UNIVERSITY WEST

2010-09-01

Performance of Network Redundancy in SCTP

- Introducing effect of different factors on Multi-homing

Rashid Ali

THESIS PROJECT
Master program in Computer science

MASTER THESIS

Performance of Network Redundancy in SCTP - Introducing effect of different factors on Multi-homing

Abstract

The main purpose of designing the Stream Control Protocol (SCTP) was to offer a robust transfer of traffic between the hosts over the networks. For this reason SCTP multi-homing feature was designed, in which an SCTP sender can access destination host with multiple IP addresses in the same session. If the primary path between the source and the destination is down, the traffic may still be sent to the destination by utilizing redundant path. And SCTP multi-homing also supports for the concurrent multipath transfer of traffic. This paper introduces the effect of different network factors like concurrent cross traffic, congestion control algorithms and SACK timers on multi-homing feature of SCTP. Throughput and end-to-end packet delay were used as performance metrics to introduce the effect of these factors. From the study it was introduced that concurrent cross traffic in the network behaves same on multi-homed interfaces and both interfaces were affected almost same. It was concluded that congestion control algorithms also affects on multi-homing, the RED congestion control algorithm reduced delay and improved throughput of the SCTP multi-homing. In RFC4960 recommended SACK timer is 200ms, but when 100ms SACK timer was used with concurrent multipath transfer in SCTP (CMT-SCTP) multi-homing, the high throughput and low delay was achieved as compared with 200ms and 300ms, which indicated that different SACK timers affects on multi-homing feature of SCTP. All the simulation works have been conducted in NS2 network simulator.

Date:	September 01, 2010
Author:	Rashid Ali
Examiner:	Stefan Christiernin
Advisor:	Stanislav Belenki
Program:	Master Program in Computer Science
Main field of study:	Advance Network Design
Education level:	second cycle
Credits:	60 HE credits
Course code:	EXD908
Keywords	Transmission Transport Layer Protocols, SCTP, Multi-homing, NS2
Publisher:	University West, Department of Economics and IT SE-461 86 Trollhättan, SWEDEN Phone: + 46 520 22 30 00 Fax: + 46 520 22 32 99 Web: www.hv.se

Preface

Stream Control Transmission Protocol (SCTP) is a reliable transport protocol operating on top of an unreliable protocol that is internet protocol (IP). For the past twenty years the reliable transmission is provided by TCP and unreliable data delivery is provided by UDP. Many of the features of TCP and UDP can also be found in SCTP. For this reason SCTP multi-homing feature was designed, in which an SCTP sender can access destination host with multiple IP addresses in the same session. If the primary path between the source and the destination is down, the traffic may still be sent to the destination by utilizing redundant path. This paper aims to determine the impact of concurrent cross traffic in network, congestion control algorithms and default SACK timers on the multi-homing feature of SCTP. A pre-study for this Master these report was performed in the subject "Managing Computer Science Projects (PDD900)" offered by University West. In this subject a complete time schedule was designed for the achievement of the goals of this paper. According to that time plan a background study for SCTP and overview of its multi-homing feature was performed in last session. For this purpose RFCs for TCP, UDP and SCTP, multiple research papers were under study. Before selecting this master final theses I was unknown to Linux environment and using Network Simulator NS2 for network researches so I learned these tools from scratch during this study, then finally during the time period of master theses different simulation experiments were performed in Network Simulator NS2 to introduce the effect of assumed factors on the multi-homing feature of SCTP.

I would like to make a good use of this page to thanks all people that helped me to complete this final these. First of all in the name of Almighty Allah Who enabled me to complete this project. Let us start from supervisor Prof. Dr. Stanislav Belenki who was abundantly helpful and he offered invaluable assistance, support and guidance. I also wish to express my deep gratitude the member of final master theses subject especially to Prof. Dr. Stefan Christiernin and Prof. Dr. Linn Gustavsson Christier. A special thanks to research assistant Nasif Ekniz at P.E.L labs at University of Delaware for his immediate help regarding SCTP implementations in NS2. Finally special thanks to all my friends and people who directly and indirectly helped me during this theses study. And at last but not least, I would like to thank my parents and siblings for everything they have done during these last 26 years.

Trollhättan, Sweden

September the 01, 2010

Rashid Ali

Performance of Network Redundancy in SCTP
(Introducing effect of different factors on Multi-homing)

Rashid Ali
Master Program in Computer Science
University West
P.O.Box 975 SE-46129 Trollhättan Sweden
E-mail: rashid.ali2993@gmail.com

Abstract—--- **The main purpose of designing the Stream Control Protocol (SCTP) is to offer a robust transfer of traffic between the hosts over the networks. For this reason SCTP multi-homing feature is designed, in which an SCTP sender can access destination host with multiple IP addresses in the same session. If the primary path between the source and the destination is down, the traffic may still be sent to the destination by utilizing redundant path. And SCTP multi-homing also supports for the concurrent multipath transfer of traffic. This paper introduces the effect of different network factors like concurrent cross traffic, congestion control algorithms and SACK timers on multi-homing feature of SCTP. Throughput and end-to-end packet delay are used as performance metrics to introduce the effect of these factors. From the study it is introduced that concurrent cross traffic in the network behaves same on multi-homed interfaces. It is concluded that congestion control algorithms also affects on multi-homing, the RED congestion control algorithm reduces delay and improves throughput of the SCTP multi-homing. In RFC4960 recommended SACK timer is 200ms, but when 100ms SACK timer is used with concurrent multipath transfer in SCTP (CMT-SCTP) multi-homing, the high throughput and low delay is achieved as compared with 200ms and 300ms, which indicates that different SACK timers affects on multi-homing feature of SCTP. All the simulation works have been conducted in NS2 network simulator.**

Keywords **Transport Layer Protocols, SCTP, Multi-homing, NS2**

1. INTRODUCTION

SCTP is a reliable transport layer protocol which operates on an unreliable packet service that is internet protocol (IP). The SCTP is defined in RFC 4960 [2]._For the past twenty years, Transport Control Protocol [18] has provided reliable communication and the unreliable data delivery is provided by User Datagram Protocol [17]. Many of the features of the TCP and UDP can also be found in SCTP. It provides acknowledged error-free non-duplicated transfer of data. SCTP discovers path maximum transmission unit (MTU) for data transmission. It can bundle multiple user messages into a single SCTP packet. SCTP also provides a network-level fault tolerance through the support of multi-homing at either ends or both ends of the association [6]. Flow and congestion control in SCTP have been designed to assure that its traffic behaves similar to the TCP traffic, it allows SCTP for seamless introduction into existing IP networks [4]. Further sections include more details about SCTP and its features.

The objective of this paper is to provide a simulated study to show the performance of network redundancy (Multi-homing) in SCTP by introducing effect of different factors on multi-homing. This paper aims to determine the impact of concurrent cross traffic, different SACK (Selective Acknowledgment) timer values and the congestion control algorithms on the multi-homing feature of SCTP. The rest of the paper will be laid out as follows: In the remaining of this section research questions are given which will be covered throughout this paper. Section-2 will provide a background study of SCTP with important features and terminologies. In section-3 the paper will get into some of the overview of multi-homing feature of SCTP. The section-4 contains experimental design including scope of the project, requirements for experimental design and different network scenarios for experiments. In the section-5 results obtained from the experimental design are analyzed. Finally in the section-6 the paper will close with brief review and conclusion, it will also provide a necessary future work.

The research questions of the paper are to come up the simulated results showing performance of network redundancy (Multi-homing) in SCTP by introducing the effect of concurrent cross traffic, congestion control algorithms and different SACK timers on multi-homing feature of SCTP.

2. BACKGROUND OF SCTP

The Stream Control Transmission Protocol was designed by the SIGTRAN (Signaling Transport) working group of IETF (Internet Engineering Task Force). It was designed for the purpose of transporting of real time signaling over IP networks. For many years SS7 has been the only bearer for the signaling traffic in telecommunication networks [3]. Within SIGTRAN, special attention has been given to a scenario when telecommunication signaling is transported between a Signaling Gateway (SG) and a Media Gateway Controller (MGC) [4].

Besides that SCTP can be directly used in IP networks, in-fact SCTP has provided many of its properties which are alternatives to established Transport protocols TCP (Transmission Control Protocol) and UDP (User Datagram Protocol).

2.1 Features of SCTP

The UDP is a connectionless transport layer protocol, which means that it cannot provide error control and flow control, which are two vital properties of a transport layer to support the real time signaling information. SCTP provides un-order data delivery and preservation of message boundaries like UDP while it's a connection oriented protocol. TCP is a byte stream protocol and SCTP transfers data in chunks, the SCTP discovers PATH-MTU, in PATH-MTU discovery it determines the maximum possible packet size that can be sent to a destination without being segmented. SCTP segments large data into chunks which fit into such a maximum transmission unit (IP packet). Several small chunks resulting from messages may be multiplexed into one IP packet (Bundling). The TCP transmits data strictly in-sequence per connection, whereas SCTP has more flexible data transmission scheme, it transfers data messages in different distinguished streams within one SCTP association. In which message sequence is only maintained per stream to reduce the un-necessary head-of-line blocking among independent streams.

SCTP also provides an exponential back-off algorithm similar to the TCP in order to provide the adaptive service with the other end during the transmission [12]. Another feature of SCTP is a solution for Denial of Service attacks, also known as the blind attacks, which can easily attack on TCP during handshaking. The SCTP four-way handshaking is designed to overcome such type of attacks. It uses cookie fields during handshaking which contain signature for authentication and a time string to prevent replay attacks using cookies [5]. Some important features of SCTP are briefly described in the next sections.

Multi-homing

One of the key features of SCTP is the support for multi-homing that is the destination nodes can be reached under the several IP addresses (multi-homed). A TCP connection is defined by a pair of transport addresses IP address and port number while In SCTP both sides of the association provides multiple IP addresses combined with a single SCTP port number. Thus each multi-homed node can be reached from another node using several paths. The multi-honing facility makes SCTP more robust against partial network failures than TCP [4].

Multi-Streaming

Another key feature of SCTP is multi-streaming. It refers to the parallel transmission of messages over the same association between sender and the receiver. The stream independently carries fragmented messages from one end to the other. In TCP connection all bytes transmitted are delivered in strict transmission order. It is possible that protocol wastes its bandwidth due to the strict sequences of message delivery. The independent transmission of streams gives an advantage to SCTP over TCP and it achieves a cumulative throughput [12].

Security

In the modern era internet where security is a big issue a transport layer protocol that is subject to easy attack is not

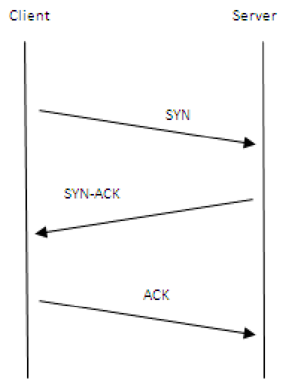


Figure 2.1 TCP Three-Way hand-Shaking

acceptable. SCTP has its unique features that increase its security against blind attacks like Denial of Service attacks. The TCP and SCTP both initiate a connection with a handshake between sender and receiver. TCP uses three-way handshaking to set up a new connection, whereas SCTP uses a four-way handshake. The figure 2.1 shows the three-way handshake process of TCP. The Client sends a SYN (Synchronize) packet to Server, upon receiving the SYN packet Server allocates the resources for the connection and sends a SYN-ACK (Synchronize Acknowledgment) packet to Client. The Client than sends an ACK (Acknowledgment) packet to confirm the receipt of SYN-ACK packet. Now the connection between Client and Server is setup, the Client can start sending data. The TCP three-way handshaking is insecure because an attacker can send Denial of Service attack during SYN packet exchange.

The SCTP four-way hand-shake is designed to overcome such type of attacks. Figure 2.2 shows four-way hand-shaking of SCTP. In the SCTP four-way hand-shake Client initiates an association by sending an INIT packet to the Server. Server

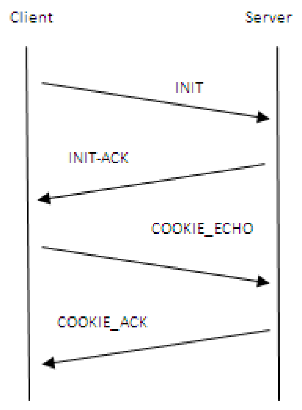


Figure 2.2 SCTP Four-Way hand-shaking

responds with an INIT-ACK packet to the Client that contains the verification tag and cookie field. The TCP SYN-ACK does not contain these fields. The cookie contains the necessary state information which the Server uses to allocate the resources for the association. The cookie field includes a signature for authenticity and a time string to prevent replay attacks using cookies. Unlike TCP the Server does not allocate resources to the Client at this stage. The verification tag provides a key that enables Client to verify that this packet belongs to the current SCTP association. After receiving INIT-ACK packet, the Client sends COOKIE-ECHO packet to the Server, if Client has a forged IP address it will never receives the INIT-ACK chunk. This prevents Client from sending COOKIE-ECHO packet. As a result the conservation ends without the server allocating any resources for the connection. After receiving COOKIE-ECHO Server sends COOKIE-ACK chunk to the Client and allocates resources for the connection.

Now the SCTP association is established between Client and Server. It seams that in the SCTP the transfer of data may be delayed due to the additional handshake. To overcome this delay SCTP permits data to be exchanged during handshake in the COOKIE-ECHO and the COOKIE-ACK chunks [5].

2.2 Important Terminologies of SCTP

Following are some important terminologies of Stream Control Transmission Protocol (SCTP).

Chunks

The protocol data unit (PDU) of SCTP is called SCTP packet, which is the actual payload of IP packet. An SCTP packet is composed of chunks. Chunk is the minimal data unit that can be transmitted in SCTP stream. Multiple chunks of data may be multiplexed into a single packet up to the calculated MTU size of the path. A chunk is either a control data or user data. In SCTP a chunk can be used to control association, to test validity of the path and to discover network events like failure, disconnection and half-open connection.

SCTP Association

The connection establishment between two nodes after four-way handshaking is called association. An association in SCTP is similar to the connection in a TCP. When an association is established now an SCTP data application is able to transmit data to the destination end. During the creation of association various information are exchanged between the sender and receiver [6].

Selective Acknowledgments (SACKs)

The loss and duplication of data chunks is performed by numbering all data chunks with a Transport Sequence Number (TSN) in the sender. The acknowledgments sent by receiver are based on the TSNs. The retransmission of the data is timer controlled, the timer value is derivative of continues measurements of the Round Trip delay. After the expiration of retransmission timer, unacknowledged data chunks are resent and the timer is started again. When the receiver end detects gaps in the sequence of the received chunks, the each received SCTP chunk is acknowledged by sending a specific "Selective Acknowledgment" (SACK) control chunks which report all gaps. Whenever the sender receives four consecutive gap chunks from the receiver for the same data chunk, this data chunk is immediately retransmitted over the multi-homed paths. Most up-to-date operating systems support a similar optional extension to TCP to achieve such type of robustness [4].

Termination

The SCTP provides termination of its association, while TCP supports half-open connection. In the SCTP established association ABORT messages are used to cleanly terminate the rest of the associations that still alive. Normally an association is terminated by starting SHUTDOWN messages [12].

3. Overview of SCTP Multi-homing

The applications that require high degree of fault tolerance rely on redundancy at multiple levels of transmission. For such applications an essential property of SCTP is the support for multi-homing that is the destination node can be reached under several IP addresses. TCP does not support multi-homing, anytime if an IP address become inaccessible due to an interface failure or any other reason, the TCP connection will timed out and will force upper layers to recover loss due to this failure, such type of delay can be unacceptable for mission critical applications such as IP Telephony. To cover this problem SCTP multi-homing was designed. SCTP supports multi-homing at transport layer level to allow sessions or associations to remain alive even when one of the IP address becomes unreachable. When a sender is connected through multiple interfaces with a receiver, it informs the receiver about all its available IP addresses when sending INIT chunk and receiver also provides all IP addresses in its INIT-ACK chunk. The SCTP multi-homing supports both IPv4 and IPv6 protocols and even it works with mixed environment that is IPv4 and IPv6 paths for the same receiver node. An SCTP entity assumes each IP address of its peer as one separate transmission path [4]. If no explicit IP addresses are sent in INIT or INIT-ACK chunks, the source IP address of the packet carrying SCTP chunk is used for association, this feature of SCTP is valuable when NAT is used at the edge of large private networks. And additional optional feature is also introduced in [2] allowing the use of hostnames instead of one or more IP addresses for multi-homing.

Path Selection

At the time of association start-up, a primary path is defined as "active" for each SCTP destination points and is used for

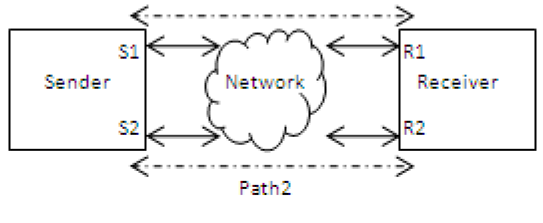


Figure 3.1: Example of Multi-homed Architecture.

normal sending of SCTP packets (chunks). The alternate or "secondary" path is only used for redundancy or load-balancing purpose, either to retransmit the packets lost, or when the primary destination address cannot be reached.

Path Monitoring

The all transmission paths of an association are monitored using special control chunks called HEARTBEATs, periodically the sender will send a HEARTBEAT chunk to the receiver, when the peer receiver receives the HEARTBEAT it copies the sender's specific heart beat information into its HEARTBEAT-ACK chunk and transmits it back to the sender immediately. When the sender receives the HEARTBEAT-ACK message it uses its contents to update the Round-Trip Time (RTT) for the destination address. Each path in an association is assigned either "active" or "inactive" state. A path is active if it has been used in the recent past transmission and has replayed for all acknowledgments to its peer. If transmission of a certain path fails repeatedly then this path is assumed as "inactive" [4].

Path Handover

In an SCTP association all data is transmitted on the primary path and other paths are monitored through HEARTBEAT messages. When a destination host becomes unreachable for the sender host a timer will expire and the host performing the monitoring will increment an error counter and wait a certain "back off" period before sending another HEARTBEAT to the receiver. When the error counter will exceed a user defined time limit the destination address will be marked as unreachable and will not be further considered for transmission of data. The monitoring will continue in case the problem is corrected and the unreachable destination address responding again. When the primary path becomes unreachable and the transmission is handed over to the redundant secondary path, all the unacknowledged data and the remaining data is transmitted on this secondary path [9].

4. EXPERIMENTAL DESIGN

This paper assumes an SCTP Concurrent Multipath Transfer (CMT-SCTP) [16] source and an CMT-SCTP receiver connected through dual-homed nodes. The network design contains the background traffic on both interface nodes according to the internet survey that is on internet TCP traffic is about 80 % - 83 % and UDP traffic is about 17% - 20 % [14].

4.1 Scope

As described previously, the focus of the experiments described in this paper is the impact of concurrent traffic, congestion control algorithms and different SACK timers on SCTP multi-homing. The scope of the experiments includes five different configurations. In the first configuration the SCTP multi-homing is tested without any background traffic. The background traffic could affect on the multi-homing so in the second configuration TCP and UDP background traffic is introduced with default SACK timer (200ms) and Drop Tail congestion control algorithm. In the third simulation experiment RED congestion control algorithm is used instead of Drop Tail. The behavior of congestion control algorithms is almost same for TCP and SCTP transport layer protocols but the scope of this experiment is to introduce the effect of

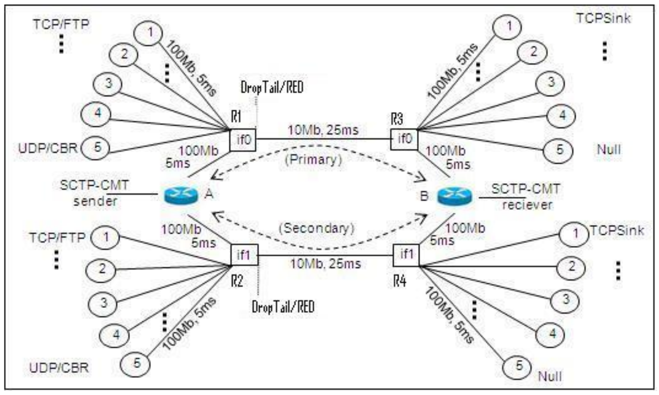


Figure 4.2.1 Network topology for CMT-SCTP

congestion control algorithms on multi-homing when data is being transferred on more than one interfaces. SCTP performs retransmission of the data on the basis of SACK received which shows the gap between the data packets. So the SACK timers could also have impact on the performance of multi-homing. According to RFC4960 SCTP default SACK timer to send the gap acknowledgment is 200ms. The fourth and fifth simulation uses CMT-SCTP with 100ms and 300ms, respectively.

The figure 4.2.1 shows network topology used for the simulation [11]. It is modeled using ns2.34 network simulator [15]. All results are obtained from implementation of SCTP multi-homing in the network simulator with a patch from university of Delaware and AWK scripts are used to extract required data from the results of simulations. The graphs are drawn in XGRAPH version 12.1.

4.2 Experimental Scenario

To evaluate the performance of CMT-SCTP in multi-homing a more realistic topology is considered as shown in figure 4.2.1. In this dual dumbbell topology, each router node R1 – R4 is connected to five edge nodes. The dual homed edge nodes A and B are the SCTP transport sender and receiver, respectively. The other edge nodes are single homed for the background traffic at the routers. The propagation delay between the edge nodes and routers is set to 5ms with 100mb of bandwidth. Each single homed edge node is attached with a traffic generator, introducing cross traffic with 80% (four nodes on each edge) of TCP traffic and 20% (one node on each edge) [14]. R1 and R2 are bottleneck for the whole traffic and their buffer size is set to twice the link bandwidth-delay product which is a reasonable setting in practice [11]. The propagation delay between dual homed interfaces is set to 25ms. The two paths between the end points are fully separated. The path between R1 to R4 is set as primary path, and CMT-SCTP uses concurrent multipath transfer on both paths. After 0.5 seconds of simulation CMT-SCTP sender source starts initiating association with receiver CMT-SCTP. On 1.0 other cross traffic that is TCP and UDP is started in the network. The traffic of each test is ended after 30 seconds which is more than enough to check the effects on the performance of SCTP multi-homing. Although the tests are also performed for 60 and 100 seconds of simulation, the results from 30 seconds are almost 99% same as from 60 and 100 seconds simulation.

Configuration Parameters

Five simulations were run with different network configurations. First simulation is started from one CMT-SCTP source without any background traffic (non-realistic environment). After testing the behavior of dual interfaces in a non-realistic environment, a realistic configuration is used in which TCP and UDP cross traffic is introduced. This configuration is used with Drop Tail congestion control algorithm on the bottleneck and default SACK timer (200ms). One simulation is performed with changing congestion control algorithm to RED in previous simulation configuration. Then finally two simulations are performed by changing the SACK timers to 100ms and 300ms.

Performance Metrics

We have considered following performance metrics to conclude the effect of concurrent traffic, congestion control algorithms and different SACK timers on SCTP multi-homing.

Average end-to-end Delay: This average end-to-end delay defines all the possible delays for successful transmitted data of SCTP through multi-homing. There are many factors causing delay in the network, such as queuing delay, buffering during congestion, latency and retransmission delay.

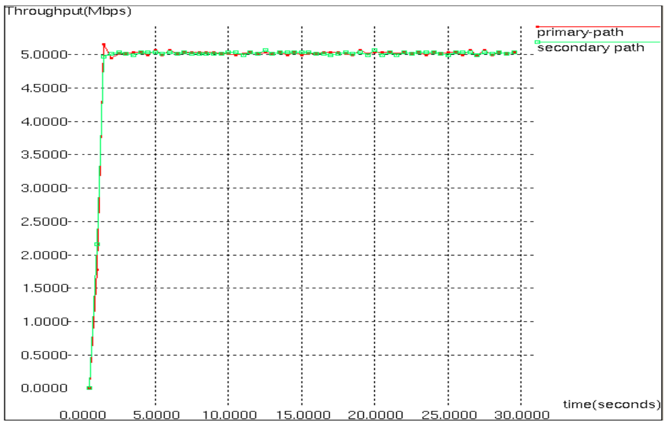


Figure 5.1.1 Throughput of SCTP dual homed interfaces without Background concurrent traffic

Throughput: Throughput is the total number of successful transmitted bits to the destination. We will measure throughput of both dual homed interfaces of SCTP nodes.
These metrics are checked and discussed with all the five simulations performed in this experimental setup.

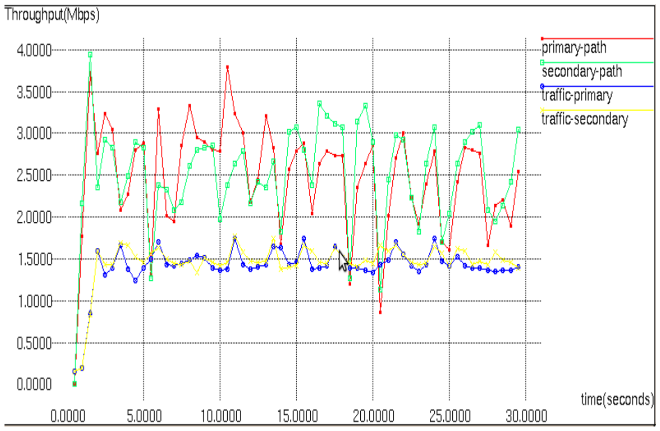


Figure 5.1.2 throughput of SCTP dual homed interfaces with Background traffic

5 Results and Analysis

We performed set of experiments in order to introduce the effect of assumed factors on multi-homing feature. The performance metrics described in Section-4.2 are tested for factors (Concurrent traffic, Congestion control Algorithms and SACK timers) during the simulations. The results obtained from simulations are analyzed to get conclusion for the research questions described in Section-1. The simulation results are divided into three sections according to factors.

5.1 Results and analysis for Concurrent Traffic

This section contains two simulation experiments which studied the effect of concurrent traffic on multi-homing feature of SCTP. Throughout this study we kept some parameters constant so that only the effect of cross traffic can be shown in the results. Figure 5.1.1 and figure 5.1.3 show the throughput and delay results of simulation when no background traffic is

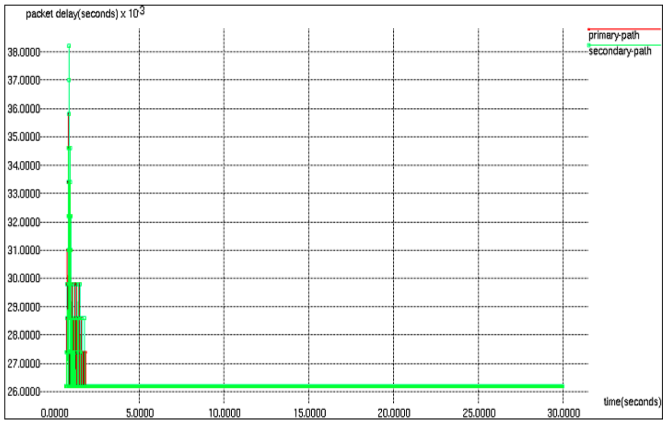


Figure 5.1.3 Delay of SCTP dual homed interfaces without Background traffic

present. The initial delay in figure 5.1.3 is caused when SCTP started its association from sender to the receiver and when association is established it starts sending data packets without any extra delay. Figure 5.1.2 shows throughput results of both dual homed interfaces when background traffic is introduced on the routers R1 and R2. In the graphs traffic-primary and traffic-secondary lines are showing background

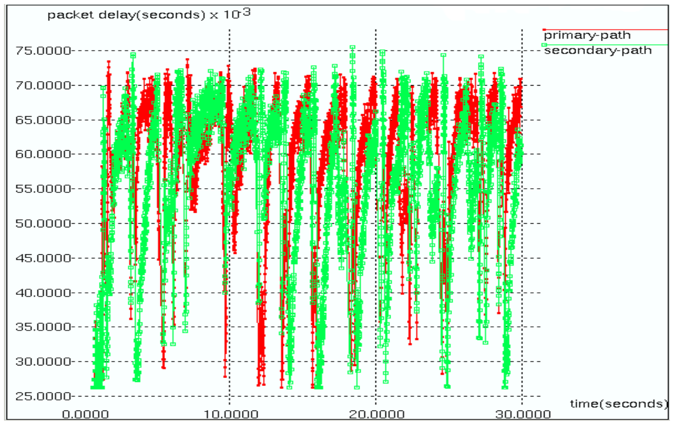


Figure 5.1.4 Delay of SCTP dual homed interfaces with background

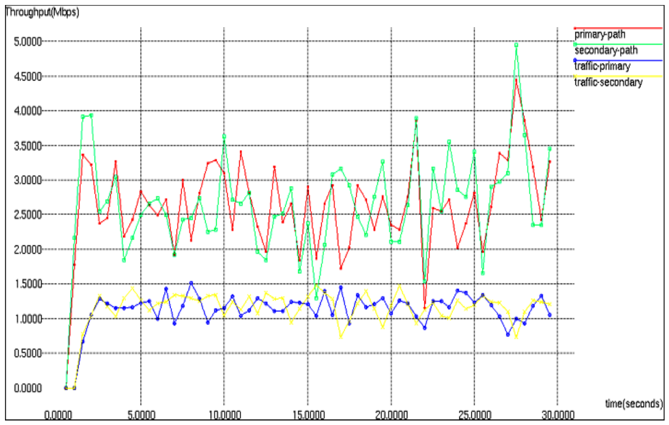


Figure 5.2.1 Throughput of SCTP dual homed interfaces with background traffic and RED congestion control algorithm.

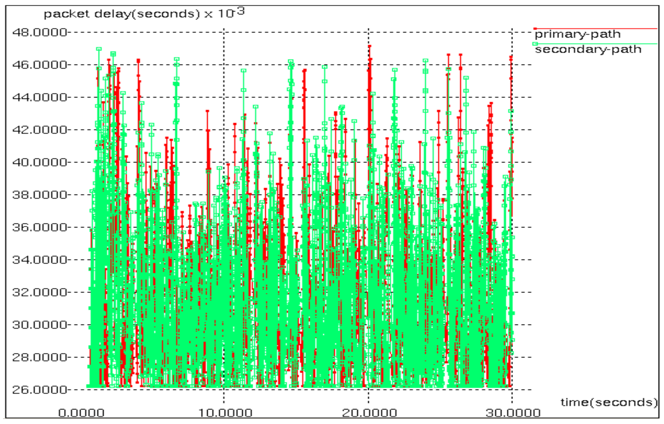


Figure 5.2.2 Delay of SCTP dual homed interfaces with background traffic and RED.

traffic on primary and secondary paths respectively. During cross traffic simulations we used Drop Tail congestion control algorithm on both interfaces with default 200ms SACK timer. Concurrent multipath transfer in SCTP sends concurrent traffic on both interfaces, the background traffic in the network affects SCTP multi-homing interfaces. Most of the bandwidth is used to deliver SCTP data packets as compared to background data traffic (TCP and UDP). The figure 5.1.4 shows delay on both. At the end of this chapter a table 5.1 shows that average throughput for both interfaces is 4.9847Mbps and average delay is 0.0451ms.

5.2 Results and analysis for Congestion control Algorithms

In the section 5.1 we performed a simulation with background traffic on SCTP multi-homing in which Drop Tail congestion control algorithm was used. As we also want to know the impact of congestion control algorithms on CMT-SCTP, therefore we performed another simulation in which RED (Random Early Detection) congestion control algorithm is used on the bottleneck routers R1 and R2. Figure 5.2.1 shows throughput result of simulation and figure 5.2.2 shows delay on dual homed interfaces during the simulation.

The comparison of figures 5.1.2 and 5.2.1 show that different congestion control algorithms have different affects on multi-

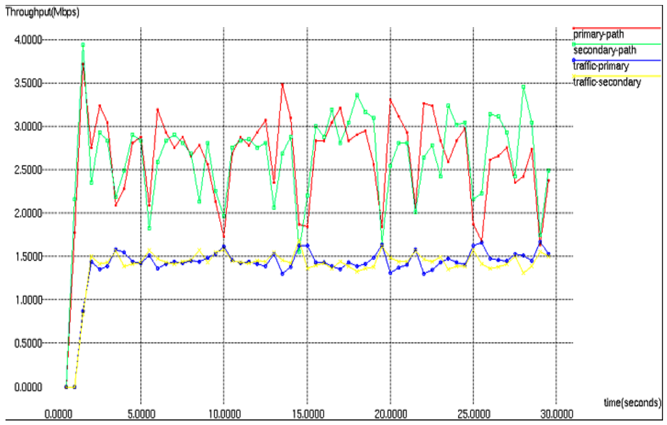


Figure 5.3.1 Throughput of SCTP dual homed interfaces with background traffic and 100ms SACK timer.

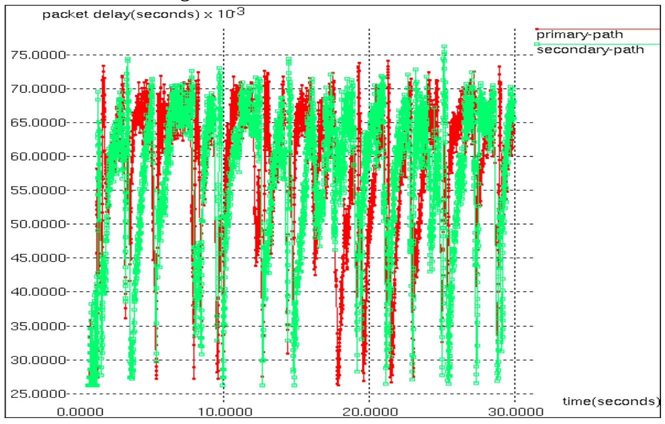


Figure 5.3.2 Delay of SCTP dual homed interfaces with background traffic and 100ms SACK timer.

homing feature. The figure 5.1.2 shows the effect of Drop Tail algorithm on both interfaces, a global synchronization can be found in the graph. On the other hand when RED congestion control algorithm is used the figure 5.2.1 shows the fair behavior of RED on both interfaces, which avoids global synchronization. The difference between the two paths for RED and Drop Tail is the same. The figures 5.1.4 and 5.2.2 are showing the delay of Drop Tail and RED algorithms on both interfaces. If we see the values in Table 5.1 it shows that RED gives about 10% more throughput and low delay as compared

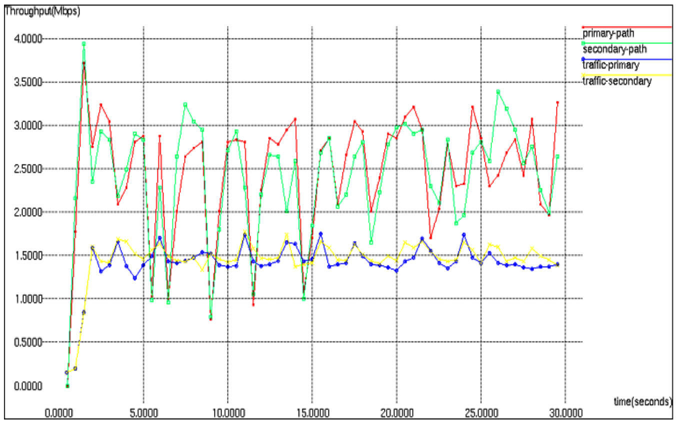


Figure 5.3.3 Throughput of SCTP dual homed interfaces with background traffic and 300ms SACK timer.

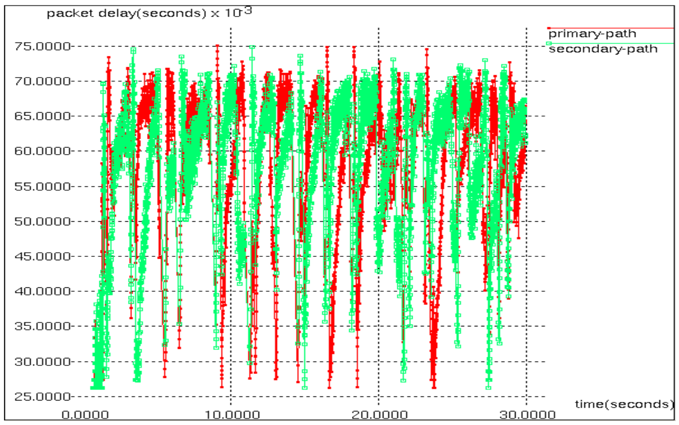
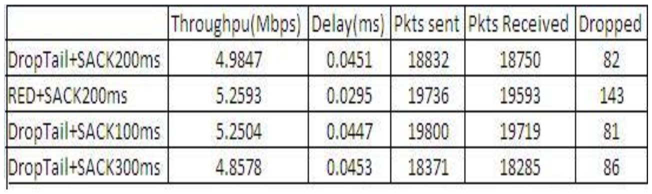


Figure 5.3.4 Delay of SCTP dual homed interfaces with background traffic and 300ms SACK timer.

to Drop Tail algorithm. Table 5.1 shows that in case of RED algorithm average throughput of both interfaces is 5.2593Mbs and average delay is 0.0295ms. The RED drops more packets because RED is more fair than Drop Tail, in the sense that it possess a bias against traffic that uses larger portion of the bandwidth. The more a sender transmits, the more packets are dropped and that is what RED is doing with SCTP sender.

5.3 Results and analysis for different SACK timers

SCTP receiver responds to sender using SACK packets about the gap of packets received. In CMT-SCTP it is suggested that the SACK information be treated as a concise description for the transmission sequence numbers (TSNs) from the receiver, hence from the SACK information a loss may not be immediately considered. The sender implies lost TSNs using information in SACKs and history information in the retransmission queue [16]. In our research question it is also



	Throughpu(Mbps)	Delay(ms)	Pkts sent	Pkts Received	Dropped
DropTail+SACK200ms	4.9847	0.0451	18832	18750	82
RED+SACK200ms	5.2593	0.0295	19736	19593	143
DropTail+SACK100ms	5.2504	0.0447	19800	19719	81
DropTail+SACK300ms	4.8578	0.0453	18371	18285	86

Table 5.1 Factors affecting results from simulations.

assumed that changing SACK timers could also affect on the performance of SCTP multi-homing. In this section we performed two more simulations with changing the SACK timer values in second simulation configuration with background traffic and Drop Tail congestion control algorithm. The figures 5.1.2 and 5.1.4 shows throughput and delay results of default SACK timer which is 200ms, we tested two more simulations with 100ms and 300ms SACK timers to check its affect on multi-homing.

Figures 5.3.1 and 5.3.3 show throughput results for 100ms and 300ms SACK timers respectively. Comparing these figures with figure 5.1.2 we come to know that in the case of 100ms SACK timer the throughput behavior of both interfaces is almost consistent and similar and the average throughput is also higher than 200ms and 300ms SACK timers. Figures 5.1.3, 5.3.2 and 5.3.4 show results of the delay for 100ms, 200ms and 300ms SACK timers, respectively which are almost similar in all three cases. The increase of throughput in case of 100ms SACK timer is because CMT-SCTP consider received SACK as report for TSNs not only the lost packets.

6 Conclusion and Future Work

In this paper we have presented an introduction to SCTP and briefly touched on some of its features. An overview of SCTP multi-homing is also outlined. The main work presented in this paper is to show the impact of different factors (like concurrent traffic in the network, congestion control algorithms and the different SACK timers) on multi-homing feature of SCTP. Concurrent Multipath Transfer using SCTP multi-homing (CMT-SCTP) is used in Network Simulator NS2 to introduce the effect of these factors. The results and analysis support our hypothesis and it can be concluded that when concurrent traffic is available in the network SCTP multi-homing provides efficient throughput as compared to other transport layer protocols (TCP and UDP). The SCTP multi-homing proves its usefulness in term of efficiency while sending data over multiple paths. RED and Drop Tail congestion control algorithms have same impact on SCTP as TCP, but in case of multi-homed destinations the RED algorithm provides low delay and about 10% of high throughput as compared to Drop Tail algorithm. In RFC4960 recommended SACK timer is 200ms, but when 100ms SACK timer is used with concurrent multipath transfer in SCTP (CMT-SCTP) multi-homing, the high throughput and low delay is achieved as compared with 200ms and 300ms, which indicates that different SACK timers affects on multi-homing feature of SCTP.

This paper is a first step in evaluating SCTP multi-homing performance and introducing the effect of different network factors like concurrent traffic, congestion control algorithms and different SACK timers. There is a significint weakness in our work which we have planned to address in near future. We have used only one CMT-SCTP source to estabalish SCTP multi-homined association, multiple CMT-SCTP associations can be tested for such network factors. The researchers working on SCTP have also introduced another concurrent multipath transfer using SCTP multi-homing called CMT with a potentially failed destination state (CMT-PF). In the future a work can be presented to evaluate CMT vs CMT-PF on the basis of factors like concurrent traffic, congestion control algorithms and SACK timers.

REFERENCES

[1] Eklund, J., Brown, A. and Strom (2005), *Performance of Network Redundancy Mechanisms in SCTP*. Research Report: Karlstads University

[2] Stewart, R. (September 2007), *Stream Control Transmission Protocol*, RFC4960 Std.

[3] Ravier, T., Brennan, R. and Curran, T. (2001), *Experimental Studies of SCTP multi-homing*, Teltec DCU: Ireland.

[4] Jungmaier, A., Erwin, P. Rathgeb, Schopp, M. and Tuxen, M. (2001), *SCTP- A Multi-Link End-to-end protocol for IP-based Network*. AEU- International Journal of Electronics and communications, Volume 55, Issue 1, pp. 46-54.

[5] Kang, S. and Fields, M. (2003) *Experimental Study of the SCTP compared to TCP*, Project Report, Engineering Department, Texas A&M University USA.

[6] Alamgir, R., Atiquzzaman, M. and Ivancic, W. (2002). *Effect of Congestion Control on the Performance of TCP and SCTP over Satellite Networks*: proceedings of the NASA Earth Science Technology Conference, Pasadena.

[7] Armando, L., Caro, Jr., Iyengar, J., Amer, P. and Heinz, G. (2002), *A Two Level Threshold Recovery Mechanism for SCTP:* proceedings of the SCI.

[8] Noonan, J., Perry, P., Murphy, S. and Murphy, J. (2006), *Stall and Path monitoring Issues in SCTP*: Proceedings of IEEE Infocom.

[9] Kelly, A., Perry, P. and Murphy, J. (July 2003), *A Modified SCTP Handover Scheme for Real Time Traffic*, HETNETs Working Conference, Ilkeley, England.

[10] Perotto, F., Casetti, C. and Galante, G. (March 2007), *SCTP-based Transport Protocols for Concurrent Multipath Transfer*, Wireless Communications & Networking Conference, WCNC 2007. *IEEE*, vol., no., pp.2969-2974

[11] Natarajan, P., Iyengar, J.R., Amer, P.D. and Stewart, R. (Oct. *2006*), *Concurrent Multipath Transfer using Transport Layer Multi-homing: Performance under Network Failures, Military Communications Conference, MILCOM 2006. IEEE*, vol., no., pp.1-7, 23-25

[12] Mena, J., Rusich, R., (Jan. *2006*), *SCTP: Stream Control Transmission Protocol an Analysis*, Master theses, UC Riverside University of California.

[13] Stewart, R., Tuxen, M., and Peter, L. (*2008*), SCTP: What is it, and how to use it? http://www.bsdcan.org/2008/schedule/attachments/

[14] Fomenkov, M., Keys, K., Moore, D. and Claffy, K. (Jan. 2004). *Longitudinal study of Internet traffic in 1998-2003*: proceedings of the Winter International Symposium on Information and Communication Technologies (WISICT), pp. 1-6, on January 5-8th, 2004 in Cancun, Mexico.

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

[16] Iyengar, J.R., Shah, K.C., Amer, P.D. and Stewart, R. (July 2004), *Concurrent Multipath Transfer Using SCTP Multihoming*, In *SPECTS*, California.

[17] Postel, J. (August 1980), *User Datagram Protocol*, RFC768 Std.

[18] Postel, J. (Sept. 1981), *Transmission Control Protocol*, RFC793 Std.

[19] Brennan, R. and Curran, T. (2001), *SCTP Congestion Control: Initial Simulation Studies*, Proc. 17th Int'l Teletraffic Congress, Elsevier Science.

[20] Siddiqui, F., Zeadally, S. (May 2006), *SCTP multihoming support for handoffs across heterogeneous networks*, Communication Networks and Services Research Conference, *CNSR 2006, Proceedings of the 4th Annual* , vol., no., pp.8 pp.-250.

[21] Fall, K. and Varadhan, K. (August 2000), *The Network Simulator ns-2: Documentation*, http://www.isi.edu/nsnam/ns/ns-documentation.html

[22] Chung, J. and Claypool, M., *NS by Example*, http://nile.wpi.edu/NS/

[23] Greis, M., *Tutorial for the Network Simulator "ns"*, http://www.isi.edu/nsnam/ns/tutorial/

[24] Issariyakul, T. and Hossain, E. (2009), *Introduction to Network Simulator NS2*. New York: Springer.

[25] Wei, G., Yuehui, J., Huihua, L., Hongtao, W. and Dongmei, Z. (2001), *SCTP simulation on NS*, Info-tech and Info-net, *Proceedings. ICII 2001 - Beijing 2001 International Conferences on*, vol.2, no., pp.345-350 vol.2.

[26] Chih-Heng, K. (September 2005), *How to measure packet loss rate, jitter, and end-to-end delay for UDP-based applications?*, *http://hpds.ee.ncku.edu.tw/~smallko/ns2/tool_en.htm*

APPENDIX: A

Following are the *TCL* and *awk* code files for SCTP in NS2 implementation, throughput and delay calculation.

A. #th-cmt-traffic.tcl

```
#Rashid Ali .. Master in computer Science.. University West
#Trollhattan Sweden
#Performance of Network redundancy in SCTP:
#Introducing effect of different factors on Multihoming
# Demonstrates Concurrent Multipath Transfer (CMT). Two
#endpoints with
# 2 interfaces with direct connections between each pair. Data
#is
# transfered using both paths concurrently.

Trace set show_sctphdr_ 1

set ns [new Simulator]
set nf [open th-cmt-traffic-sack300-30ms.nam w]
$ns namtrace-all $nf

set allchan [open th-cmt-traffic-sack300-30ms.tr w]
$ns trace-all $allchan

proc finish {} {
   global ns nf allchan
   $ns flush-trace
   close $nf
   close $allchan
  #exec xgraph out.tr -geometry 800x400 &
 # exec nam th-simple.nam &
   exit 0
}
#creation of SCTP sender node and interface nodes
set host0_core [$ns node]
set host0_if0 [$ns node]
set host0_if1 [$ns node]

$host0_core color Red
$host0_if0 color Red
$host0_if1 color Red
$host0_if0 shape box
$host0_if1 shape box
#attaching interface nodes as multihomed interfaces
$ns multihome-add-interface $host0_core $host0_if0
$ns multihome-add-interface $host0_core $host0_if1
#creation of SCTP reciever node
set host1_core [$ns node]
set host1_if0 [$ns node]
set host1_if1 [$ns node]

$host1_core color Blue
$host1_if0 color Blue
$host1_if1 color Blue
$host1_if0 shape box
$host1_if1 shape box
#attaching interface nodes as multihomed interfaces
$ns multihome-add-interface $host1_core $host1_if0
$ns multihome-add-interface $host1_core $host1_if1

$ns duplex-link $host0_if0 $host1_if0 10Mb 25ms DropTail
$ns duplex-link-op $host0_if0 $host1_if0 orient right
[[$ns link $host0_if0 $host1_if0] queue] set limit_ 50
$ns duplex-link $host0_if1 $host1_if1 10Mb 25ms DropTail
$ns duplex-link-op $host0_if1 $host1_if1 orient right
[[$ns link $host0_if1 $host1_if1] queue] set limit_ 50

set sctp0 [new Agent/SCTP/CMT]
$ns multihome-attach-agent $host0_core $sctp0
$sctp0 set sackDelay_ 0.300   #SACK timers checked
0.100,0.200.300, and 0.200 is a default
$sctp0 set fid_ 5
$sctp0 set debugMask_ -1
$sctp0 set debugFileIndex_ 0
$sctp0 set mtu_ 1500
$sctp0 set numOutStreams_ 1
$sctp0 set useCmtReordering_ 1   # turn on Reordering algo.
$sctp0 set useCmtCwnd_ 1         # turn on CUC algo.
$sctp0 set useCmtDelAck_ 1       # turn on DAC algo.
$sctp0 set eCmtRtxPolicy_ 4      # rtx. policy : RTX_CWND

#creation of SCTP agents for sender and reciever
set sctp1 [new Agent/SCTP/CMT]
$ns multihome-attach-agent $host1_core $sctp1
$sctp1 set debugMask_ -1
$sctp1 set debugFileIndex_ 1
$sctp1 set mtu_ 1500
$sctp1 set initialRwnd_ 65536
$sctp1 set useDelayedSacks_ 1
$sctp1 set useCmtDelAck_ 1

$ns color 0 Red
$ns color 1 Blue
$ns color 2 Green
$ns connect $sctp0 $sctp1
#setting up ftp application for SCTP traffic
set ftp0 [new Application/FTP]
$ftp0 attach-agent $sctp0

# set primary before association starts
$sctp0 set-primary-destination $host1_if0

#####primary0 path settings#############
#creations of nodes for tcp and UDP source
for { set i 0 } { $i < 5 } { incr i } {
set source0_($i) [$ns node]
}
#creation of sinks for tcp
for { set i 0 } { $i < 5 } { incr i } {
 set sinknode0_($i) [$ns node]
}
#linkage of nodes to the host0_if0
for { set i 0 } { $i < 5 } { incr i } {
$ns duplex-link $source0_($i) $host0_if0 100Mb 5ms DropTail
}
#linkage of sink nodes to the host1_if0
for { set i 0 } { $i < 5 } { incr i } {
$ns duplex-link $sinknode0_($i) $host1_if0 100Mb 5ms
DropTail
}
proc attach-tcp1-traffic { tcpnode sinknode size fid } {
   #Get an instance of the simulator
   set ns [Simulator instance]
#Create a TCP agent and attach it to the node
   set tcp [new Agent/TCP]
      $tcp set class_ 2
   $ns attach-agent $tcpnode $tcp
      $tcp set packetSize_ $size
      $tcp set fid_ $fid
   set sink [new Agent/TCPSink]
   $ns attach-agent $sinknode $sink
#Create ftp traffic agent and set its configuration parameters
   set traffic [new Application/FTP]
  #     # Attach traffic source to the traffic generator
      $traffic attach-agent $tcp
##Connect the source and the sink
   $ns connect $tcp $sink
   return $traffic
}
#
set null0 [new Agent/Null]
$ns attach-agent $sinknode0_(4) $null0
#Create a UDP agent and attach it to the node
   set udp0 [new Agent/UDP]
   $ns attach-agent $source0_(4) $udp0
#Create cbr traffic agent and set its configuration parameters
   set cbr0 [new Application/Traffic/CBR]
   $cbr0 set packetSize_ 1000
   $cbr0 set type_ CBR
   $cbr0 set random_ false
   $cbr0 set rate_ 1mb

# Attach traffic source to the traffic generator
     $cbr0 attach-agent $udp0
##Connect the source and the sink
   $ns connect $udp0 $null0

set tcp00 [attach-tcp1-traffic $source0_(0) $sinknode0_(0) 1040
1]
set tcp01 [attach-tcp1-traffic $source0_(1) $sinknode0_(1) 1040
1]
set tcp02 [attach-tcp1-traffic $source0_(2) $sinknode0_(2) 1040
1]
set tcp03 [attach-tcp1-traffic $source0_(3) $sinknode0_(3) 1040
1]

######primary pathe setting closed###########
######secondary path settings################
#creations of nodes for tcp and UDP source
for { set i 0 } { $i < 5 } { incr i } {
set source1_($i) [$ns node]
}
#cfreation of sinks for tcp
for { set i 0 } { $i < 5 } { incr i } {
 set sinknode1_($i) [$ns node]
}
#linkage of nodes to the host0_if0
for { set i 0 } { $i < 5 } { incr i } {
$ns duplex-link $source1_($i) $host0_if1 100Mb 5ms DropTail
}
#linkage of sink nodes to the host1_if0
for { set i 0 } { $i < 5 } { incr i } {
$ns duplex-link $sinknode1_($i) $host1_if1 100Mb 5ms
DropTail
}

proc attach-tcp2-traffic { tcpnode sinknode size fid } {
   #Get an instance of the simulator
   set ns [Simulator instance]
#  #Create a TCP agent and attach it to the node
   set tcp [new Agent/TCP]
      $tcp set class_ 2
   $ns attach-agent $tcpnode $tcp
      $tcp set packetSize_ $size
      $tcp set fid_ $fid
   set sink [new Agent/TCPSink]
   $ns attach-agent $sinknode $sink
#Create ftp traffic agent and set its configuration parameters
   set traffic [new Application/FTP]
  #     # Attach traffic source to the traffic generator
      $traffic attach-agent $tcp
   ##Connect the source and the sink
   $ns connect $tcp $sink
   return $traffic
}
#
set null1 [new Agent/Null]
$ns attach-agent $sinknode1_(4) $null1
#  #Create a UDP agent and attach it to the node
   set udp1 [new Agent/UDP]
   $ns attach-agent $source1_(4) $udp1
#  #Create cbr traffic agent and set its configuration parameters
   set cbr1 [new Application/Traffic/CBR]
   $cbr1 set packetSize_ 1000
   $cbr1 set type_ CBR
   $cbr1 set random_ false
   $cbr1 set rate_ 1mb

  #     # Attach traffic source to the traffic generator
     $cbr1 attach-agent $udp1
   ##Connect the source and the sink
   $ns connect $udp1 $null1

set tcp10 [attach-tcp2-traffic $source1_(0) $sinknode1_(0) 1040
2]
set tcp11 [attach-tcp2-traffic $source1_(1) $sinknode1_(1) 1040
2]
```

```
set tcp12 [attach-tcp2-traffic $source1_(2) $sinknode1_(2) 1040
2]
set tcp13 [attach-tcp2-traffic $source1_(3) $sinknode1_(3) 1040
2]
####secondary pathe setting
closed##################################

$ns at 0.1 "$cbr0 start"
$ns at 0.5 "$ftp0 start"
$ns at 1.0 "$tcp00 start"
$ns at 1.0 "$tcp01 start"
$ns at 1.0 "$tcp02 start"
$ns at 1.0 "$tcp03 start"

$ns at 0.1 "$cbr1 start"
$ns at 1.0 "$tcp10 start"
$ns at 1.0 "$tcp11 start"
$ns at 1.0 "$tcp12 start"
$ns at 1.0 "$tcp13 start"


$ns at 30.0 "finish"

$ns run
 ######################
```

B. #sctp-bw.awk

```
BEGIN {
# Initialization. fsDrops: packets drop. numFs: packets sent
       fsDrops = 0;
       packets = 0;
        throughput = 0.0;
average=0.0;
clock = 0.0;
sum = 0.0;
interval = 0.5;
}
{
   action = $1;     time = $2;
   from = $3;       to = $4;
   type = $5;       pktsize = $6;
   flow_id = $8;    src = $9;
   dst = $10;       seq_no = $11;
   packet_id = $12;
       if (time-clock<=interval)
        {
           #checking receing packets
           if ( action == "r" )
           {
              #checking destination 4 for primary and 5 for
        secondary
if (to==5)
              { #checking packet type SCTP
                 if (type=="sctp")
                 {
                      sum=sum+pktsize;
                      packets++;
                 }
              }
           }
        }
else
        {
#average=sum/packets;
        throughput=(sum/interval)*(8/1000000);
#average=throughput/packets;
        printf("%f\t%f\n", time, throughput);
        clock=clock+interval;
        sum=0;
        }

}
END {

}
##########################
```

C. #sctp-delay.awk

```
BEGIN {
highest_packet_id = 0;
drop=0;
}
{
 action = $1;
 time = $2;
 to =$4;
 from = $3;
 tracename = $4;
 type = $5;
pktsize=$6;
 packet_id =$12;
 if (packet_id > highest_packet_id)
        highest_packet_id = packet_id;
 if (action == "+")
        {
        if (type == "sctp" && pktsize = 1500 && to == 5)
           {
           send_time[packet_id] = time;
           }
        else
           {
           send_time[packet_id] = 0;
           }
        }
else
        if (action == "r")
           {
#to=4 for primary and to=5 for secondary path
           if (type == "sctp" && pktsize = 1500 && to == 5)
```

```
            rcv_time[packet_id] = time;
            }
        else
            {
            rcv_time[packet_id] = 0;
            }



if (action == "d" && type == "sctp")
drop++;
}
 END {
        packet_no = 0; total_delay = 0;
        for   (packet_id   =   0;   packet_id   <=highest_packet_id;
        packet_id++)
            {
          #if                  ((send_time[packet_id]!=0)                  &&
        (rcv_time[packet_id]!=0)){
              start = send_time[packet_id];
              end = rcv_time[packet_id];
               packet_duration = end-start;
if ( start < end ) printf("%f\t%f\n", start, packet_duration);      }
#printf("total packets droped= %d\n", drop);
}
######################
```