Figs. 5.35 and 5.36 plots the goodput and Fast Retransmit rate of SF-SCTP as the number of subflows increasing from 1 to 25. As the number of subflows increases, we observe

the goodput of SF-SCTP decreasing in Experiment XVI, while in Experiment XVII, the goodput is increasing. In Experiment XVI, the queue is always busy transmitting packets, thus, under full utilization. When $WGR = 1$ or $1/n$, as the number of subflows increasing, SF-SCTP becomes more aggressive and incur more packet losses since the network is already under full utilization. Those excessive packet losses harm the goodput of aggregate in Experiment XVI. When $WGR = 1/n^2$ or $1/(2n^2)$, although the increasing of number of subflows does not increase Fast Retransmit rate, it does increase the time-out events of SF-SCTP, which hurts SF-SCTP's goodput in Experiment XVI. In Experiment XVII, since the network still has bandwidth available, the increasing of number of subflows in SF-SCTP helps to consume those unused bandwidth.

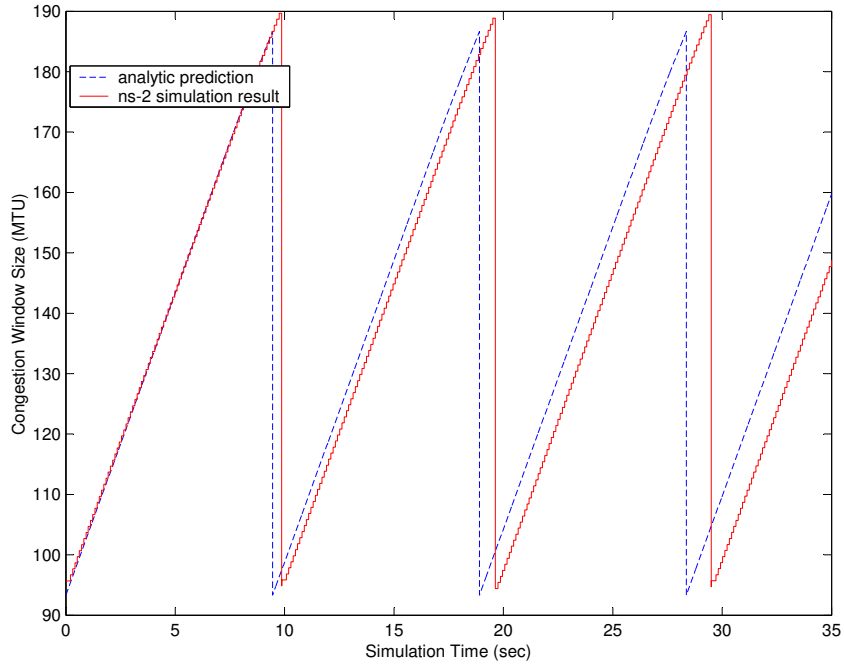
5.4.4 Fairness Study in Single Bottleneck Network

The simulation topology shown in Fig. 5.8 has been used in Experiments XVIII to XXI to investigate the fairness between SF-SCTP and single SCTP under various network environments. Each of the Experiments from XVIII to XXI has two scenarios. In scenario (a), we have SF-SCTP competing with a single SCTP flow, while in scenario (b), SF-SCTP is competing against the same number of single SCTP flows. The network settings for Experiments XVIII to XXI are described as follows:

- *Experiment XVIII:* The bottleneck link bandwidth is 10Mbps. The basic RTT without considering of the queue delay is 100ms. The queue size at the bottleneck router is 100

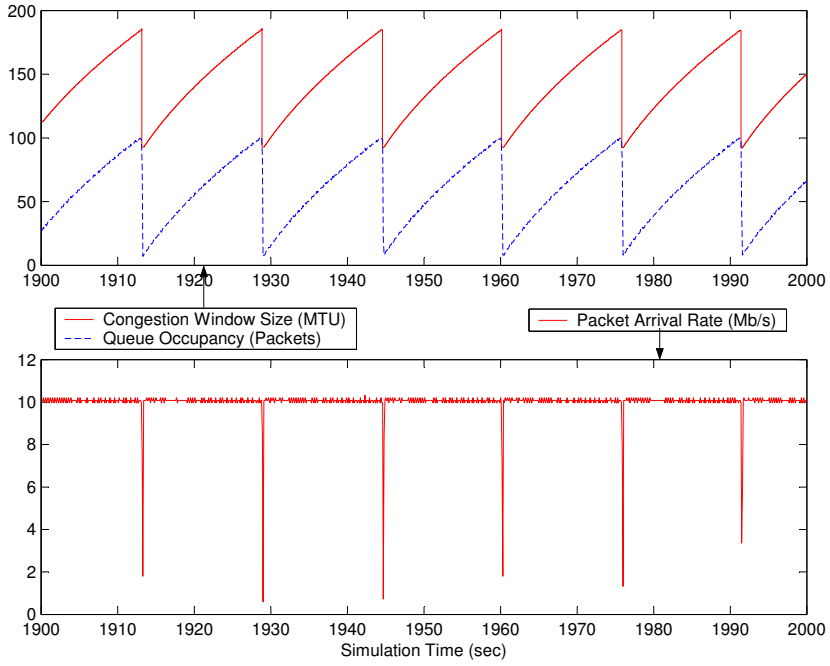


(a) Experiment XVI.

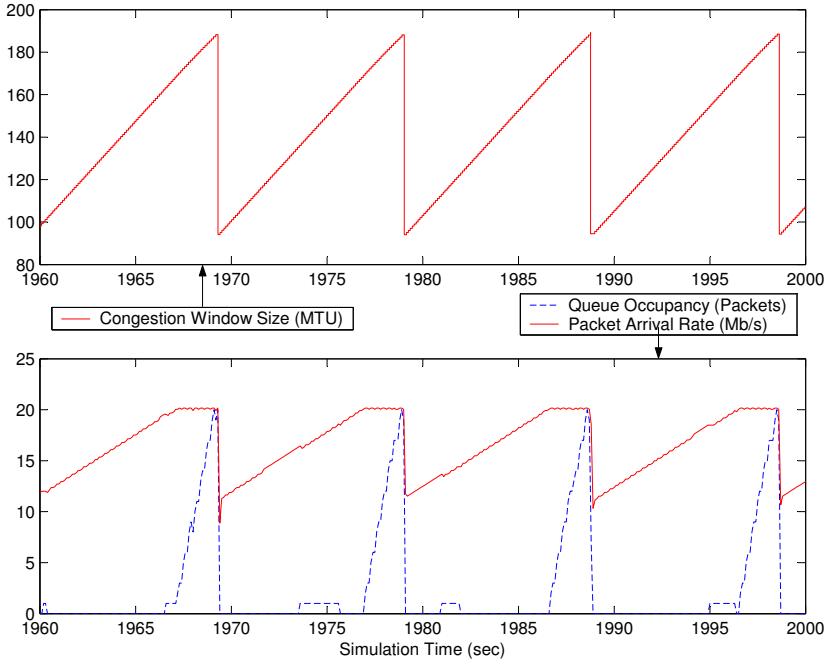


(b) Experiment XVII.

Figure 5.29: Comparison of the *cwnd* behavior from analytic model and ns-2 simulation for SF-SCTP with one subflow.

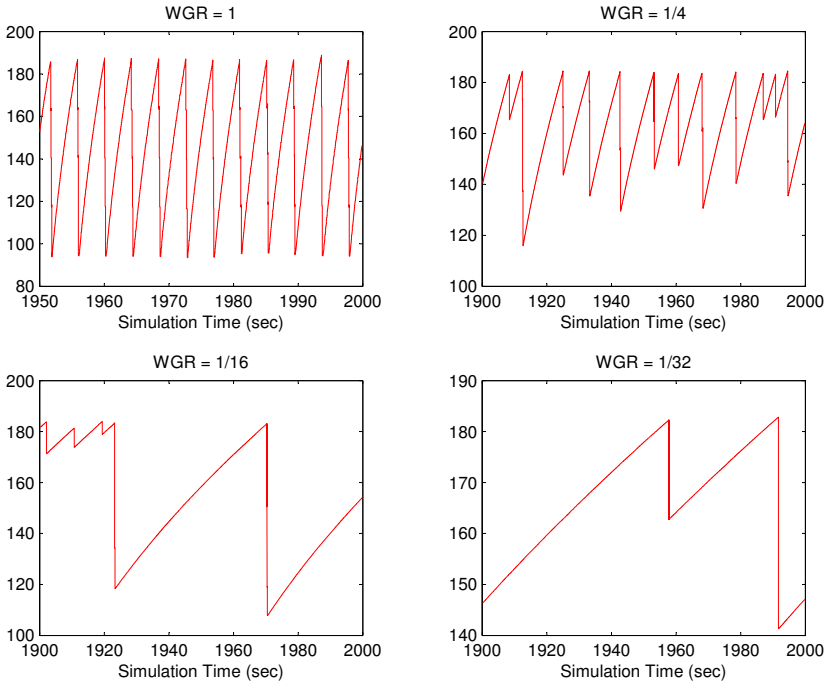


(a) Experiment XVI.

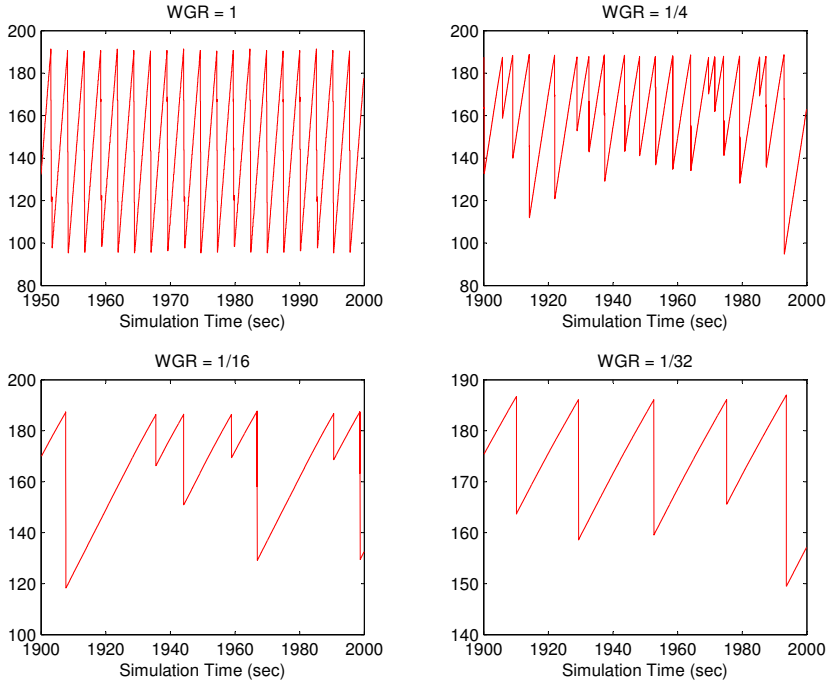


(b) Experiment XVII.

Figure 5.30: Queue occupancy, transmission rate and *cwnd* of SF-SCTP with one subflow.

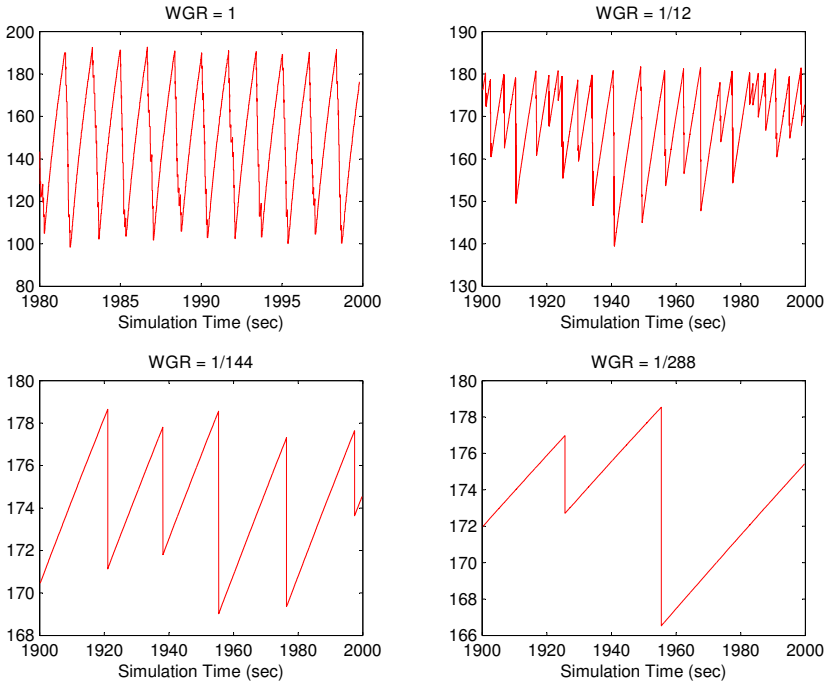


(a) Experiment XVI.

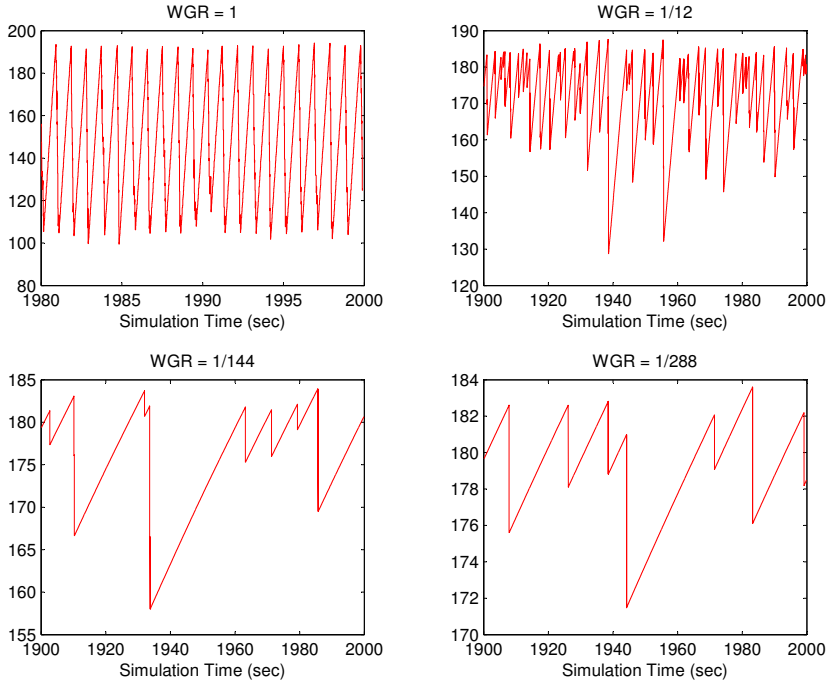


(b) Experiment XVII.

Figure 5.31: The *cwnd* behavior of SF-SCTP with four subflows.

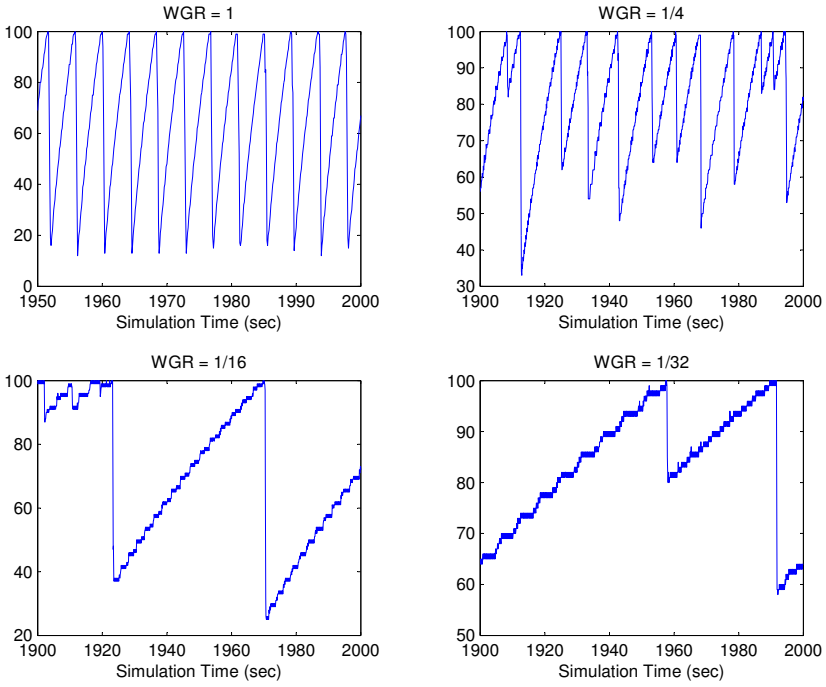


(a) Experiment XVI.

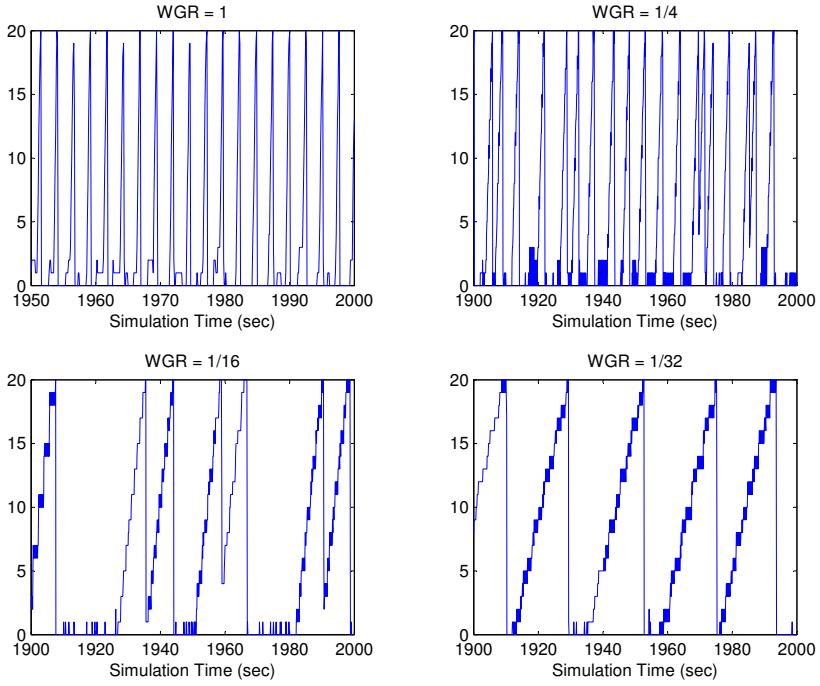


(b) Experiment XVII.

Figure 5.32: The *cwnd* behavior of SF-SCTP with twelve subflows.

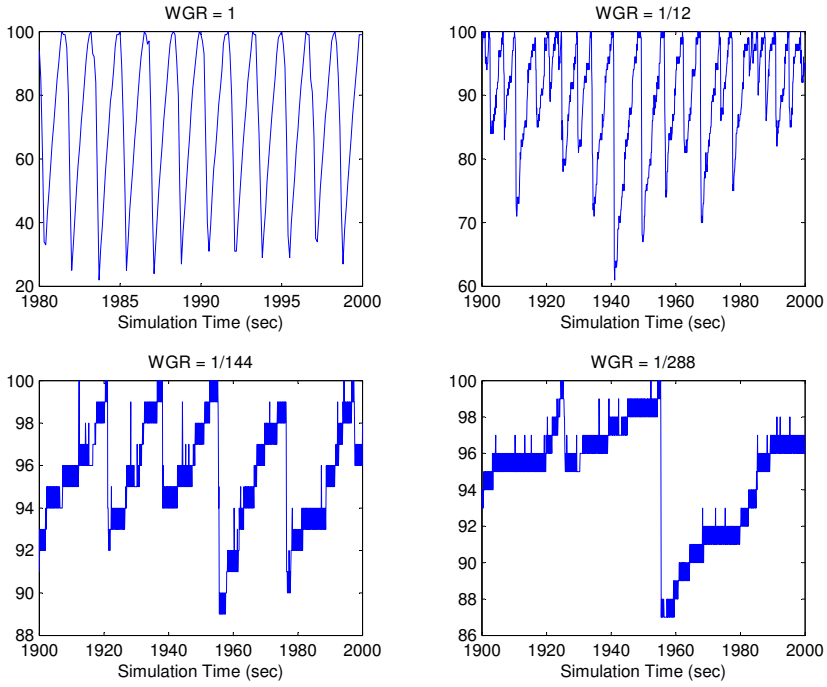


(a) Experiment XVI.

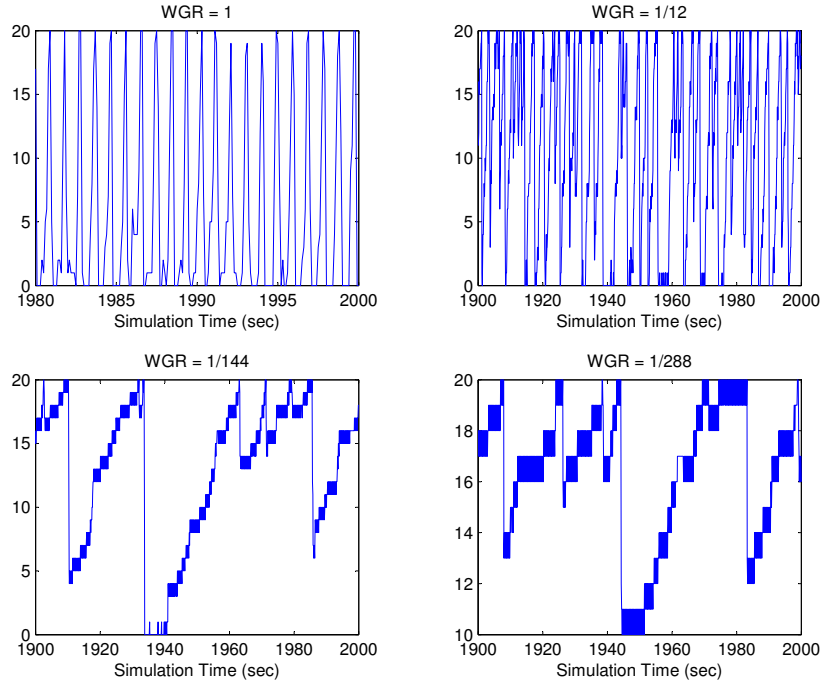


(b) Experiment XVII.

Figure 5.33: The queue occupancy of behavior of SF-SCTP with four subflows.

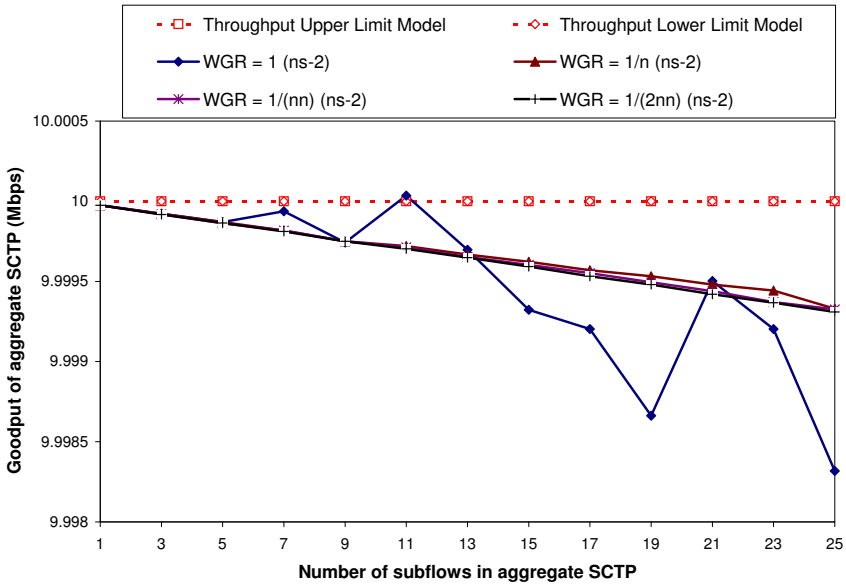


(a) Experiment XVI.

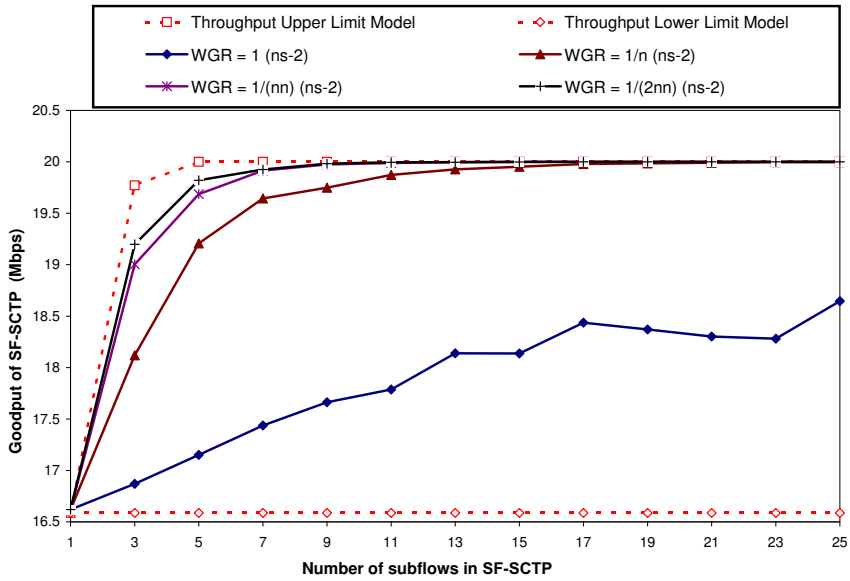


(b) Experiment XVII.

Figure 5.34: The queue occupancy behavior of SF-SCTP with twelve subflows.

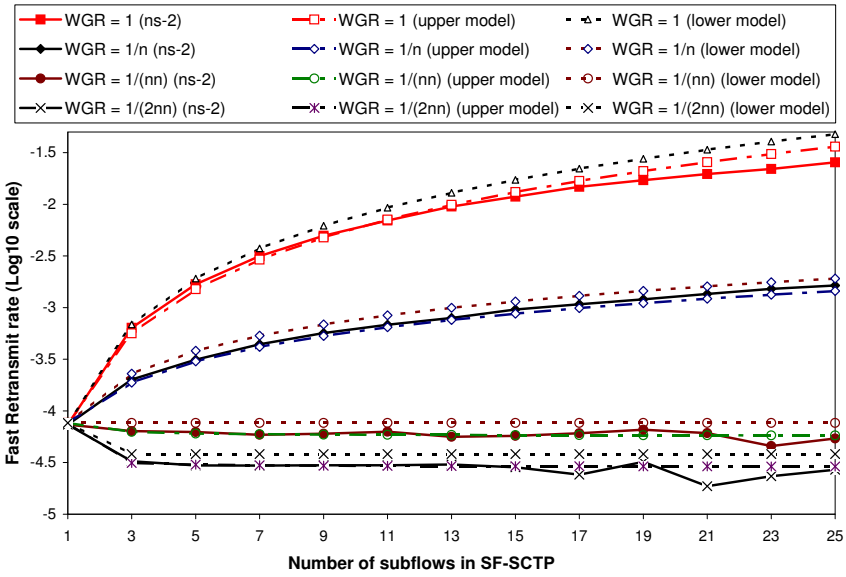


(a) Experiment XVI.

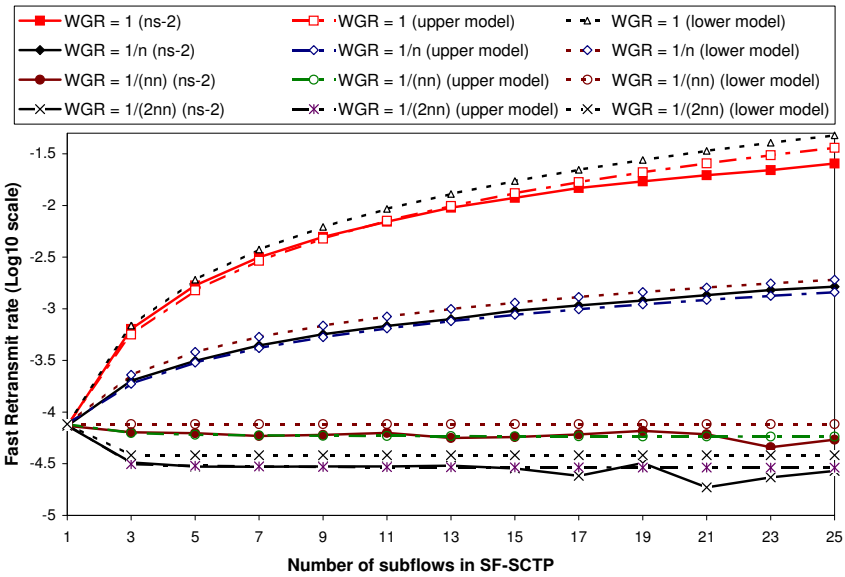


(b) Experiment XVII.

Figure 5.35: The throughput of SF-SCTP when number of subflows varies from 1 to 25.



(a) Experiment XVI.



(b) Experiment XVII.

Figure 5.36: The throughput of SF-SCTP when number of subflows varies from 1 to 25.

packets. The packet size of SF-SCTP and single SCTP is 1000 bytes. All packet drops in this experiment are due to congestion and there are no link errors. Except the SF-SCTP and single SCTP, there are no other traffics in the network. In scenario (a), we vary the number of subflows in SF-SCTP from 1 to 15 to collect simulation results, while in scenario (b), the number of subflows in SF-SCTP and the number of single SCTP flows are varying together from 1 to 15.

- *Experiment IXX:* With the other network settings same as Experiment XVIII, we introduce a link error rate of 10^{-3} between the Router and Host B in Fig. 5.8.

- *Experiment XX:* With the network settings similar as Experiment IXX, we fix the number of subflows in SF-SCTP as four and vary the link error rate from 10^{-5} to 10^{-2} to collect simulation results.

- *Experiment XXI:* With the other network settings same as Experiment XVIII, we introduce a background traffic between Hosts A and B. The background traffic is of squarewave whose shape has 4sec deterministic "on" and "off" recurrent periods. The transmission rate is 8Mbps during the "on" period and zero during the "off" period.

Two important indexes of the simulation results for Experiments XVIII–XXI, the goodput and Fast Retransmit rate of SF-SCTP and single SCTP, are summarized in Figs. 5.37, 5.38, 5.39, 5.40, 5.41, 5.42, 5.43, and 5.44.

There are two special cases where SF-SCTP is totally fair towards single SCTP flow(s). When the number of subflows in SF-SCTP equals one, SF-SCTP becomes single SCTP and it must be towards one single SCTP flow. As shown in Figs. 5.37(a), 5.38(a), 5.39(a), 5.40(a), 5.43(a), and 5.44(a), the goodput of SF-SCTP equals that of single SCTP when number of subflows in SF-SCTP is one. The other case is, when SF-SCTP is competing against the same number of single SCTP flows and FCC is not applied for SF-SCTP. As shown in Figs. 5.38(a), 5.40(a), 5.42(a), and 5.44(a), when $WGR = 1$, the solid line for SF-SCTP and the dash line for single SCTP overlap with each other.

Examining Figs. 5.37(a) and 5.38(a) for Experiment XVIII, we observe that 1) for a network where packet drops are due to queue overflow, FCC alleviates the aggressiveness of SF-SCTP and makes it more, but not 100%, fair towards single SCTP; 2) with SF-SCTP implementing FCC, single SCTP's performance benefits from both the reduced throughput of SF-SCTP; 3) FCC reduces the packet drop rates for both SF-SCTP and single SCTP and SF-SCTP has a smaller packet drop rate than single SCTP after FCC is implemented.

Also from Experiment XVIII, we find that, when Φ is already large enough and in the rank of n^2, adjusting the Φ parameter does not necessarily help SF-SCTP become more fair towards the single SCTP when packet losses are primary due to network congestion. When Φ becomes larger, the overall *cwnd* of SF-SCTP increase slower, which harms the goodput of SF-SCTP. On the other side, a large Φ value reduces the packet drop rate

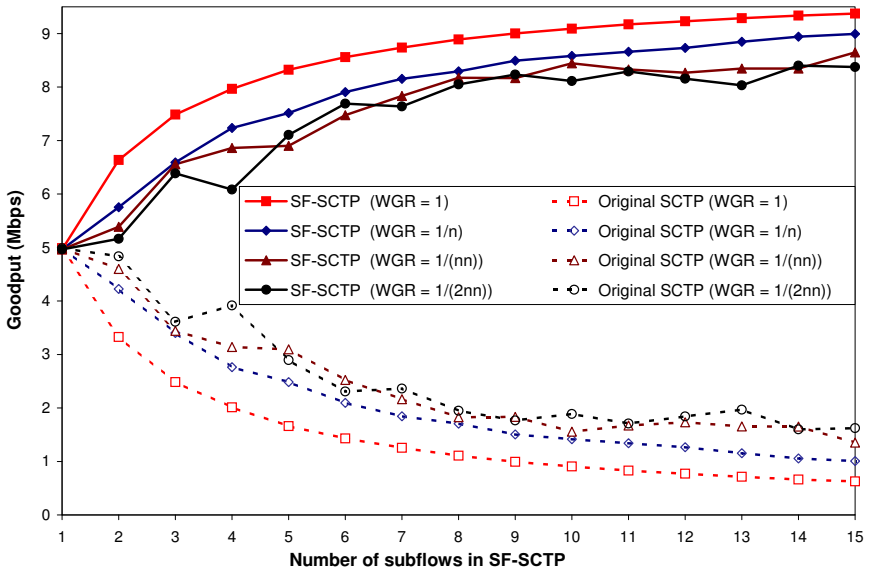
of SF-SCTP, which benefits the goodput of SF-SCTP. The effects of smaller packet drop rate of SF-SCTP may partially offsets the effects of slower *cwnd* increase rate. Therefore, it is hard to compare the goodput of SF-SCTP with $\Phi = n^2$ or $2n^2$.

The only difference between Experiments XVIII and IXX is that Experiment XVIII introduce a link error rate of 10^{-3}. Comparing the goodput results of those two experiments, we observe that SF-SCTP with a larger Φ value is more likely to be affected by the link error rate. This result is intuitive and can be explained by the packet drop rate of SF-SCTP. As Experiment XVIII points out, SF-SCTP with a larger Φ has a smaller packet drop rate, which means smaller number of packets dropped due to congestion. After a fixed link error rate is introduced, SF-SCTP with small packet drop rate are more likely to be overwhelmed. Take SF-SCTP with $\Phi = n^2$ and $2n^2$ as example, in Experiment XVIII, the packet drop rates of both SF-SCTP are much smaller than 10^{-3}, while in Experiment IXX, we found both SF-SCTP have the packet drop rates of roughly 10^{-3}, which means almost all packet drops are due to link losses.

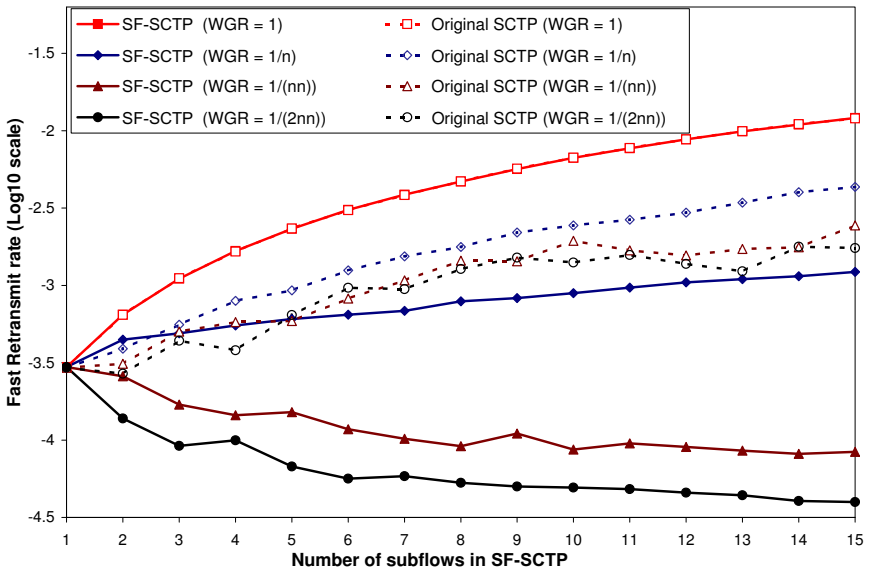
From the goodput results of Experiment XX, we observe that, compared to single SCTP, SF-SCTP with FCC is more likely to be affected by the link error rate. Take the SF-SCTP with $\Phi = n^2$ as the example, in Fig. 5.41(a) when the link error rate increases, the goodput of SF-SCTP decreases continuously, while the goodput of SF-SCTP increases before link error rate reaches $10^{-3.4}$, seizing the released bandwidth from SF-SCTP. After the link error rate reaches above $10^{-3.4}$, when $\Phi = n^2$, both the goodput of SF-SCTP

and single SCTP decreases since the link errors begin to take domination to control the goodput.

Comparing the simulation result of Experiment XXI with Experiment XVIII, we observe that SF-SCTP with larger Φ value is more likely to be affected by background traffic. As we know, the parameter Φ control the *cwnd* growth rate of SF-SCTP. When the background squareware traffic suddenly appear, consuming almost 80% of the bandwidth, all SCTP flows in the network are likely to be affected and have to reduce their *cwnd*. Since SCTP reduces its *cwnd* aggressively, one half for each Fast Retransmit event, after almost all SCTP flows in the network reduces their *cwnd*, there maybe some available bandwidth in the network. Also when the background traffic suddenly disappear, 80% of the network bandwidth will immediately become available, letting SF-SCTP and single SCTP to consume. When SF-SCTP has a larger Φ parameter, its *cwnd* will recover slower, making it harder to seize the available bandwidth compared to single SCTP flow(s).

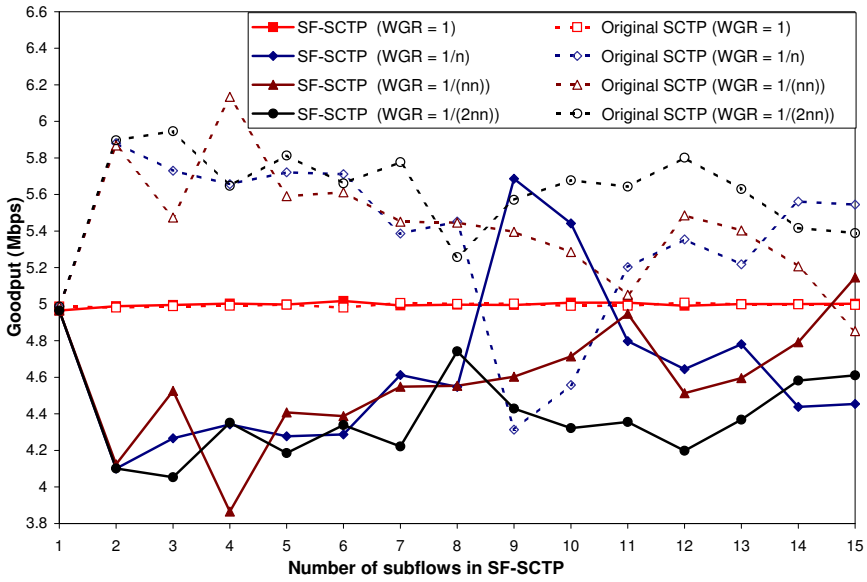


(a) Goodput.

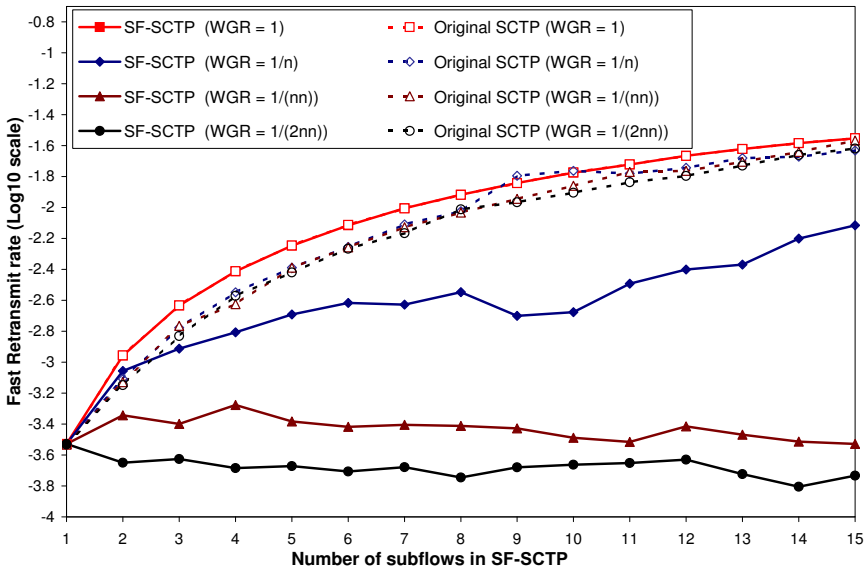


(b) Fast Retransmit rate.

Figure 5.37: Experiment XVIII(a): SF-SCTP competing with one single SCTP flow.

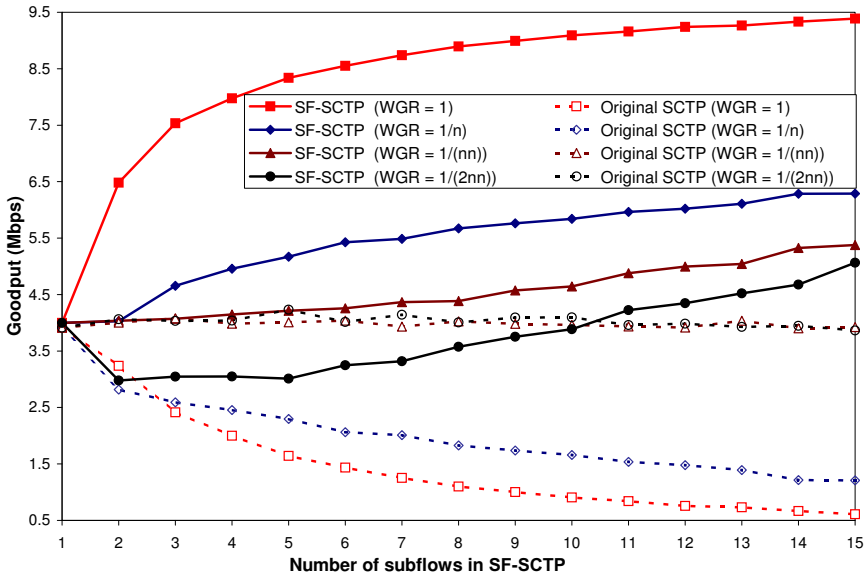


(a) Goodput.

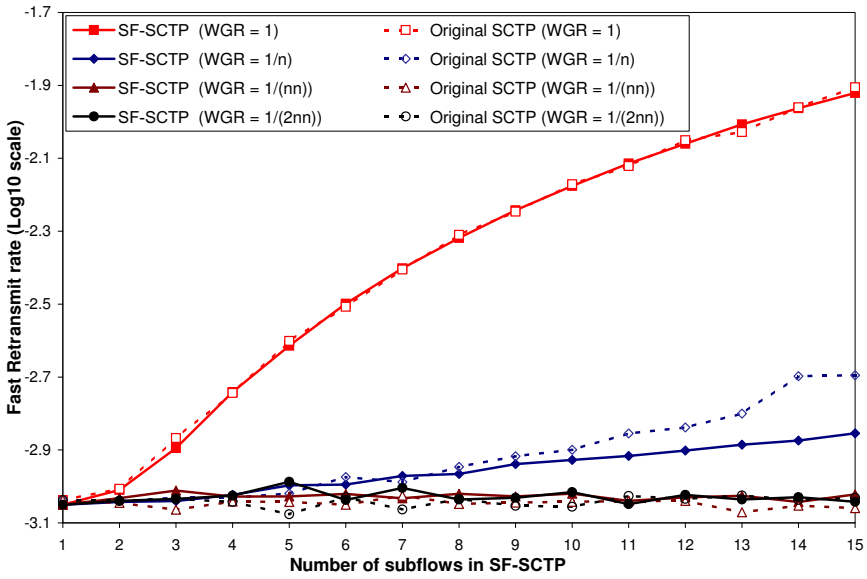


(b) Fast Retransmit rate.

Figure 5.38: Experiment XVIII(b): SF-SCTP competing with the same number of single SCTP flows.

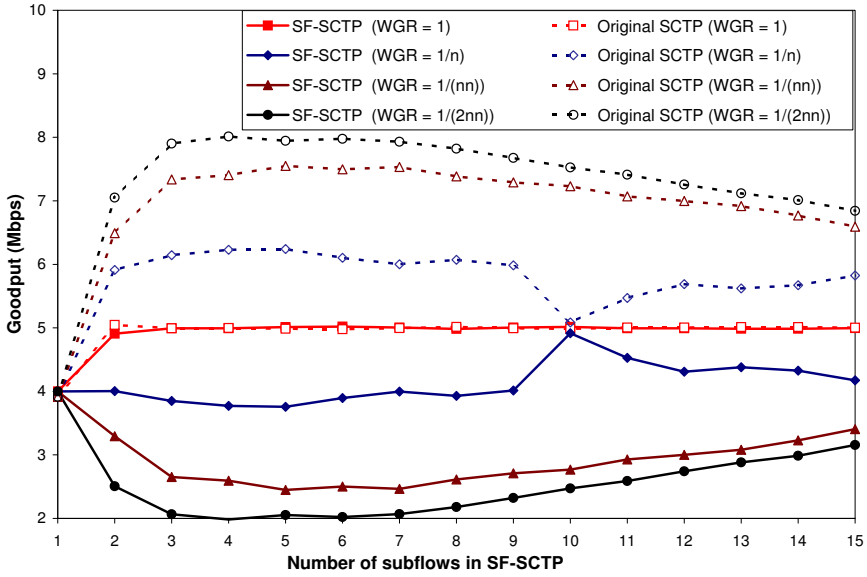


(a) Goodput.

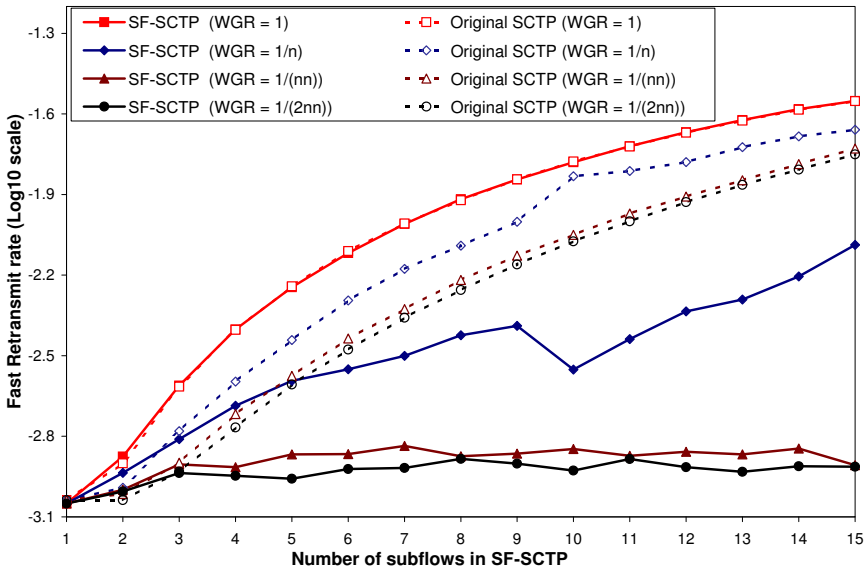


(b) Fast Retransmit rate.

Figure 5.39: Experiment IXX(a): SF-SCTP competing with the same number of single SCTP flows.

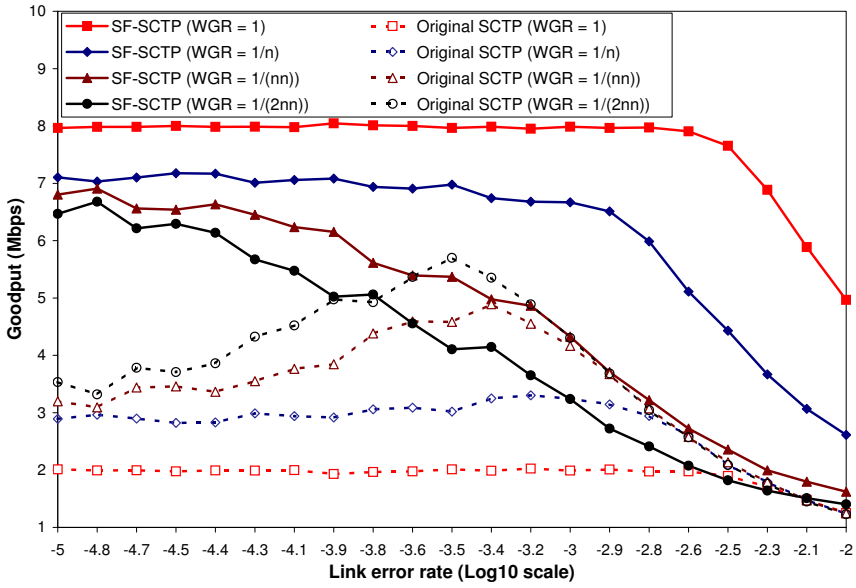


(a) Goodput.

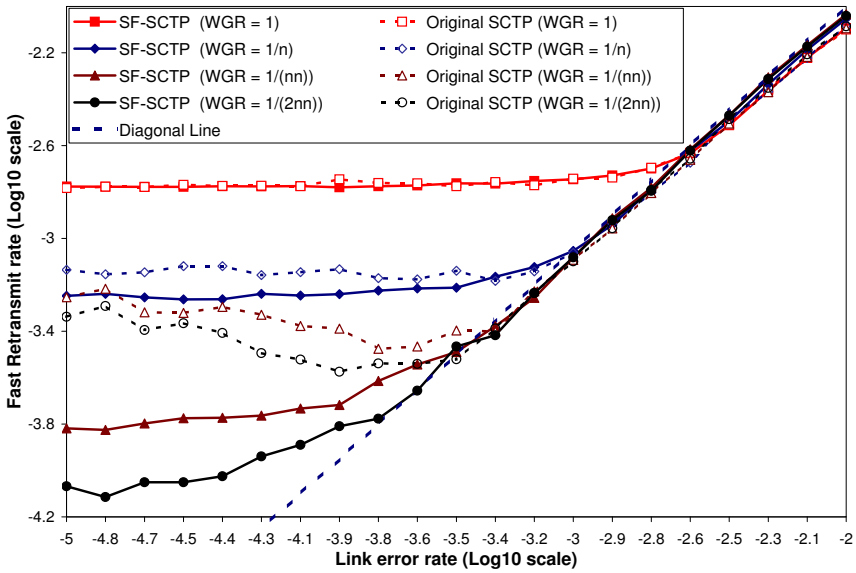


(b) Fast Retransmit rate.

Figure 5.40: Experiment IXX(b): SF-SCTP competing with the same number of single SCTP flows.

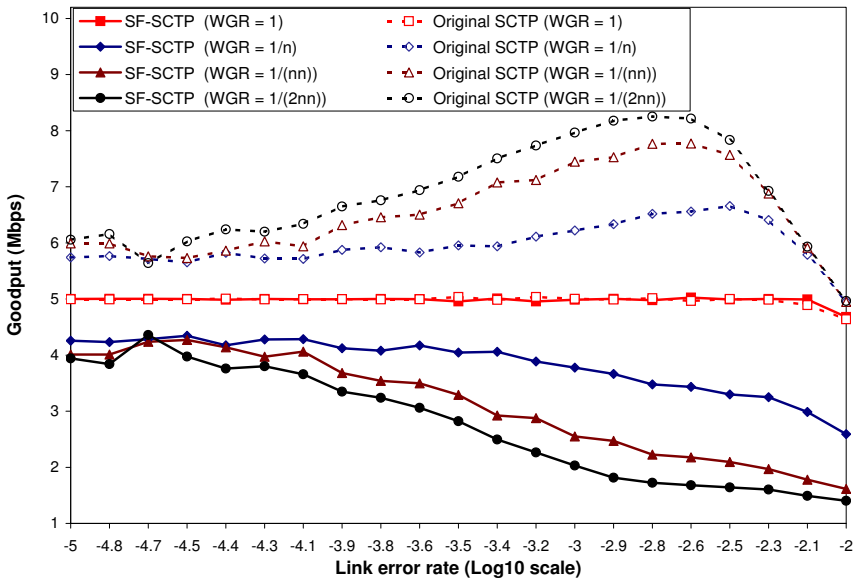


(a) Goodput.

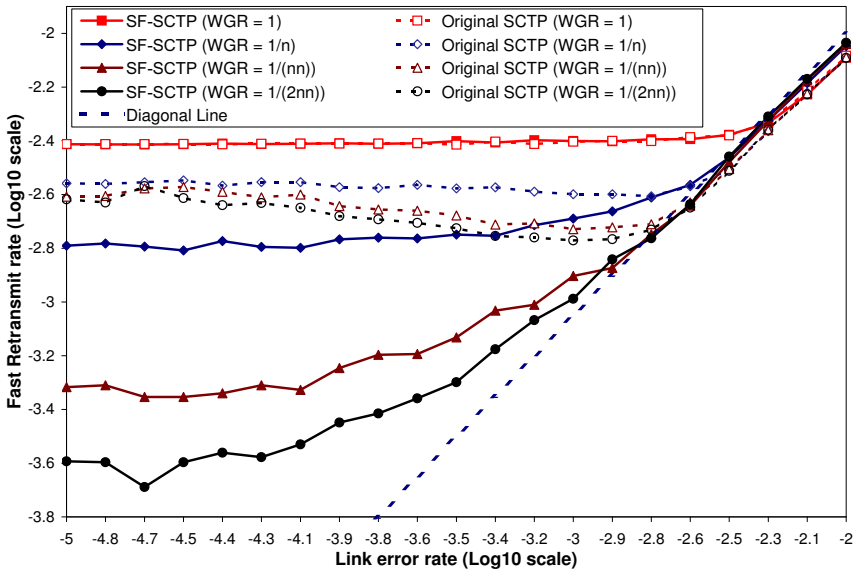


(b) Fast Retransmit rate.

Figure 5.41: Experiment XX(a): SF-SCTP competing with the same number of single SCTP flows.

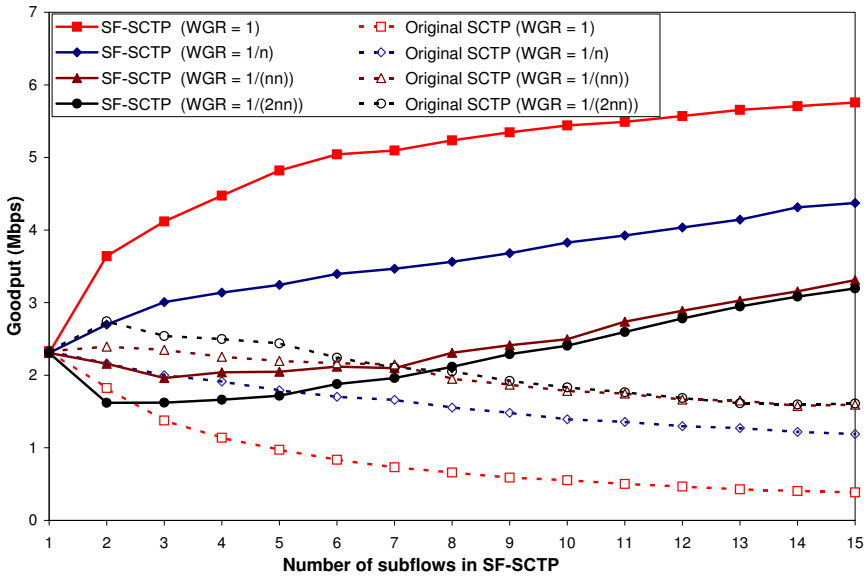


(a) Goodput.

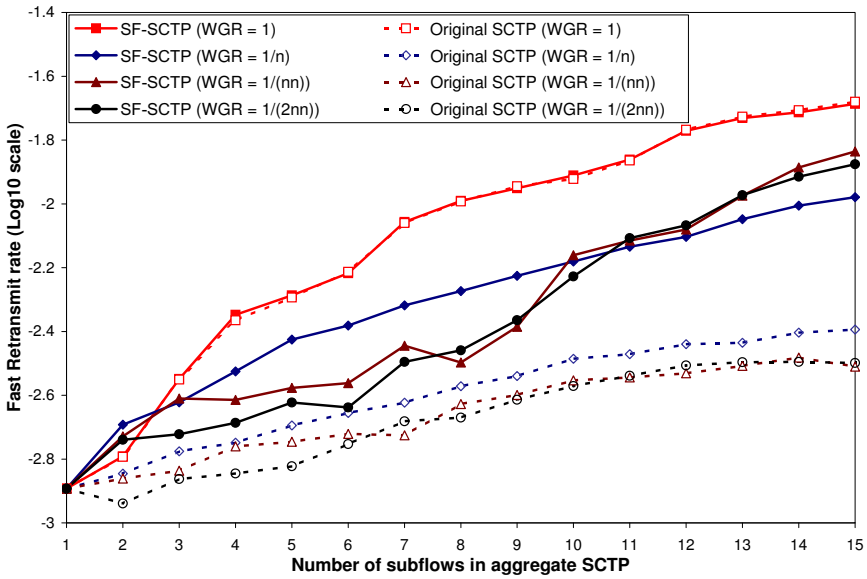


(b) Fast Retransmit rate.

Figure 5.42: Experiment XX(b): SF-SCTP competing with the same number of single SCTP flows.

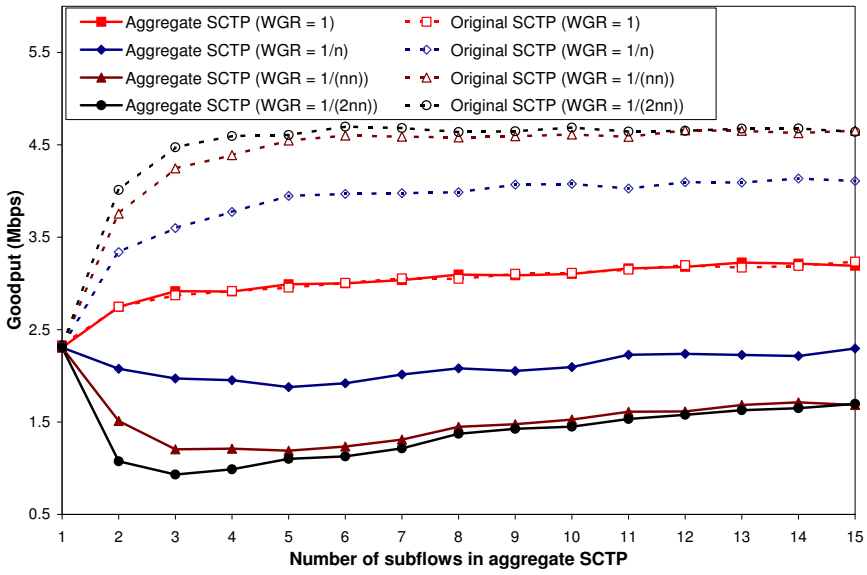


(a) Goodput.

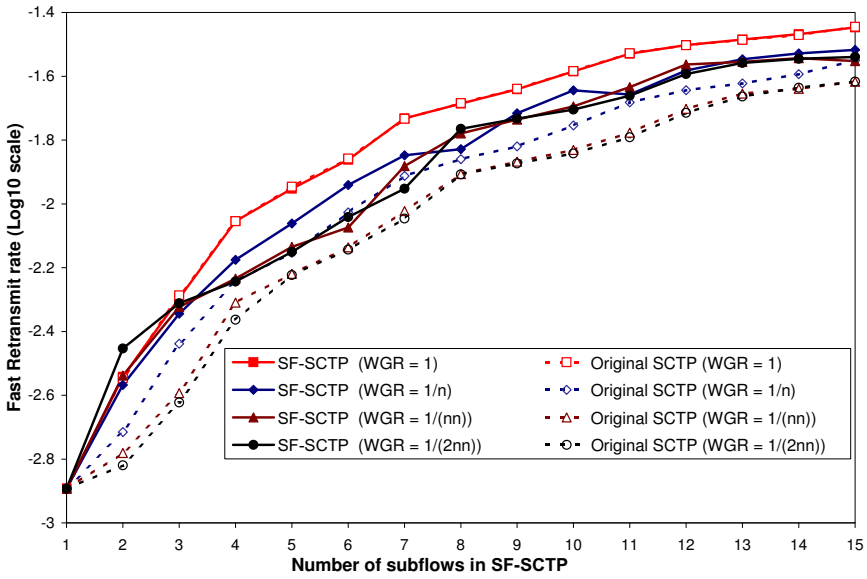


(b) Fast Retransmit rate.

Figure 5.43: Experiment XXI(a): SF-SCTP competing with the same number of single SCTP flows.



(a) Goodput.



(b) Fast Retransmit rate.

Figure 5.44: Experiment XXI(b): SF-SCTP competing with the same number of single SCTP flows.

Chapter 6

Conclusions and Future Work

With multi-streaming, SCTP is equipped with the capability to support concurrent transmission of messages with different QoS requirements. However, when SCTP was originally introduced, supporting QoS inside an SCTP association was not a concern. In this thesis, we proposed Subflow-Capable SCTP (SF-SCTP), which modified the current SCTP specification to support preferential treatment of SCTP streams.

In our modified SF-SCTP, streams are grouped into subflows based on their QoS requirements, and only messages coming from the same subflow could be bundled together. In SF-SCTP, flow and congestion control have been migrated from the association level to subflow level to avoid false sharing or any other possible interactions between subflows. With these modifications, a subflow can be regarded as the carrier of user messages which require orderly transmission and the same type of QoS from the network. The number of

subflows an SCTP association can open depends on the many types of QoS requested by an SCTP application. With respect to data transmission regulation, subflows are independent since each one of them implements its own flow and congestion control. Subflows also have their own transmitting and receiving buffers from an SCTP communication peer. Some common functionalities, such as association startup, maintenance, teardown and path monitoring are shared by all subflows and provided at the association level.

By splitting an SCTP association into several subflows, we initially make SF-SCTP's congestion window behavior much more aggressive than the original SCTP. To alleviate SF-SCTP's over-aggressiveness, we further incorporate fractional congestion control (FCC). FCC reduces SF-SCTP's aggressiveness by restraining a subflow's congestion window growth rate, while preserving SF-SCTP's effectiveness in utilizing network bandwidth by reducing network packet drop rate.

Compared to the Original SCTP, our SF-SCTP has the several advantages. (1) With the subflow level design of flow and congestion control, SF-SCTP is able to provide preferential treatment among its streams and take advantage of the Diff-Serv network by avoiding the problem of false sharing; (2) Having multiple subflow inside, SF-SCTP is similar to the aggregation of SCTP associations and is very effective in utilizing network bandwidth, which presents SF-SCTP as an ideal transport layer candidate for grid and high speed computing community; (3) With FCC integrated, SF-SCTP can alleviate the global synchronization problem of Drop-Tail queue, reduce the network drop rate, while

being fair to other TCP-friendly network traffic.

We derive analytic stochastic models for four cases of SCTP implementations. Our stochastic models effectively capture the congestion control behavior of SCTP and have been verified by simulations that, given the packet drop rate and RTT of a network, can accurately predict the throughput of SCTP implementations. Our models improve the existing TCP models and are suitable for both Drop-Tail and RED queues, for different packet sizes, and also for different networks such as wide area networks, single bottleneck networks and Diff-Serv networks. In addition to the stochastic models which take packet drop rate and RTT as a input parameters, for a single bottleneck network with Drop-Tail queue where query delay plays a large factor, we also derive deterministic models which define the lower and upper boundaries for the throughput and packet drop rates.

Extensive simulation experiments have been performed to study SF-SCTP's capability to support QoS in a Diff-Serv network, the behavior of SF-SCTP under various network conditions, and the effect of fractional congestion control had on SF-SCTP. Our simulations confirm the capability of SF-SCTP to support preferential treatment among its subflows by avoiding the problem of false sharing and by dynamically adjusting its subflow throughput to network packet drop rate and Round Trip Time fluctuation. We especially studied in detail the behavior of SF-SCTP in single bottleneck network. Those simulations reveal SF-SCTP's aggressive nature in exploring network bandwidth. Compared to the original SCTP, a SF-SCTP with n subflows and without FCC opens its congestion

window n times faster, while reducing its congestion window by $1/n$ times slower. The simulations which compare SF-SCTP with various network background traffic further illustrate SF-SCTP's capability to seize the available network bandwidth. Our simulations for SF-SCTP with FCC show that, for a network under congestion, FCC effectively reduces SF-SCTP aggressiveness, making it much more fair to other TCP-friendly traffics. Our simulation results even show that original SCTP, in a single bottleneck network with Drop-Tail queue, benefits from the presence of SF-SCTP with FCC because FCC alleviates the problem of Global Synchronization, reduces the network packet drop rate and improves the overall network usage. The results from the simulations also verified the accuracy of our analytic models.

The work presented in this thesis opens many opportunities for future research. The SF-SCTP with FCC can be applied to wireless networks where link errors may dominate. We expect that SF-SCTP shall greatly improve a wireless network's utilization by having multiple subflows and smooth congestion window. Given the multi-homing feature of SCTP, SF-SCTP design can be extended so that it can support QoS and load balancing. Although currently SF-SCTP provides no cooperation among the subflows it manages, it makes sense to investigate possible cooperation mechanisms among subflows that may help further improve SF-SCTP throughput. The analytic models presented in this thesis mainly captured the Fast Retransmission period of SF-SCTP congestion control behavior; more complex analytic models can be derived to capture the Slow Start and Time-Out

periods. FCC has many advantages, however, it retrains SF-SCTP for better throughput in a network with plenty of bandwidth. Improvements can be made to FCC mechanism to make it more intelligent to limit SF-SCTP throughput only in a congested network.

References

[1] A. Abd El Al, T. Saadawi, and M. Lee. Improving throughput and reliability in mobile wireless networks via transport layer bandwidth aggregation. In Parr et al., eds, *Military Communications Systems and Technologies*, *[Elsevier] Comput. Netw.* **46**(5), pp. 635–649. 2004.

[2] ———. LS-SCTP: A bandwidth aggregation technique for Stream Control Transmission Protocol. In Langendoerfer and Tsaoussidis, eds, *Protocol Engineering for Wired and Wireless Networks*, *[Elsevier] Comput. Commun.* **27**(10), pp. 1012–1024. 2004.

[3] A. Akella, H. Balakrishnan, and S. Seshan. The impact of false sharing on shared congestion management. In *Proc. IEEE Int'l Conf. Netw. Protocols (ICNP)*, pp. 84–94, Atlanta, GA, 2003.

[4] B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, Ĩ. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke. Data management and transfer in high-performance computational grid environments. *Parallel Comput.* **28**(5), pp. 749–771, 2002.

[5] W. Allcock, J. Bester, J. Bresnahan, S. Meder, P. Plaszczak, and S. Tuecke. GridFTP: Protocol extensions to FTP for the grid. Technical Report GFD.20, Argonne National Laboratory, 2003.

[6] P. Almquist. Type of service in the internet protocol suite. RFC 1349, IETF, 1992.

[7] M. Atiquzzaman and W. Ivancic. Evaluation of SCTP multistreaming over satellite links. In *Proc. IEEE Int'l Conf. Comput. Commun. Netw. (IC3N)*, Dallas, TX, 2003.

[8] F. Baker, K. Chan, and A. Smith. Management information base for the differentiated services architecture. RFC 3289, IETF, 2002.

[9] H. Balakrishnan, H.S. Rahul, and S. Seshan. An integrated congestion management architecture for Internet hosts. In *Proc. ACM SIGCOMM*, pp. 175–187, Cambridge, MA, 1999.

[10] A. Balk, M. Sigler, M. Gerla, and M.Y. Sanadidi. Investigation of MPEG-4 video streaming over SCTP. In *Proc. IIIS World Multi-Conf. System. Cybern. Inform. (SCI)*, Orlando, Florida, 2002.

[11] N. Biswajit. Diffserv model for the NS-2 simulator. Technical report, Nortel Networks, 2000. (http://www7.nortel.com:8080/CTL/software).

[12] D. Black. Differentiated services and tunnels. RFC 2983, IETF, 2000.

[13] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. RFC 2475, IETF, 1998.

[14] R. Brennan and T. Curran. SCTP congestion control: Initial simulation studies. In *Proc. of 17th Int'l Teletraffic Congress*, Salvador, Brazil, 2001.

[15] A. Caro. The ns manual, chapter 33: Sctp agents. (www.isi.edu/nsnam/ns/nsdocumentation.html.).

[16] ———. End-to-end fault tolerance using transport layer multihoming. Phd disser-

tation, CISC Dept, Univ of Delaware, 2005.

[17] A. Caro, P. Amer, P. Conrad, and G. Heinz. Improving multimedia performance over lossy networks via SCTP. Technical report, Fifth Advanced Telecomunications and Information Distribution Research Program (ATIRP) Conference (Army Research Laboratory), College Park, MD, 2001.

[18] A. Caro, P. Amer, and R. Stewart. Transport layer multihoming for fault tolerance in FCS networks. In *In 1st Annual CTA Symposium*, College Park, MD, 2003.

[19] ———. End-to-end failover thresholds for transport layer multihoming. In *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Monterey, CA, 2004.

[20] A. Caro and J. Iyengar. NS-2 SCTP module, version 3.2, 2002. (http://pel.cis.-edu.edu).

[21] A. Caro, J. Iyengar, P. Amer, G. Heinz, and R. Stewart. A two-level threshold recovery mechanism for SCTP. In *SCI*, Orlando, FL, 2002.

[22] ———. Using SCTP multihoming for fault tolerance and load balancing. In *Proc. ACM SIGCOMM*, Pittsburgh, PA, 2002.

[23] A.L. Caro, J.R. Iyengar, P.D. Amer, S. Ladha, G.J. Heinz, and K.C. Shah. SCTP: A proposed standard for robust Internet data transport. *IEEE Computer* **36**(11), pp. 56–63, 2003.

[24] K. Chan, R. Sahita, S. Hahn, and K. McCloghrie. Differentiated services quality of service policy information base. RFC 3317, IETF, 2003.

[25] D. Clark and W. Fang. Explicit allocation of best-effort packet delivery service. *IEEE/ACM Trans. Netw.* **6**, pp. 362–373, 1998.

[26] L. Coene. Stream control transmission protocol applicability statement. RFC 3257, IETF, 2002.

[27] Kevin Fall and Sally Floyd. Simulation-based comparisons of Tahoe, Reno and SACK TCP. *Computer Communication Review* **26**(3), pp. 5–21, 1996.

[28] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Trans. Netw.* **7**, pp. 458–472, 1999.

[29] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.* **1**(4), pp. 397–413, 1993.

[30] S. Floyd, J. Mahdavi, M. Mathis, M. Podolsky, and A. Romanow. An extension to the selective acknowledgement (sack) option for tcp. RFC 2018, IETF, 1999.

[31] T.J. Hacker, B.D. Athey, and B.D. Noble. The end-to-end performance effects of parallel TCP sockets on a lossy wide-area network. In *Proc. IEEE/ACM Int'l Parall. Distrib. Process. Symp. (IPDPS)*, Fort Lauderdale, Florida, 2002.

[32] T.J. Hacker, B.D. Noble, and B.D. Athey. The effects of systemic packet loss on aggregate TCP flows. In *Proc. ACM/IEEE Conf. Supercomput.*, Baltimore, MD, 2002.

[33] ———. Improving throughput and maintaining fairness using parallel TCP. In *Proc. IEEE INFOCOM*, pp. 2480–2489, Hong Kong, 2004.

[34] A. Hanushevsky, A. Trunov, and L. Cottell. Peer-to-peer computing for secure high performance data copying. In *Proc. of Int'l Conf. on Comput. in High Energy and Nuclear Physics*, Beijing, China, 2001.

[35] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured forwarding PHB group. RFC 2597, IETF, 1999.

[36] J. Heinanen and R. Guerin. A two-rate three-color marker. RFC 2698, IETF, 1999.

[37] J. Iyengar. Concurrent multipath transfer using sctp multihoming. Phd dissertation, CISC Dept, Univ of Delaware, 2006.

[38] J. Iyengar, P. Amer, and R. Stewart. Concurrent multipath transfer using sctp multihoming. Technical report, CISC Dept, Univ of Delaware, 2003.

[39] ———. Concurrent multipath transfer using transport layer multihoming: Performance under varying bandwidth proportions. In *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Monterey, CA, 2004.

[40] J. Iyengar, P. Amer, R. Stewart, and I. Rodriguez. Preventing SCTP congestion window overgrowth during changeover. Internet draft, IETF, 2002. [work in progress].

[41] J. Iyengar, A. Caro, P. Amer, G. Heinz, and R. Stewart. Making SCTP more robust to changeover. Technical report, CISC Dept, Univ of Delaware, 2002.

[42] ———. SCTP congestion window overgrowth during changeover. In *SCI*, Orlando, FL, 2002.

[43] J. Iyengar, K. Shah, P. Amer, and R. Stewart. Concurrent multipath transfer using SCTP multihoming. In *Proc. SCS Int'l Symp. Perf. Eval. Comput. Telecommun. Syst. (SPECTS)*, San Jose, CA, 2004.

[44] V. Jacobson. Congestion avoidance and control. In *Proc. ACM SIGCOMM*, pp. 314–329, Stanford, CA, 1988.

[45] Y. Krishnamurthy, I. Pyarali, C. Rodrigues, P. Manghwani, and G. Thaker. Using SCTP to improve qos and network fault-tolerance of dre systems. In *OMG Real-Time and Embedded Systems Workshop*, Reston, Virginia, 2004.

[46] S. Ladha and P. Amer. Improving multiple file transfers using SCTP multistreaming. In *Proc. IEEE Int'l Perform. Comp. Comm. Conf. (IPCCC)*, Phoenix, Arizona, 2004.

[47] J. Lee, D. Gunter, B. Tierney, B. Allcock, J. Bester, J. Bresnahan, and S. Tuecke. Applied techniques for high bandwidth data transfers across wide area networks. In *Proc. of Int'l Conf. on Comput. in High Energy and Nuclear Physics*, Beijing, China, 2001.

[48] M. Marwah. ST-TCP: Tcp layer server fault tolerance. In *IEEE Int. Conf. on Dependable Systems and Networks*, San Francisco, CA, 2003.

[49] M. Mathis, J. Semke, and J. Mahdavi. The macroscopic behavior of the TCP congestion avoidance algorithm. *ACM Comput. Commun. Rev.* **27**(3), 1997.

[50] M. Molteni and M. Villari. Using SCTP with partial reliability for MPEG-4 multimedia streaming. In *Proc. of BSDCon Europe*, Amsterdam, Netherlands, 2002.

[51] P. Natarajan, J. Iyengar, P. Amer, and R. Stewart. Sctp: An innovative transport layer protocol for the web. In *WWW 2006*, Edinburgh, Scotland, 2006.

[52] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the differentiated services field (ds field) in the ipv4 and ipv6 headers. RFC 2474, IETF, 1998.

[53] T. Ott, J. Kemperman, and M. Mathis. The stationary behavior of ideal TCP congestion avoidance. Technical report, Telcordia Technologies, Inc., 1997. ftp://ftp.bellcore.com/pub/tjo/TCPwindow.ps.

[54] J. Padhye, V. Firoiu, and D. Towsley. A stochastic model of tcp reno congestion avoidance and control. Technical report, CMPSCI Technical Report 99-02, University of Massachusetts, MA, 1999.

[55] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *Proc. ACM SIGCOMM*, pp. 303–314, 1998.

[56] C. Parsa and J. J. Garcia-Luna-Aceves. Improving TCP congestion control over internets with heterogeneous transmission media. In *Proc. IEEE Int'l Conf. Netw. Protocols (ICNP)*, pp. 213–221, Toronto, ON, 1999.

[57] R. Rajamani, S. Kumar, and N. Gupta. SCTP versus TCP: Comparing the performance of transport protocols for web traffic. Technical report, University of Wisconsin-Madison, 2002.

[58] T. Ravier, R. Brennan, and T. Curran. Experimental studies of SCTP multihoming. In *First Joint IEI/IEE Symposium on Telecommunications Systems Re-*

search, Dublin, Ireland, 2001.

[59] S. Samtani, J.R. Iyengar, and M.A. Fecko. SCTP multistreaming: Preferential treatment among streams. In *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, pp. 966–970, Boston, MA, 2003.

[60] H. Sivakumar, S. Bailey, and R. L. Grossman. PSockets: The case for application-level network striping for data intensive applications using high speed wide area networks. In *Proc. of SuperComput.: High-Performance Networking and Comput.*, 2000.

[61] W. Stevens. *TCP/IP Illustrated, Vol.1 The Protocols*. Addison-Wesley, 1994.

[62] ———. Tcp slow start, congestion avoidance, fast retransmit, and fast recovery algorithms. RFC 2001, IETF, 1997.

[63] R. Stewart and P. Amer. Why is SCTP needed given TCP and UDP are widely available? In *Internet Society Member Briefing 17*, 2004.

[64] R. Stewart and C. Metz. SCTP: New transport protocol for TCP/IP. *IEEE Internet Comput.* **5**(6), pp. 64–69, 2001.

[65] R. Stewart, L. Ong, I. Arias-Rodriguez, K. Poon, and A. Caro. Stream control transmission protocol (SCTP) partial reliability extension. Internet draft, IETF, 2002.

[66] R. Stewart, L. Ong, I. Arias-Rodriguez, K. Poon, P. Conrad, A. Caro, and M. Tuexen. Stream Control Transmission Protocol (SCTP) implementer's guide. Internet draft, IETF, 2002.

[67] R. Stewart, M. Ramalho, Q. Xie, M. Tuexen, and P. Conrad. Stream control transmission protocol (SCTP) partial reliability extension. RFC 3758, IETF, 2004.

[68] R. Stewart and Q. Xie. *Stream Control Transmission Protocol (SCTP): A Reference Guide*. Addison-Wesley, 2001.

[69] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. Stream Control Transmission Protocol. RFC 2960, IETF, 2000.

[70] UC Berkeley and LBL and USC/ISI and Xeror Parc. *NS-2 documentation and software, Version 2.28*, 2005. (http://www.isi.edu/nsnam/ns).

[71] I. Yeom and A.L.N. Reddy. Modeling TCP behavior in a differentiated services network. *IEEE/ACM Trans. Netw.* **9**(1), pp. 31–46, 2001.

[72] Z. Yi, T. Saadawi, and M. Lee. Analytic model of Stream Control Transmission Protocol. In *Proc. Int'l Wksp Perform. Model. Comput. Commun. Syst. (PMCCS)*, Monticello, IL, 2003.

[73] J. Zou, M.U. Uyar, M.A. Fecko, and S. Samtani. Preferential treatment of SCTP subflows: Analysis and simulation. In *Proc. IEEE Int'l Symp. Comput. Commun. (ISCC)*, pp. 810–815, Alexandria, Egypt, 2004.

[74] ———. SCTP subflows for survivable FCS applications. In *Battlespace Digitization and Network-Centric Systems IV*, *Proc. SPIE* **5441**, pp. 192–203. (SPIE, Bellingham, WA), 2004.

[75] ———. Supporting QoS and maintaining fairness for SCTP with parallel subflows. In *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Atlantic City, NJ, 2005.

[76] ———. Integrating Fractional Congestion Control into Subflow Capable SCTP design. In *Proc. IEEE Sarnoff Symp. Adv. Wired Wirel. Netw.*, Princeton, NJ, 2006.

[77] ———. Performance evaluation of subflow capable SCTP. *[Elsevier] Comput. Commun.* **29**(14), pp. 2413–2432, 2006.

[78] ———. SF-SCTP: a new transport protocol to support QoS for FCS applications. In *Int'l Society for Optical Engineering*, Orlando, FL, 2006.

[79] ———. SF-SCTP: An extension of Stream Control Transmission Protocol to support QoS. In *IEEE International Conference On Networking, Sensing and Control*, Ft. Lauderdale, FL, 2006.

[80] ———. Throughput models for SCTP with parallel subflows. *[Elsevier] Comput. Netw.* **50**(13), pp. 2160–2182, 2006.