# Performance Analysis of SCTP & TCP SACK in Heterogeneous Networks



*An MS Final Year Project by*

**Muhammad Rafiq**
**07/FAS/MSTE/F04**
**Mansoor Waheed**
**09/FAS/MSTE/F04**

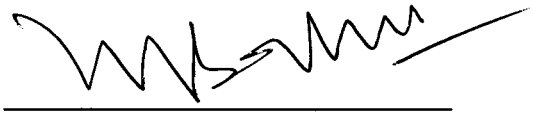
*Supervised by*

**Engr. Raza Hasan Abedi**
**Dr. Syed Afaq Hussain**
**Department of Electronic Engineering**
**Faculty of Engineering & Technology**

**International Islamic University, Islamabad.**
**August 2007**

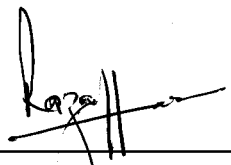
# Certificate of Approval

It is certified that we have read the project report submitted by **Muhammad Rafiq [07-FAS/MSTE/F04] and Mansoor Waheed 09-FAS/MSTE/F04.** It is our judgment that this project is of sufficient standard to warrant its acceptance by the International Islamic University, Islamabad for MS(TE) Degree in Telecommunication Engineering.



External Examiner
**Dr. Muhammad Aamir Saleem**
Department of Electrical Engineering
Faculty of Engineering
Air University
Islamabad

Internal Examiner
**Dr. Aqdas Naveed**
Assistant Professor
Department of Electronic Engineering
Faculty of Engineering & Technology
International Islamic University
Islamabad



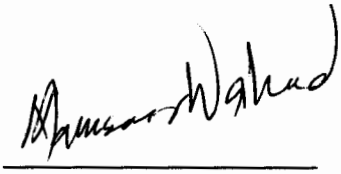
Supervisor
**Raza H. Abedi**
Assistant Professor
Department of Electronic Engineering
Faculty of Engineering & Technology
International Islamic University
Islamabad

# DECLARATION

I/We, hereby declare that this software, neither as a whole nor as a part thereof has been copied out from any source. It is further declare that we have developed this software and the accompanied report entirely on the basis of our personal effort made under the guidance of our teachers.

If any part of this report to be copied out or found to be reported, We shall standby the consequences, no portion of this work presented in this report has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

**Muhammad Rafiq**
**07-FAS/MSTE/F04**



**Mansoor Waheed**
**09-FAS/MSTE/F04**

*In the name of Allah (SWT) the most benificient, the most merciful.*

*An MS Final Year Project submitted to the*
*Department of Electronic Engineering*
*International Islamic University, Islamabad*
*In partial fulfillment of the requirements*
*For the award of the degree of*
*MS in Telecommunication Engineering*

*Dedicated To,*

*Our parents,*

*Who always supported us and prayed for our success*

# Acknowledgement

First of all we would like to take this opportunity to our humble gratitude to Almighty Allah who enabled us to accomplish this task. Who helped us in every crucial stage and bestowed us the stamina to achieve our aim.

We are also fortunate to have had Professor Dr. Syed Afaq Hussain and Sir Raza Hasan Abedi as our MS. advisor. Their ability to ask the right questions and attention to detail has never ceased to surprise us. They were always able to make time for us, even if we just walked into office without any notice. We would also like to mention their patience, giving us inspiration and hope when we were stuck at dead-ends.

Special thanks go to our parents and friends for their support throughout our entire academic career.

**Muhammad Rafiq**
**&**
**Mansoor Waheed**

# Project in Brief

| | |
|---|---|
| **Project Title:** | **Performance Analysis of SCTP vs TCP SACK in Heterogeneous Network** |
| **Undertaken By:** | Muhammad Rafiq<br>07-FAS/MSTE/F04<br>Mansoor Waheed<br>09-FAS/MSTE/F04 |
| **Supervised By:** | Engr. Raza Hasan Abedi<br>Dr. Syed Afaq Hussain |
| **Date Started:** | September 2006 |
| **Date Completed:** | August 2007 |
| **Tools Used:** | NS-2, C++, Linux, MS Excel |

# ABSTRACT

The quality of Internet Applications perceived by end users depends on many factors; more important are delay, Packet loss and throughput. This quality of service parameters are influenced by the network condition, the use of different transport protocol and the application. In today's world the wireless network is growing rapidly and heterogeneity of networks continues to increase so this heterogeneity of the network motivates us to work in heterogeneous environment. The provision of QoS over such heterogeneous network is complex and challenging. SCTP is a new IP transport protocol, working an equivalent level with TCP which provide transport layer functions to Internet traffic. SCTP has been approved by the IETF as a Proposed Standard. SCTP has some extra feature such as multihoaming, multistreaming and prevention against denial of service attack. TCP SACK is an extension to TCP that uses selective ACKs in addition to the cumulative ACKs. A simulation based comparison was performed using FTP traffic, between the two transport protocol SCTP, and TCP SACK to analyze the delay and throughput in heterogeneous environment in two different scenarios. In first case wired node act as a sender and wireless node act as receiver while in second scenarios wireless act as sender and wired as receiver. Simulation results shows that SCTP has less mean delay than TCP SACK but similar throughput with different loss probability. This leads to the conclusion that the introduction of this new protocol SCTP into a TCP/IP network does not degrade the performance of the existing protocols.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

*Chapter 1*

# 1 INTRODUCTION AND MOTIVATION

The quality of Internet Applications perceived by end users depends on many factors. Among the more important ones there are delay, Packet loss, and throughput. Understanding how these parameters are influenced by network conditions, by used protocols, and by traffic profiles may prove to be useful in several cases: (i) in the definition and implementation of Service Level Agreements (SLAs); (ii) in the design of new network applications; (iii) in building new network infrastructures; (iv) in the design of new QoS aware routing algorithms; etc.

Delay is the moment when the application passed the packet to the transport protocol and on the other side when it is received by the application. As there are two type of delay Round-trip-delay and the one-way delay. We are only interesting to measure the one way delay because the maximum possible one way delay is of great importance to some application.

In today's world the wireless network is growing rapidly and heterogeneity of networks continues to increase so most of the transport protocols is designed to tackle the problems related to the wired networks and they simply ignore or assume it will also perform well in wireless networks but it is not true and there are a lot of problems when tested different types of application with different transport protocols. These different transport protocols and the heterogeneity of existing networks motivated us to do research in heterogeneous networks with different transport protocols.

We performed a simulation based comparison using FTP traffic of two transport protocol SCTP, and TCP SACK to analyze the delay and throughput in heterogeneous environment. We also evaluated the behavior of transport protocol in competing traffic and the impact of loss probability on delay and throughput.

This report is organized as follow. Chapter 2 gives an overview of the related work has accomplished in this field. Chapter 3 gives some over view of the SCTP. Chapter 4 gives an overview of TCP, TCP Sacks and the difference between the SCTP and TCP Sacks. Chapter 5 present simulation results carried out in Network Simulator (NS - 2.30) [1]. And finally chapter 6 concludes our finding and present future work.

*Chapter 2*

# 2 LITRATURE REVIEW

In case of [2] authors proposed a new protocol which could be used for the transport of telecommunication signaling messages over an IP based network is currently being discussed within the IETF. The protocol is called Stream Control Transmission Protocol (SCTP). Authors evaluate how the protocol performs in a wide area network, especially when competing with TCP. The results were obtained in a test-bed consisting of two local networks which are interconnected via an emulator of a wide area network.

Author also shows that SCTP traffic has the same impact on TCP traffic as normal TCP traffic. This leads to the conclusion that the introduction of this new protocol into a TCP/IP network does not degrade the performance of the existing protocols.

In [3] author investigates the suitability of SCTP for data communications over satellite links. They describe SCTP features that allow SCTP to better utilize the bandwidth of satellite networks, while at the same time avoiding congestion collapse in a shared network. They recommend SCTP for the satellite network.

In [4] author presents an SCTP variant, called New-Reno SCTP, which introduces three modifications. First, a Fast Recovery mechanism, similar to that of New-Reno TCP, is included to avoid multiple congestion window (cwnd) reductions in a single round-trip time. Secondly they introduce a new policy which restricts the cwnd from being increased during Fast Recovery, thus ensuring that the newly introduced Fast Recovery mechanism maintains conservative behavior. Thirdly they modify SCTP's HTNA (Highest TSN Newly Acked) algorithm to ensure that Fast Retransmits are not unnecessarily delayed. Their results show that New-Reno SCTP performs better, and still conforms to AIMD principles. Also, they compare these two variants of SCTP with New-Reno TCP and SACK TCP under different loss scenarios. Results show that

New-Reno SCTP performs significantly better than New-Reno TCP, maintains conservative behavior similar to SACK TCP, and is as robust as SACK TCP to multiple losses in a window.

In [5] writers propose the Stream Control Transmission Protocol (SCTP), a recent IETF transport layer protocol, for reliable web transport. Although TCP has traditionally been used, they argue that SCTP better matches the needs of HTTP-based network applications. Paper also discusses SCTP features: (i) head-of-line blocking within a single TCP connection, (ii) vulnerability to network failures, and (iii) vulnerability to denial-of-service SYN attacks. They also discuss the benefits of using SCTP in other web domains through two example scenarios - multiplexing user requests, and multiplexing resource access. Finally, they highlight several SCTP features that will be valuable to the design and implementation of current HTTP-based client-server applications.

In [6] writer analyze that the Stream Control Transmission Protocol (SCTP) is a reliable message-based transport protocol developed by the IETF that could replace TCP in some applications. SCTP allows endpoints to have multiple IP addresses for the purposes of fault tolerance. There is on-going work to extend the SCTP multihoming functions to support dynamic addressing and endpoint mobility. Their paper also explains how the multihoming and mobility features can be exploited for denial-of-service attacks, connection hijacking, and packet flooding.

In case of the wired scenario, authors in [7] presented results related to the experimentation of SCTP over high speed WAN. Their result show that SCTP congestion control performs badly in high speed wide area network because of its slow response with large congestion window. So they proposed a new congestion window modification for the SCTP to improve its performance in high speed wide area network. The result of several experiment proved that their suggested congestion window improved the throughput of the original SCTP congestion control scheme significantly.

In [8] the authors compare the performance of SCTP and TCP with respect to Web traffic. The results show that SCTP can help reduce the latency and improve throughput. This, together with some other features of SCTP like multihoming and better protection against Denial of Service attacks, makes it a very attractive choice for future web traffic. In this paper author focus is on the web traffic and the author feel that the results bring out the deleterious effect of head of line blocking that applications using TCP suffer from, because TCP couples the delivery mechanism and reliability. By separating these two important issues and also by providing a reliable message-oriented transport, SCTP provides developers with both flexibility and efficiency. It will be interesting to know the mix of applications that use TCP as a byte stream and those that use TCP as a reliable transport for messages that are delineated by the application level protocol. There are some significant weaknesses in their work which they plan to address in the near future. As they compare SCTP against TCP Reno (default mode of TCP on BSD). Since, SCTP uses Selective Acknowledgements (SACK), we believe this is not a very fair comparison.

In [9], using ns-2, the authors study the multi-streaming and the multi-homing SCTP features. They prove that these features have advantages over TCP in the given scenarios. In particular, they define the optimal number of streams in multi-streaming and explain how it affects network performance. In the case of wireless networks, in [10] the authors developed an analytical model that takes into account the congestion window, the round trip time, the slow start and congestion avoidance processes to predict the SCTP performance. By comparing numerical results from the analytical model with simulation results, they demonstrate that the proposed model is able to accurately predict SCTP throughput.

In [11] the authors presented a simulation study of delay spike of SCTP, TCP Reno, and Eifel over wireless links. They found that Eifel performs better than TCP Reno and SCTP when there are no packet losses. However, the opposite is happen when packets are lost in the presence of delay spikes. Also they showed that a higher link bandwidth does not always increase the data throughput of SCTP, TCP Reno, and

Eifel. In [12] the authors provide a simulation-based performance comparison of SCTP vs. TCP in MANET environments. They found that SCTP and TCP have similar behavior in MANETs environment, but TCP outperforms SCTP in most cases because of the extra overhead present in SCTP.

And finally in [13] author present a packet level experimental analysis of SCTP in wired, wireless and heterogeneous (wired/wireless) scenarios in term of Throughput and jitter and compare SCTP with TCP and UDP. They introduced a traffic generation tool that support UDP, TCP and SCTP traffic generation at packet level. Result presented may define a reference framework for the development of the throughput and jitter aware SCTP based network application.

# 3 INTRODUCTION TO SCTP

SCTP is a new IP transport protocol, existing at an equivalent level with TCP (Transmission Control Protocol), which provide transport layer functions to many Internet applications. SCTP has been approved by the IETF as a Proposed Standard [14].

Like TCP, SCTP is a session-oriented mechanism, meaning that a relationship is created between the endpoints of an SCTP association prior to data being transmitted, and this relationship is maintained until all data transmission has been successfully completed.

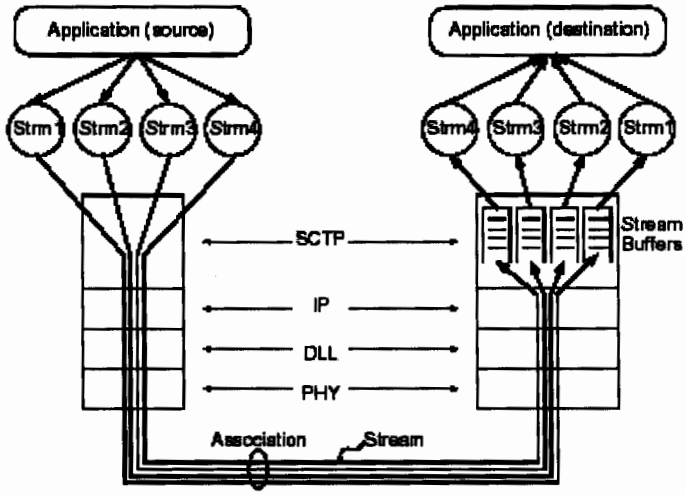
Unlike TCP, SCTP provides a number of functions that are critical for telephony signaling transport and at the same time can potentially benefit other applications needing transport with additional performance and reliability.

## 3.1 Basic SCTP Features

SCTP is a unicast protocol, and supports data exchange between exactly 2 endpoints, although these may be represented by multiple IP addresses. SCTP provides reliable transmission, detecting when data is discarded, reordered, duplicated or corrupted, and retransmitting damaged data as necessary. SCTP transmission is full duplex. SCTP is message oriented and supports framing of individual message boundaries. In comparison, TCP is byte oriented and does not reserve any implicit structure within a transmitted byte stream without enhancement. SCTP is rate adaptive similar to TCP, and will scale back data transfer to the prevailing load conditions in the network. It is designed to behave cooperatively with TCP sessions attempting to use the same bandwidth.

## 3.2 Multi-Streaming

The name Stream Control Transmission Protocol is derived from the multi-streaming function provided by SCTP. This feature allows data to be partitioned into multiple streams that have the property of independently sequenced delivery, so that message loss in any one stream will only initially affect delivery within that stream, and not delivery in other streams as shown in the Fig 3.1.

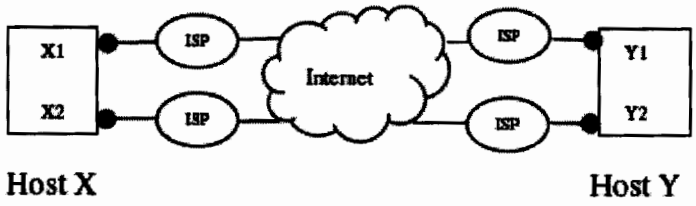



3.1 SCTP Multistreaming

In contrast, TCP assumes a single stream of data and ensures that delivery of that stream takes place with byte sequence preservation. While this is desirable for delivery of a file or record, it causes additional delay when message loss or sequence error occurs within the network. When this happens, TCP must delay delivery of data until the correct sequencing is restored, either by receipt of an out-of-sequence message, or by retransmission of a lost message. For a number of applications, the characteristic of strict sequence preservation is not truly necessary. In telephony signaling, it is only necessary to maintain sequencing of messages that affect the same resource (e.g., the same call, or the same channel). Other messages are only loosely correlated and can be delivered without having to maintain overall sequence integrity.

## 3.3 Multi-Homing

Another core feature of SCTP is multi-homing, or the ability for a single SCTP endpoint to support multiple IP addresses. As shown in the Figure 2 where Host X has two paths to communicate with the Host Y. If one path goes down the alternative path can be use. This also increases the reliability of the system. The benefit of multi-homing is potentially greater survivability of the session in the presence of network failures. In a conventional single-homed session, the failure of a local LAN access can isolate the end system, while failures within the core network can cause temporary unavailability of transport until the IP routing protocols can reconverted around the point of failure. Using multi-homed SCTP, redundant LANs can be used to reinforce the local access, while various options are possible in the core network to reduce the dependency of failures for different addresses. Use of addresses with different prefixes can force routing to go through different carriers, for example, route-pinning techniques or even redundant core networks can also be used if there is control over the network architecture and protocols. The following figure 3.2 shows that there are two paths available from the host X to host Y. In case of one link failure the redundant path can be used.




3.2 SCTP Multihoming

In its current form, SCTP does not do load sharing, that is, multi-homing is used for redundancy purposes only. A single address is chosen as the "primary" address and is used as the destination for all DATA chunks for normal transmission. Retransmitted

DATA chunks use the alternate address (es) to improve the probability of reaching the remote endpoint, while continued failure to send to the primary address ultimately results in the decision to transmit all DATA chunks to the alternate until heartbeats can reestablish the reach ability of the primary. To support multi-homing, SCTP endpoints exchange lists of addresses during initiation of the association. Each endpoint must be able to receive messages from any of the addresses associated with the remote endpoint; in practice, certain operating systems may utilize available source addresses in round robin fashion, in which case receipt of messages from different source addresses will be the normal case. A single port number is used across the entire address list at an endpoint for a specific session. In order to reduce the potential for security issues, it is required that some response messages be sent specifically to the source address in the message that caused the response. For example, when the server receives an INIT chunk from a client to initiate an SCTP association, the server always sends the response INIT ACK chunk to the source address that was in the IP header of the INIT.

## 3.4 SCTP Initiation Procedure

The SCTP Initiation Procedure relies on a 4-message sequence, where DATA can be included on the 3rd and 4th messages of the sequence, as these messages are sent when the association has already been validated. A "cookie" mechanism has been incorporated into the sequence to guard against some types of denial of service attacks.

### *3.4.1 Cookie Mechanism*

The "cookie" mechanism guards specifically against a blind attacker generating INIT chunks to try to overload the resources of an SCTP server by causing it to use up memory and resources handling new INIT requests. Rather than allocating memory for a Transmission Control Block (TCB), the server instead creates a Cookie parameter with the TCB information, together with a valid lifetime and a signature for

authentication, and sends this back in the INIT ACK. Since the INIT CK always goes back to the source address of the INIT, the blind attacker will not get the Cookie. A valid SCTP client will get the Cookie and return it in the COOKIE ECHO chunk, where the SCTP server can validate the Cookie and use it to rebuild the TCB. Since the server creates the Cookie, only it needs to know the format and secret key, this is not exchanged with the client. Otherwise, the SCTP Initiation Procedure follows many TCP conventions, so that the endpoints exchange receiver windows, initial sequence numbers, etc. In addition to this, the endpoints may exchange address lists as discussed above, and also mutually confirm the number of streams to be opened on each side.

### 3.5 INIT Collision Resolution

Multi-homing adds to the potential that messages will be received out of sequence or with different address pairs. This is a particular concern during initiation of the association, where without procedures for resolving the collision of messages, you may easily end up with multiple parallel associations between the same endpoints. To avoid this, SCTP incorporates a number of procedures to resolve parallel initiation attempts into a single association.

### 3.6 DATA Exchange

DATA chunk exchange in SCTP follows TCP's Selective ACK procedure. Receipt of DATA chunks is acknowledged by sending SACK chunks, which indicate not only the cumulative Transmission Sequence Number (TSN) range received, but also any non-cumulative TSNs, implying gaps in the received TSN sequence. Following TCP procedures, SACKs are sent using the "delayed ack" method, normally one SACK per every other received packet, but with an upper limit on the delay between SACKs and an increase to once per received packet when there are gaps detected.

Flow and Congestion Control follow TCP algorithms. The advertised receive window indicates buffer occupancy at the receiver, while a per-path congestion window is

maintained to manage the packets in flight. Slow start, Congestion avoidance, Fast recovery and Fast retransmit are incorporated into the procedures as described in RFC 2581, with the one change being that the endpoints must manage the conversion between bytes sent and received and TSNs sent and received, since TSN is per chunk rather than per byte.

The application can specify a lifetime for data to be transmitted, so that if the lifetime has expired and the data has not yet been transmitted, it can be discarded (e.g., time-sensitive signaling messages). If the data has been transmitted, it must continue to be delivered to avoid creating a hole in the TSN sequence.

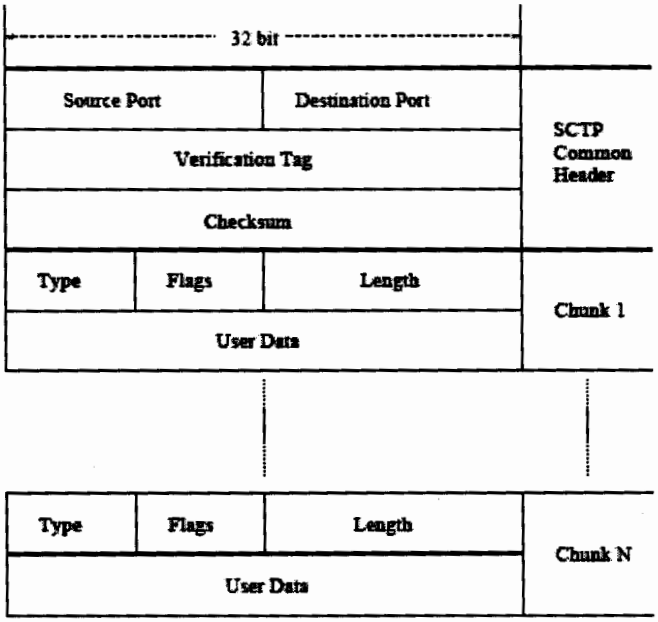
### 3.7 SCTP Shutdown

SCTP Shutdown uses a 3-message procedure to allow graceful shutdown, where each endpoint has confirmation of the DATA chunks received by the remote endpoint prior to completion of the shutdown. An Abort procedure is also provided for error cases when an immediate shutdown must take place. Note that SCTP does not support the function of a "half-open" connection as can occur in TCP, when one side indicates that it has no more data to send, but the other side can continue to send data indefinitely. SCTP assumes that once the shutdown procedure begins, both sides will stop sending new data across the association, and only need to clear up acknowledgements of previously sent data.

### 3.8 SCTP Message Format

The SCTP Message includes a common header plus one or more chunks, which can be control or data. The common header has source and destination port numbers to allow multiplexing of different SCTP associations at the same address, a 32-bit verification tag that guards against insertion of an out-of-date or false message into the SCTP association, and a 32-bit checksum for error detection.

Each chunk includes chunk type, flag field, length and value. Control chunks incorporate different flags and parameters depending on the chunk type. DATA chunks in particular incorporate flags for control of segmentation and reassembly, and parameters for the TSN, Stream ID and Stream Sequence Number, and a Payload Protocol Identifier.




3.3 SCTP Message Format

The Payload Protocol ID has been included for future flexibility. It is envisioned that the functions of protocol identification and port number multiplexing will not be as closely linked in the future as they are in current usage. Payload Protocol ID will allow the protocol being carried by SCTP to be identified independent of the Port numbers being used.

The SCTP message format naturally allows support of bundling of multiple DATA and control chunks in a single message, to improve transport efficiency. Use of bundling is controllable by the application, so that bundling of initial transmission can

be prohibited. Bundling will naturally occur on retransmission of DATA chunks, to further reduce any chance of congestion.

## 3.9 Congestion control

Congestion control is one of the basic functions in SCTP. For some applications, it may be likely that adequate resources will be allocated to SCTP traffic to assure prompt delivery of time-critical data thus it would appear to be unlikely, during normal operations, that transmissions encounter severe congestion conditions. However SCTP must operate under adverse operational conditions, which can develop upon partial network failures or unexpected traffic surges.

In such situations SCTP must follow correct congestion control steps to recover from congestion quickly in order to get data delivered as soon as possible. In the absence of network congestion, these preventive congestion control algorithms should show no impact on the protocol performance.

### *3.9.1 SCTP Differences from TCP Congestion control*

Gap Ack Blocks in the SCTP SACK carry the same semantic meaning as the TCP SACK. TCP considers the information carried in the SACK as advisory information only. SCTP considers the information carried in the Gap Ack Blocks in the SACK chunk as advisory. In SCTP, any DATA chunk that has been acknowledged by SACK, including DATA that arrived at the receiving end out of order, are not considered fully delivered until the Cumulative TSN Ack Point passes the TSN of the DATA chunk (i.e., the DATA chunk has been acknowledged by the Cumulative TSN Ack field in the SACK). Consequently, the value of cwnd controls the amount of outstanding data, rather than (as in the case of non-SACK TCP) the upper bound between the highest acknowledged sequence number and the latest DATA chunk that can be sent within the congestion window. SCTP SACK leads to different implementations of fast-retransmit and fast-recovery than non-SACK TCP.

The biggest difference between SCTP and TCP, however, is multi-homing. SCTP is designed to establish robust communication associations between two endpoints each of which may be reachable by more than one transport address. Potentially different addresses may lead to different data paths between the two endpoints, thus ideally one may need a separate set of congestion control parameters for each of the paths. The treatment here of congestion control for multi-homed receivers is new with SCTP and may require refinement in the future. The current algorithms make the following assumptions:

- The sender usually uses the same destination address until being instructed by the upper layer otherwise; however, SCTP may change to an alternate destination in the event an address is marked inactive .Also, SCTP may retransmit to a different transport address than the original transmission.
- The sender keeps a separate congestion control parameter set for each of the destination addresses it can send to (not each source-destination pair but for each destination). The parameters should decay if the address is not used for a long enough time periods.
- For each of the destination addresses, an endpoint does slow-start upon the first transmission to that address.

### *3.9.2 SCTP Slow-Start and Congestion Avoidance*

- The slow start and congestion avoidance algorithms must be used by an endpoint to control the amount of data being injected into the network. The congestion control in SCTP is employed in regard to the association, not to an individual stream. In some situations it may be beneficial for an SCTP sender to be more conservative than the algorithms allow, however, an SCTP sender must not be more aggressive than the following algorithms allow. Like TCP, an SCTP endpoint uses the following three control variables to regulate its transmission rate.
- Receiver advertised window size (rwnd, in bytes), which is set by the receiver based on its available buffer space for incoming packets.

- Congestion control window (cwnd, in bytes), which is adjusted by the sender based on observed network conditions.
- Slow-start threshold (ssthresh, in bytes), which is used by the sender to distinguish slow start and congestion avoidance phases.

SCTP also requires one additional control variable, partial_bytes_acked, which is used during congestion avoidance phase to facilitate cwnd adjustment.
Unlike TCP, an SCTP sender must keep a set of these control variables cwnd, ssthresh and partial_bytes_acked for EACH destination address of its peer (when its peer is multi-homed). Only one rwnd is kept for the whole association (no matter if the peer is multi-homed or has a single address).

### *3.9.3 Slow-Start*

Beginning data transmission into a network with unknown conditions or after a sufficiently long idle period requires SCTP to probe the network to determine the available capacity. The slow start algorithm is used for this purpose at the beginning of a transfer, or after repairing loss detected by the retransmission timer.

- The initial cwnd before DATA transmission or after a sufficiently long idle period MUST be <= 2*MTU.
- The initial cwnd after a retransmission timeout MUST be no more than 1*MTU.
- The initial value of ssthresh MAY be arbitrarily high (for example, implementations MAY use the size of the receiver advertised window).
- Whenever cwnd is greater than zero, the endpoint is allowed to have cwnd bytes of data outstanding on that transport address.
- When cwnd is less than or equal to ssthresh an SCTP endpoint MUST use the slow start algorithm to increase cwnd (assuming the current congestion window is being fully utilized). If an incoming SACK advances the Cumulative TSN Ack Point, cwnd MUST be increased by at most the lesser of

(1) the total size of the previously outstanding DATA chunk(s) acknowledged, and (2) the destination's path MTU.

In instances where its peer endpoint is multi-homed, if an endpoint receives a SACK that advances it's Cumulative TSN Ack Point, then it should update its cwnd (or cwnds) apportioned to the destination addresses to which it transmitted the acknowledged data. However if the received SACK does not advance the Cumulative TSN Ack Point, the endpoint MUST NOT adjust the cwnd of any of the destination addresses.

Because an endpoint's cwnd is not tied to its Cumulative TSN Ack Point, as duplicate SACKs come in, even though they may not advance the Cumulative TSN Ack Point an endpoint can still use them to clock out new data. That is, the data newly acknowledged by the SACK diminishes the amount of data now in flight to less than cwnd; and so the current, unchanged value of cwnd now allows new data to be sent. On the other hand, the increase of cwnd must be tied to the Cumulative TSN Ack Point advancement as specified above. Otherwise the duplicate SACKs will not only clock out new data, but also will adversely clock out more new data than what has just left the network, during a time of possible congestion. When the endpoint does not transmit data on a given transport address, the cwnd of the transport address should be adjusted to max (cwnd/2, 2*MTU) per RTO.

### *3.9.4 Congestion Avoidance*

When cwnd is greater than ssthresh, cwnd should be incremented by 1*MTU per RTT if the sender has cwnd or more bytes of data outstanding for the corresponding transport address.

In practice an implementation can achieve this goal in the following way:

- Partial_bytes_acked is initialized to 0.
- Whenever cwnd is greater than ssthresh, upon each SACK arrival that advances the Cumulative TSN Ack Point, increase partial_bytes_acked by the total number of bytes of all new chunks acknowledged in that SACK including

chunks acknowledged by the new Cumulative TSN Ack and by Gap Ack Blocks.

- When partial_bytes_acked is equal to or greater than cwnd and before the arrival of the SACK the sender had cwnd or more bytes of data outstanding (i.e., before arrival of the SACK, flightsize was greater than or equal to cwnd), increase cwnd by MTU, and reset partial_bytes_acked to (partial_bytes_acked - cwnd).
- Same as in the slow start, when the sender does not transmit DATA on a given transport address, the cwnd of the transport address should be adjusted to max (cwnd / 2, 2*MTU) per RTO.
- When all of the data transmitted by the sender has been acknowledged by the receiver, partial_bytes_acked is initialized to 0.

### *3.9.5 Packet Loss Detection*

Upon detection of packet losses from SACK, an endpoint should do the following:

Ssthresh = max (cwnd/2, 2*MTU)

Cwnd = ssthresh

Basically, a packet loss causes cwnd to be cut in half.

When the T3-rtx timer expires on an address, SCTP should perform slow start by:

Ssthresh = max (cwnd/2, 2*MTU)

Cwnd = 1*MTU

and assure that no more than one SCTP packet will be in flight for that address until the endpoint receives acknowledgement for successful delivery of data to that address.

### *3.9.6 Fast Retransmit on Gap Reports*

In the absence of data loss, an endpoint performs delayed acknowledgement. However, whenever an endpoint notices a hole in the arriving TSN sequence, it SHOULD start sending a SACK back every time a packet arrives carrying data until

the hole is filled. Whenever an endpoint receives a SACK that indicates some TSN(s) missing, it SHOULD wait for 3 further miss indications (via subsequent SACK's) on the same TSN(s) before taking action with regard to Fast Retransmit. When the TSN(s) is reported as missing in the fourth consecutive SACK, the data sender shall:

- Mark the missing DATA chunk(s) for retransmission,
- Adjust the ssthresh and cwnd of the destination address to which the missing DATA chunks were last sent, according to the formula described in Section 2.9.2
- Determine how many of the earliest (i.e. lowest TSN) DATA chunks marked for retransmission will fit into a single packet, subject to constraint of the path MTU of the destination transport address to which the packet is being sent. Call this value K. Retransmit those K DATA chunks in a single packet.
- Restart T3-rtx timer only if the last SACK acknowledged the lowest outstanding TSN number sent to that address, or the endpoint is retransmitting the first outstanding DATA chunk sent to that address.

A straightforward implementation of the above keeps a counter for each TSN hole reported by a SACK. The counter increments for each consecutive SACK reporting the TSN hole. After reaching 4 and starting the fast retransmit procedure, the counter resets to 0. Because cwnd in SCTP indirectly bounds the number of outstanding TSN's, the effect of TCP fast-recovery is achieved automatically with no adjustment to the congestion control window size.

*Chapter 4*

# 4 INTRODUCTION TO TCP

Transmission Control Protocol was specifically designed to provide a reliable end-to-end byte stream over an unreliable internetwork. An internetwork differs from a single network because different parts may have wildly different topologies, bandwidths, delays, packet sizes, and other parameters. TCP was designed to dynamically adapt to properties of the internetwork and to be robust in the face of many kinds of failures.
TCP was formally defined in [15]. As time went on, various errors and inconsistencies were detected, and the requirements were changed in some areas. These clarifications and some bug fixes are detailed in RFC 1122. Extensions are given in RFC 1323.
The IP layer gives no guarantee that datagram will be delivered properly, so it is up to TCP to time out and retransmit them as need be. Datagram that do arrive may well do so in the wrong order; it is also up to TCP to reassemble them into messages in the proper sequence. In short, TCP must furnish the reliability that most users want and that IP does not provide.

## 4.1 TCP Connection Establishment

Connections are established in TCP by means of the three-way handshake. To establish a connection, one side, say, the server passively waits for an incoming connection by executing the LISTEN and ACCEPTS primitives, either specifying a specific source or nobody in particular. The other side, say, the client, executes a CONNECT primitive, specifying the IP address and port to which it wants to connect, the maximum TCP segment size it is willing to accept, and optionally some user data (e.g., a password). The CONNECT primitive sends a TCP segment with the SYN bit on and ACK bit off and waits for a response.
When this segment arrives at the destination, the TCP entity there checks to see if there is a process that has done a LISTEN on the port given in the Destination port field. If not, it sends a reply with the RST bit on to reject the connection.

## 4.2 TCP Connection Release

Although TCP connections are full duplex, to understand how connections are released it is best to think of them as a pair of simplex connections. Each simplex connection is released independently of its sibling. To release a connection, either party can send a TCP segment with the FIN bit set, which means that it has no more data to transmit. When the FIN is acknowledged, that direction is shut down for new data. Data may continue to flow indefinitely in the other direction, however. When both directions have been shut down, the connection is released. Normally, four TCP segments are needed to release a connection, one FIN and one ACK for each direction. However, it is possible for the first ACK and the second FIN to be contained in the same segment, reducing the total count to three.

## 4.3 TCP Transmission Policy

Window management in TCP is not directly tied to acknowledgements as it is in most data link protocols. For example, suppose the receiver has a 4096-byte buffer. If the sender transmits a 2048-byte segment that is correctly received, the receiver will acknowledge the segment. However, since it now has only 2048 bytes of buffer space (until the application removes some data from the buffer), it will advertise a window of 2048 starting at the next byte expected.

Now the sender transmits another 2048 bytes, which are acknowledged, but the advertised window is 0. The sender must stop until the application process on the receiving host has removed some data from the buffer, at which time TCP can advertise a larger window.

When the window is 0, the sender may not normally send segments, with two exceptions. First, urgent data may be sent, for example, to allow the user to kill the process running on the remote machine. Second, the sender may send a 1-byte segment to make the receiver renounce the next byte expected and window size. The TCP standard explicitly provides this option to prevent deadlock if a window announcement ever gets lost.

## 4.4 TCP Congestion Control

When the load offered to any network is more than it can handle, congestion builds up. The Internet is no exception. Although the network layer also tries to manage

congestion, most of the heavy lifting is done by TCP because the real solution to congestion is to slow down the data rate.

The first step in managing congestion is detecting it. Nowadays, packet loss due to transmission errors is relatively rare because most long-haul trunks are fiber (although wireless networks are a different story). Consequently, most transmission timeouts on the Internet are due to congestion. All the Internet TCP algorithms assume that timeouts are caused by congestion and monitor timeouts for signs of trouble the way miners watch their canaries.

When a connection is established, a suitable window size has to be chosen. The receiver can specify a window based on its buffer size. If the sender sticks to this window size, problems will not occur due to buffer overflow at the receiving end, but they may still occur due to internal congestion within the network.

The Internet solution is to realize that two potential problems exist network capacity and receiver capacity and to deal with each of them separately. To do so, each sender maintains two windows: the window the receiver has granted and a second window, the congestion window. Each reflects the number of bytes the sender may transmit. The number of bytes that may be sent is the minimum of the two windows. Thus, the effective window is the minimum of what the sender thinks is all right and what the receiver thinks is all right. If the receiver says "Send 8 KB" but the sender knows that bursts of more than 4 KB clog the network, it sends 4 KB. On the other hand, if the receiver says "Send 8 KB" and the sender knows that bursts of up to 32 KB get through effortlessly, it sends the full 8 KB requested.

The congestion window keeps growing exponentially until either a timeout occurs or the receiver's window is reached. The idea is that if bursts of size, say, 1024, 2048, and 4096 bytes work fine but a burst of 8192 bytes gives a timeout, the congestion window should be set to 4096 to avoid congestion. As long as the congestion window remains at 4096, no bursts longer than that will be sent, no matter how much window space the receiver grants. This algorithm is called slow start, but it is not slow at all (Jacobson, 1988). It is exponential. All TCP implementations are required to support it. It uses a third parameter, the threshold, initially 64 KB, in addition to the receiver and congestion windows. When a timeout occurs, the threshold is set to half of the current

congestion window, and the congestion window is reset to one maximum segment. Slow start is then used to determine what the network can handle, except that exponential growth stops when the threshold is hit. From that point on, successful transmissions grow the congestion window linearly (by one maximum segment for each burst) instead of one per segment. In effect, this algorithm is guessing that it is probably acceptable to cut the congestion window in half, and then it gradually works its way up from there.

## 4.5 Differences between TCP and SCTP

In Table I we describe other differences between the two protocols. The first three rows compare the messages exchanged during TCP connection/SCTP association setup & shutdown. Half-open in the third row refers to a situation where one endpoint has finished its data transfer while expecting to receive further data from its correspondent endpoint, i.e. the connection/association is open only for one direction. TCP supports the half-open connection through four-way handshake shutdown sequence [16], while SCTP uses a three-way handshake for shutdown, but and does not support half-open association. The fourth and fifth rows of the table relate to the delivery of segments to the application at the receiver. TCP only supports strict ordered delivery, and can result in HOL blocking in some cases. SCTP can independently deliver to application layer the received segments that belong to different streams, provided that the sequence within the steam is preserved; SCTP can also support unordered delivery optionally, which is not possible in TCP. The next comparison considers message boundary after transmission by the transport layer protocols. TCP is a stream oriented protocol, and the application data are treated as a continuous byte stream instead of discrete messages. Therefore, the developers must add their own markings between messages, and have to use TCP PUSH flag to ensure that the complete message is received within a reasonable time. By comparison, SCTP is message-oriented. As long as there is space in the receiver buffer, the whole message is delivered without ever getting mixed with another message. The last two rows of Table I relate to the keep-alive messages. A *keep-alive* mechanism periodically probes

the other end of a connection when the connection is otherwise idle, even when there is no data to be sent. In TCP, whether this mechanism should be implemented by the transport layer or by the application itself is highly controversial. The opponents of implementing keep-alive in TCP think that this mechanism will unnecessarily waste bandwidth. If a specific TCP implementation chooses to implement the keep-alive mechanism, the default value of heart-beat interval shouldn't be less than two hours.SCTP designers believe that the ability to monitor the reach ability of the peer's address is crucial in high-availability applications. For example, in the SS7 network, it is desirable to receive a link failure alarm as soon as possible to take care of the problem immediately. In this sense, conserving bandwidth is not a principal consideration. Therefore, the keep-alive heartbeat is provided in SCTP as a standard mechanism instead of relying on implementations, as in TCP. In the case of SCTP, the default heartbeat interval is also reduced to a small value of 30 seconds. This comparison of the two transport layer protocols clearly reveals the improvements of SCTP over TCP. These improvements reflect the better understanding of the deficiencies of TCP by the research community during the past twenty years.

Table 4-1 : TCP and SCTP Comparison

| Protocol | TCP | SCTP |
|---|---|---|
| Setup messages | three-way handshake | four-way handshake |
| Shutdown messages | four-way handshake | three-way handshake |
| Half-open support | supported | not supported |
| Ordered delivery | strict ordered | ordered within stream |
| Unordered delivery | not supported | supported |
| Message boundary | no boundary stream-oriented | boundary preserved message-oriented |
| Multi-homing | not supported | supported |
| SACK support | optional | mandatory |
| Keep-alive heartbeat | optional | mandatory |
| Heartbeat interval | $\geq$ 2 hours | 30 secs by default |

## 4.6 TCP Sack

SACK is an extension to TCP that uses selective ACKs in addition to the cumulative ACKs. The cumulative ACK acknowledges the reception of all the data within a connection with a sequence number less than a certain number, whereas the selective ACK acknowledges the reception of non-contiguous ranges of sequence numbers. The cumulative ACK is the main mechanism to detect packet losses in TCP. If a TCP end-point receives packets 1, 2, 3, 5, and 7, it will send ACK (1), ACK (2), ACK (3), ACK (3), and ACK (3) again. When the sender receives ACK (3) multiple times, it knows that packet 4 was lost and thus has to be retransmitted. However, the sender

does not know which packets with a higher sequence number than 4 arrived successfully at the peer and which ones were lost as well. If the receiver had used the SACK TCP option it would have returned ACK (3)-SACK (5) and (7). With this information the sender is able to retransmit not only packet 4, but also packet 6. Performance measurements [8] show that TCP SACK recovers better than TCP Reno and New Reno from multiple losses in a single TCP window.

### *4.6.1 Difference between TCP SACK and SCTP*

There are only two differences between TCP SACK and SCTP. The 40 bytes available for TCP options can only carry a maximum of four SACK blocks per TCP segment, while SCTP does not impose any limit on the number of SACK chunks in a packet. Secondly, TCP SACK uses the information in SACKs only to retransmit lost segments, while SCTP uses this information to perform both flow and congestion control. SCTP uses SACKed DATA chunks to increase its congestion window, whereas TCP SACK does not. In this way SCTP achieves a slightly higher window growth rate than TCP SACK under congestion when both DATA chunks and SACKs get lost. SCTP is "TCP-friendly," that is, it uses about the same amount of bandwidth as a TCP connection under the same network conditions. This is not surprising since SCTP and TCP both use the same window-based congestion control. The sender implements a congestion window (cwnd) that limits the number of bytes that can be injected into the network at a given point of time. The rate at which cwnd grows depends on the state of the connection. If no congestion is detected, the slow start algorithm doubles cwnd every time a whole window of data is acknowledged. Under packet loss, the congestion avoidance algorithm increases cwnd more slowly, namely, linearly rather than exponentially.

*Chapter 5*

# 5 EXPERIMENTAL ANALYSIS

Experiments are performed in Network Simulator 2, having built in capability to support SCTP and TCP Sack. NS-2 simulator covers a large number of application, protocols, network types, network elements and different traffic models. NS-2 is an object oriented simulator, written in C++, with an OTcl interpreter as a front end. The simulator supports a class hierarchy in C++ (also called the compiled hierarchy in this document), and a similar class hierarchy within the OTcl interpreter (also called the interpreted hierarchy in this document). The two hierarchies are closely related to each other; from the user's perspective, there is a one-to-one correspondence between a class in the interpreted hierarchy and one in the compiled hierarchy. The root of this hierarchy is the class TclObject. Users create new simulator objects through the interpreter; these objects are instantiated within the interpreter, and are closely mirrored by a corresponding object in the compiled hierarchy.

NS-2 uses two languages because simulator has two different kinds of things it needs to do. On one hand, detailed simulations of protocols require a systems programming language which can efficiently manipulate bytes, packet headers, and implement algorithms that run over large data sets. For these tasks run-time speed is important and turn-around time (run simulation, find bug, fix bug, recompile, re-run) is less important. On the other hand, a large part of network research involves slightly varying parameters or configurations, or quickly exploring a number of scenarios. In these cases, iteration time (change the model and re-run) is more important. Since configuration runs once (at the beginning of the simulation), run-time of this part of the task is less important. Ns provide an environment where we can simulate real network and analysis the behavior of different network parameters.

## 5.1 Performance Metrics

Following performance metrics are analyzed during our study.

### *5.1.1 Throughput*

We have calculated the throughput using this formula

$$Throughput = \frac{Number\ of\ \text{Re}ceived\ Packets}{Number\ of\ Send\ Packets}$$

### *5.1.2 Delay*

We have calculated Delay using this formula

$$Delay = Tr - Ts \qquad \text{(Ts and Tr taken from Trace File)}$$

Where Ts is equal to the time stamp of packet sending and Tr is time stamp of packet receiving. We have taken mean delay for our results and calculated with this

$$Mean\ Delay = \frac{Total\ Delay}{N}$$

Where N is the total number of packet send in a simulation time.

## 5.2 TCP and SCTP Source Configurations

We want to calculate the delay, throughput and impact of packet loss on performance of protocols so following configuration is used for the TCP and SCTP Sources for this purpose.

1. Since TCP use only one connection between the source and destination, SCTP is configuration to use only one stream.
2. The IP payload for both protocols is 1500 byte.
3. SCTP configured for ordered delivery of data.
4. Sack option is mandatory for SCTP, so TCP therefore also use Sack.

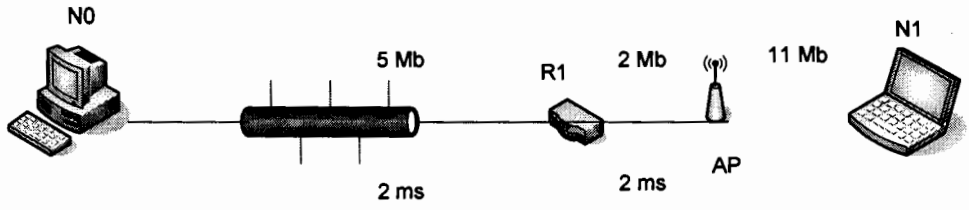
## 5.3 Experimental Setup

### *5.3.1 Networks Scenarios*

In exploring the performance of SCTP, and TCP SACK .We studied the behavior in heterogeneous network scenarios. We used two networks scenarios.

- Ethernet to Wireless
- Wireless to Ethernet

### *5.3.2 Scenario 1: Ethernet to Wireless*




5.1 Ethernet to Wireless

In the first scenario Ethernet to Wireless in Figure 5.1 we used wired node N0 act as a sender and N1 act as a receiver. In the topology R1 act as a Router connected with the access point (AP) with 2 Mb link bandwidth and 2ms of propagation delay. Node N0 connect with R1 having link bandwidth of 5Mb. The link bandwidth between access point and N1 is 11 Mb normally called 802.11b standard. DROPTAIL queuing algorithm is used with queue size of 50 packets. Total simulation time is 100 sec. FTP was used as a traffic source so that the continuous stream of bytes were transfer from source to destination to model the large file transfer.



#### *5.3.2.1 Delay*

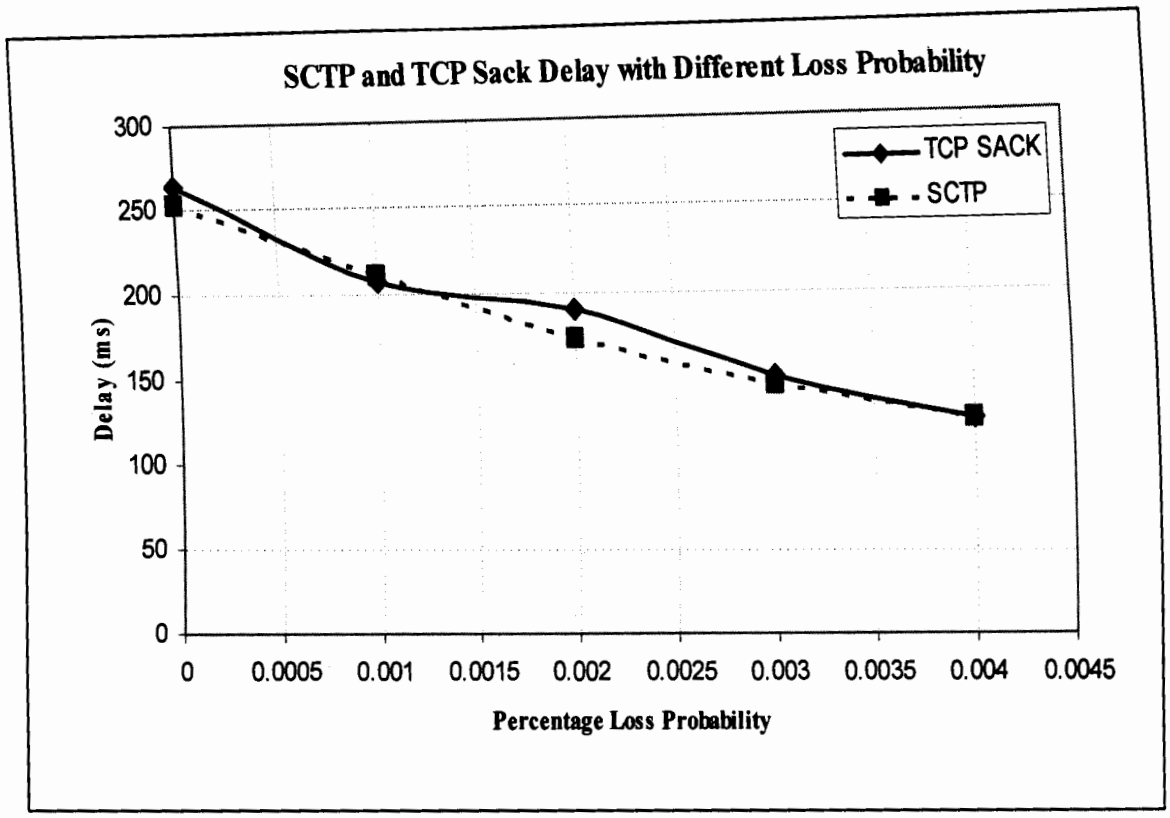
Table 5.1 shows the delay with and without different loss probabilities. To ensure fairness among transport protocols we set the error module in such a way that both SCTP source and TCP SACK source drop approximately the same numbers of

packets during transmission with loss probability in the range of 0.1% - 0.4%. It can be analyzed from the table that by increasing the loss probability the mean delay decreases this is because as throughput decreases the mean delay also decreases due to less queuing delay.

Table 5-1 SCTP and TCP SACK Delay with Different Loss Probability

| Loss Probability | TCP Sack | | SCTP | |
|---|---|---|---|---|
| | Delay (ms) | Variance (ms) | Delay (ms) | Variance (ms) |
| No loss | 263.67 | 69.31 | 252.70 | 63.92 |
| 0.1 % | 206.97 | 46.13 | 211.49 | 47.66 |
| 0.2 % | 190.12 | 40.09 | 172.76 | 33.11 |
| 0.3 % | 151.25 | 27.02 | 145.76 | 24.25 |
| 0.4 % | 126.77 | 19.37 | 126.29 | 19.46 |

The following Figure 5.2. Shows the SCTP and TCP SACK delay with different loss probabilities where x-axis shows percentage loss probabilities and y-axis shows mean delay in milliseconds. Results shows that SCTP has less mean delay than TCP sack.




5.2 SCTP and TCP SACK Delay with Different Loss Probability

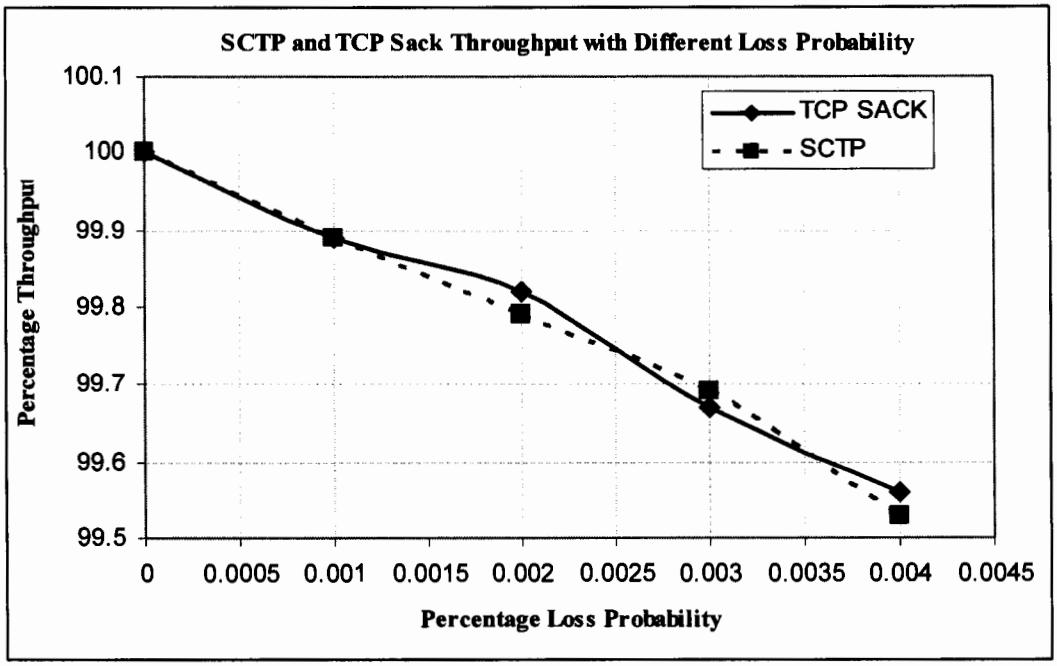
### *5.3.2.2 Impact of Loss probability on Throughput*

Figure 5.3 shows percentage throughput achieved by the SCTP and TCP SACK for the different loss probabilities. It can be observed that by increasing the loss probability the percentage throughput decreases this is because as the frequency of drops increases both TCP SACK and SCTP suffer from numerous drops.

When TCP SACK or SCTP sources detect a packet loss (such as due to error in the link), it retransmits the lost packet based on information received by the receiver in the acknowledgement. The performance of the protocol depends on the retransmission mechanism. On detection packet loss sources reduce its cwnd to half.

Table 5-2 : SCTP and TCP SACK with Different Loss Probability

| Loss Probability | TCP Sack | SCTP |
|---|---|---|
| | % age Throughput | % age Throughput |
| No loss | 100 | 100 |
| 0.1 % | 99.89 | 99.89 |
| 0.2 % | 99.82 | 99.79 |
| 0.3 % | 99.67 | 99.69 |
| 0.4 % | 99.56 | 99.53 |




5.3 : SCTP and TCP SACK with Different Loss Probability

Figure 5.4 shows congestion window. SCTP cwnd is less than TCP SACK but it sends almost same number of packets this is because

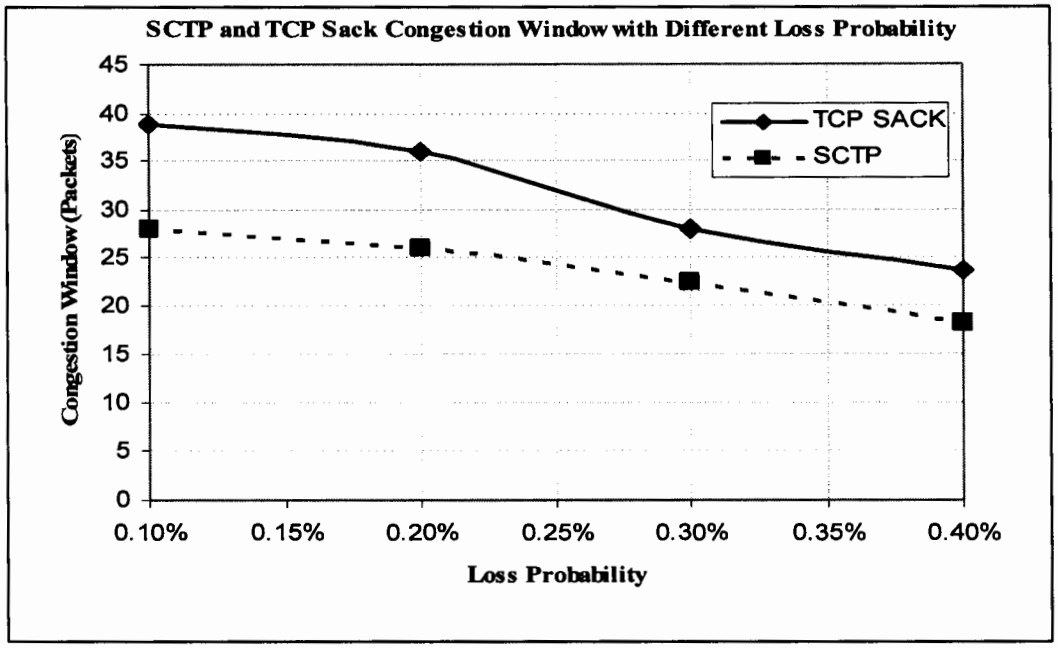
(1) The TCP SACK performs cwnd and ssthresh reductions in whole packets, whereas SCTP reduces these variables in bytes.

(2) an end point congestion window is not tied to its cumulative TSN ack point, as dup Sacks come in, even though, they may not advance the cumulative TSN ack point an end point can still used them to clock out new data. That is the data newly

acknowledge by the SACK diminishes the amount of data now in flight to less than the congestion window and so the current unchanged values of congestion window now allows new data to be send. On the other hand, the increase of congestion window must be tied to the cumulative TSN ack point advancement as specified above. Otherwise the duplicate Sacks will not only clock out new data, but also will adversely clock out more new data then what has just left the network, during a time of possible congestion.

Table 5-3 : SCTP and TCP SACK Congestion Window with Different Loss Probability

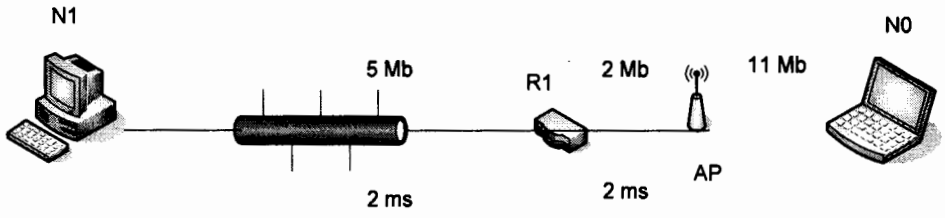
| Loss Probability | TCP Sack | SCTP |
|---|---|---|
| | Cwnd | Cwnd |
| 0.1 % | 38.86 | 27.97 |
| 0.2 % | 36.01 | 25.88 |
| 0.3 % | 27.87 | 22.37 |
| 0.4 % | 23.67 | 18.15 |




5.4 SCTP and TCP SACK Congestion Window with Different Loss Probability

### *5.3.3 Scenario 2: Wireless to Ethernet*

#### *5.3.3.1 Topology*




5.5 Wireless to Ethernet

In the second scenario Wireless to Ethernet in Figure 5.5 we used wireless node N0 act as a sender and N1 wired node act as a receiver. In the topology R1 act as a Router connected with the access point (AP) with 2 Mb link bandwidth and 2ms of propagation delay. Node N1 connect with R1 having link bandwidth of 5Mb and propagation delay of 2ms. The link bandwidth between access point and N0 is 11 Mb called 802.11b standard. DROPTAIL queuing algorithm is used with queue size of 50 packets. Total simulation time is set for 100 sec. FTP was used as a traffic source so that the continue stream of bytes were transfer from source to destination to model the large file transfer.

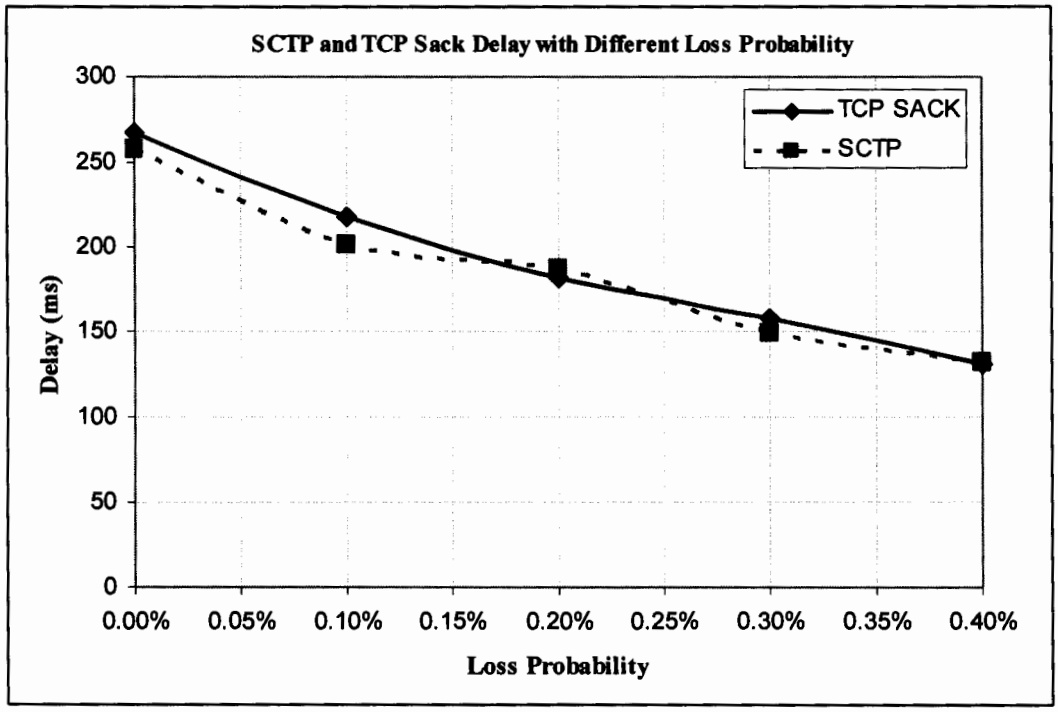
#### *5.3.3.2 Delay*

Table 5.2 shows the delay with and without different loss probabilities. To ensure fairness among transport protocols we set the error module in such a way that both SCTP source and TCP SACK source drop approximately the same numbers of packets during transmission with loss probability in the range of 0.1% - 0.4%. It can be analyzed from the table that by increasing the loss probability the mean delay decreases this is because of as throughput decreases the mean delay also decreases due to less queuing delay.

Table 5-4 SCTP and TCP SACK Delay with Different Loss Probability

| Loss Probability | TCP Sack | | SCTP | |
|---|---|---|---|---|
| | Delay | Variance | Delay | Variance |
| No loss | 267.23 | 71.41 | 257.75 | 66.50 |
| 0.1 % | 217.57 | 50.64 | 200.75 | 43.58 |
| 0.2 % | 181.48 | 37.49 | 186.72 | 39.34 |
| 0.3 % | 157.38 | 28.87 | 148.92 | 25.74 |
| 0.4 % | 130.29 | 19.99 | 131.74 | 20.40 |

The following Figure 5.6 shows the SCTP and TCP SACK delay with different loss probabilities where x-axis shows percentage loss probabilities and y-axis shows mean delay in milliseconds. Results shows that SCTP has less mean delay than TCP sack.



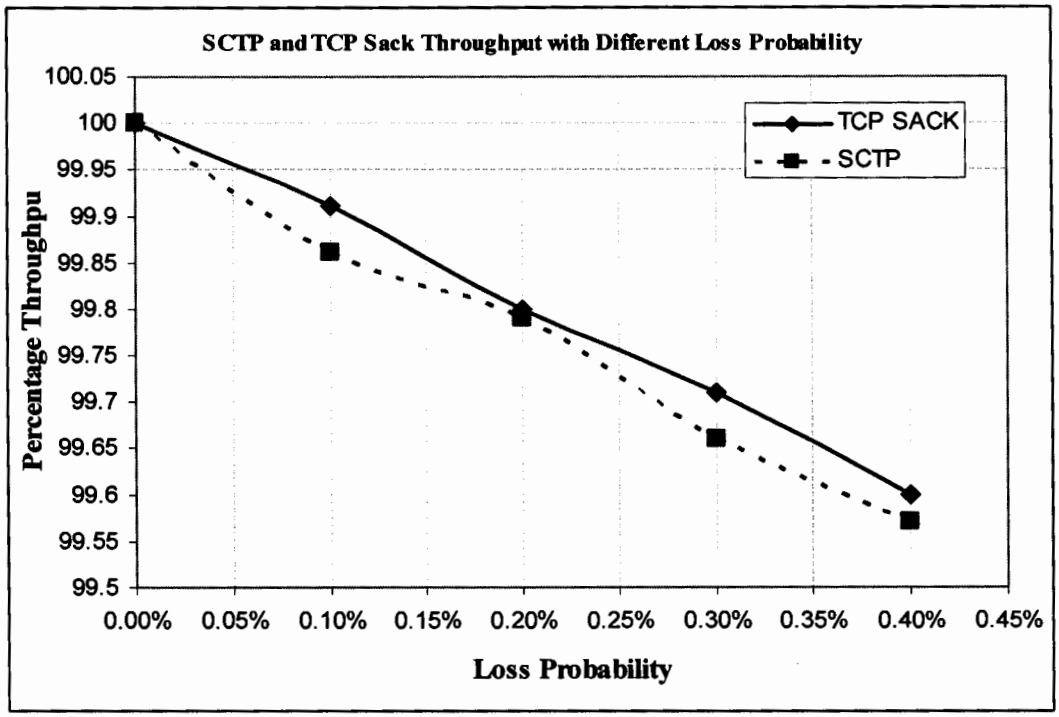

5.6 SCTP and TCP SACK Delay with Different Loss Probability

### *5.3.3.3 Impact of Loss Probability on Throughput*

Figure 5.7 shows the throughput achieved using TCP SACK and SCTP. Same behavior can be observed as in Eth to Wireless.

Table 5-5 : SCTP and TCP SACK Throughput with Different Loss Probability

| Loss Probability | TCP Sack | SCTP |
|---|---|---|
| | % age Throughput | % age Throughput |
| No loss | 100 | 100 |
| 0.1 % | 99.91 | 99.86 |
| 0.2 % | 99.80 | 99.79 |
| 0.3 % | 99.71 | 99.66 |
| 0.4 % | 99.60 | 99.57 |

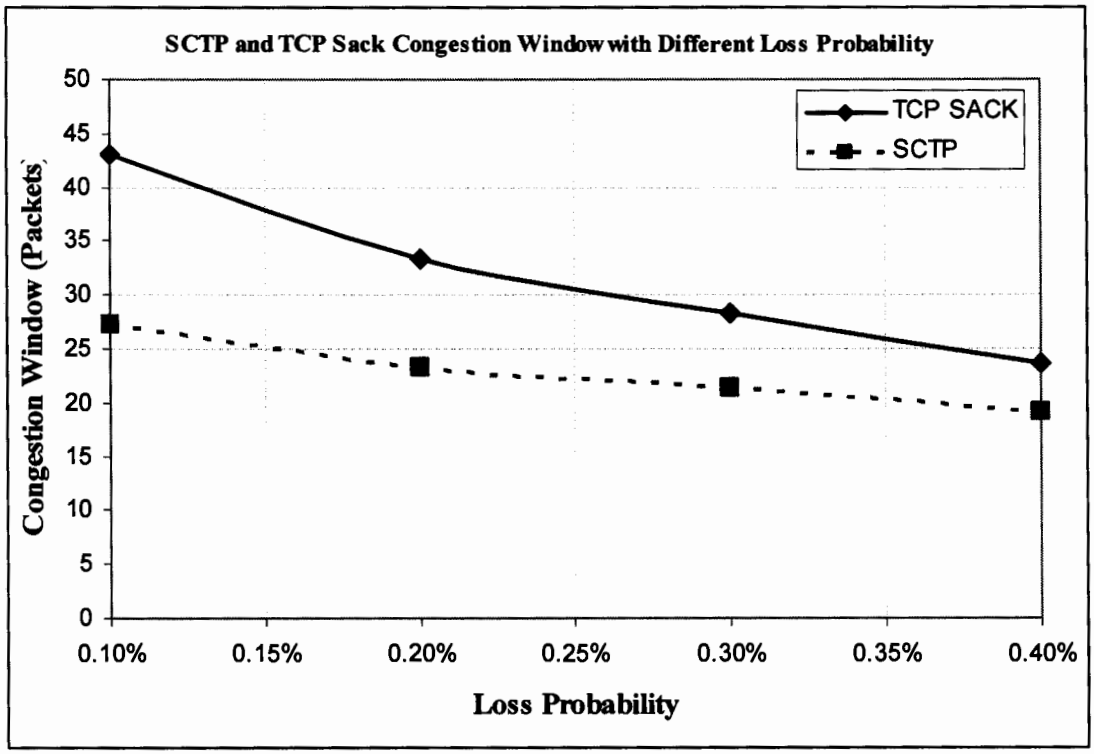



5.7 SCTP and TCP SACK Throughput with Different Loss Probability

Figure 5.8 shows the cwnd of TCP SACK and SCTP. Same behavior can be observed as in Eth to Wireless.

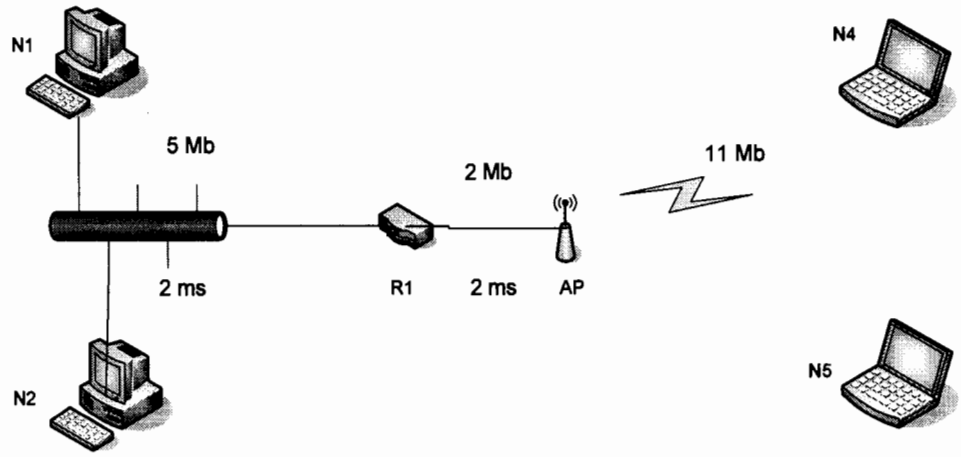
Table 5-6 : SCTP and TCP SACK Congestion Window with Different Loss Probability

| Loss Probability | TCP Sack | SCTP |
|---|---|---|
| | Cwnd | Cwnd |
| 0.1 % | 43.05 | 27.18 |
| 0.2 % | 33.27 | 23.35 |
| 0.3 % | 28.36 | 21.41 |
| 0.4 % | 23.55 | 19.11 |




5.8 SCTP and TCP SACK Congestion Window with Different Loss Probability

## 5.4 Competing Traffic




5.9 Competing Traffic

The topology used is shown in Figure 5.9 where N5 acts as SCTP source and N2 acts As SCTP receiver whereas N4 act as a TCP source and N1 act as a TCP receiver. R1 act as a router. Using the ftp traffic generator generated FTP traffic So that a continuous stream of packets (bytes) is transferred from the source to destination.

The Table 5.3 show the delay and throughput in competing where both nodes share the link bandwidth. The result show that both protocol faces similar delay and similar throughput while sharing the link bandwidth. This leads to the conclusion that the introduction of this new protocol SCTP into a TCP/IP network does not degrade the performance of the existing protocols.

Table 5-7 Competing Traffic

| | Delay (ms) | Sent packets | Received Packets | Throughput (%) |
|---|---|---|---|---|
| TCP | 251.46 | 41641 | 41510 | 99.68 |
| SCTP | 251.83 | 39055 | 38791 | 99.32 |
| | | | | |

## 5.5 Special case: multi-streaming

A special case to show effect of multi-streaming which is built-in feature of SCTP is configured. This shows that as we have introduced the multi-streaming feature of SCTP it led in performance over TCP.

Same amount of data in both TCP and SCTP results less packet loss in SCTP and less packet delay in SCTP. This case demonstrated that when SCTP is configured for its features it performs better. The result of simulation are displayed in Table 5-8

Table 5-8 Special Case Multi-streaming

| Protocol | Streams | Sent Packets | Received Packets | Average Delay (ms) |
|---|---|---|---|---|
| SCTP | 2 | 14,709 | 14,675 | 163.90 |
| TCP | 1 | 14,686 | 14,660 | 190.12 |

# 6 CONCLUSION

After the experiments the SCTP has less mean delay than TCP SACK even having congestion window through out less than TCP SACK. While Loss Probability is less the both protocols performed almost equal or some times TCP performs a better with improved throughput, but at higher loss probability SCTP taken over in throughput due to its congestion control mechanism.

Since SCTP has four-way handshake it takes little more time than TCP to make an association, Even though it performed overall better.

Both protocols exhibited similar performance in competing traffic.

Since we have used partial functionality of the SCTP to make the simulation and comparison fair, SCTP performs far better when it is left unrestricted to use full functionality.

Over all performance of SCTP remained better than TCP, especially when there is a lossy media like wireless, when a dropped packet by the media creates an illusion like there is congestion occurred which reduces the congestion window size erroneously. More over a stoppage for at least 3 acknowledgments time in TCP but not in SCTP. This make a good packet transfer count on SCTP.

In our special case for introduction of multi-streaming in SCTP for same amount of packets, SCTP has performed better in average packet delay due to multiple streams, its all due to deficiency of TCP while head of line block phenomenon occurs, it is suppressed automatically due multiple head of lines in SCTP.

# 7 BIBLIOGRAPHY


[1] http://www.isi.edu/nsnam/ns/index.html

[2] Andreas Jungmaier, Michael Schopp, Michael Tiixen, Siemens AG, "Performance Evaluation of the Stream Control Transmission Protocol"

[3] Shaojian Fu, Mohammed Atiquzzaman School of Computer Science University of Oklahoma, Norman, "SCTP over Satellite Networks".

[4] Armando L. Caro Jr., Keyur Shah, Janardhan R. Iyengar, Paul D. Amer Protocol Engineering Lab Computer and Information Sciences University of Delaware "SCTP and TCP Variants: Congestion Control under Multiple Losses".

[5] Preethi Natarajan1, Janardhan R. Iyengar1, Paul. D. Amer1 and Randall Stewart Protocol Engineering Lab, CIS Dept University of Delaware "SCTP: An innovative Transport layer protocol for the web

[6] Pekka Nikander and Gonzalo Camarillo Ericsson Research, Jorvas, Finland "Effects of Mobility and Multihoming on Transport-Protocol Security"

[7] D. Nagamalai and J.-K. Lee. "Performance of sctp over high speed wide area networks, Dec. '04."

[8] R. Rajamani, S. Kumar, and N. Gupta. "Sctp versus tcp: Comparing the performance of transport protocols for web traffic".

[9] S. Kang and M. Fields. "Experimental study of the sctp compared to tcp". Technical report, Department of Electrical Engineering of Texas A& M University, '03.

[10] L. Ma, F.Yu and V. C. M. Leung. "Modeling sctp throughput in integrated Wlan/cellular networks, May'05."

[11] W. Ivancic, S. Fu, and M. Atiquzzaman. "Effect of delay spike on sctp, tcp Reno

and Eifel in a Wireless mobile environment", Oct. '02.

[12] A. Kumar, L. Jacob, and A. L. Ananda. Sctp vs. tcp: Performance comparison

in Manets." In Proc. Of the 29th Annual IEEE International Conference on Local

Computer Networks (LCN'04), Washington, DC, USA, '04. IEEE Computer Society.

[13] Salvatore Loreto, Antonio Pescape and Giorgio Ventre "Measuring SCTP

Through put and Jitter over Heterogeneous Networks"

[14] RFC 2960

[15] RFC 793 "Transmission Control Protocol"

[16] R. Braden et. al., "Requirements for Internet hosts – communication layers."IETF

RFC 1122, October 1989

# 8 APPENDICES

## 8.1 Code Listing snippets

```
### This simulation is an example of combination of wired and
wireless
### topologies.


global opt
set opt(chan)       Channel/WirelessChannel  ;#set what
channel is to use
set opt(prop)       Propagation/TwoRayGround  ;#wireless
propagation model, 2Ray
set opt(netif)      Phy/WirelessPhy   ;#network interface
set opt(mac)        Mac/802_11    ;#MAC scheme 802.11 B/G
set opt(ifq)        Queue/DropTail/PriQueue  ;#Queue is drop
tail, priority queue
set opt(ll)         LL     ;#Link layer
set opt(ant)        Antenna/OmniAntenna   ;#omni antenna
set opt(x)             270       ;#signal grid width
set opt(y)              270       ;#signal grid height
set opt(ifqlen)         50       ;#interface queue length
set opt(tr)          wireless.tr   ;#trace filename
set opt(namtr)       wired-and-wireless.nam  ;#network
animator required data
set opt(nn)             2                        ;#number of
wire less nodes
set opt(adhocRouting)   DSDV                     ;#routing
algo used others DSR,AODV
set opt(stop)           100          ;#used for simulation
secs
set num_wired_nodes      3    ;#no of wired nodes
set num_bs_nodes         1    ;#AP/ Base station count

 Trace set show_sctphdr_ 1 ;# show sctp trace format, default
is tcp
set ns_   [new Simulator];#instance of simulator

# set up for hierarchical routing
  $ns_ node-config -addressType hierarchical ;#hierarchical
address type. expicit node def.
  AddrParams set domain_num_ 2 ;#no of domain, first is wired
and second wireless
  lappend cluster_num 1 1      ;# 1 1, one wired one wireless
  AddrParams set cluster_num_ $cluster_num
```

```
  lappend eilastlevel 3 2
  AddrParams set nodes_num_ $eilastlevel

  set tracefd  [open $opt(tr) w]
  $ns_ trace-all $tracefd
  set namtracefd [open $opt(namtr) w]
  $ns_ namtrace-all $namtracefd


set myvar  100
set old_data 0
set old_data1 0
set old_data2 0
set old_data3 0
set old_data4 0
set old_data5 0
set old_data6 0
set old_data7 0
set old_data8 0
set old_data9 0
set ttime 0

proc finish {} {
        global ns_ namtracefd tracefd trace_ch ftcp ttime
        global old_data old_data1 old_data2 old_data3
old_data4
        global old_data5 old_data6 old_data7 old_data8
old_data9

        # close the nam trace file
        $ns_ flush-trace
        close $namtracefd
 close $tracefd

        #close $ftcp

exec awk {
       {
          #  recieve     dest
        if  ($1=="r" && $3=="_4_" && $4=="AGT" && $7=="sctp"
&& $8==1500) {
               # throughput in packets
               old_data1= $6-20
               old_data1=old_data1*8/($2-ttime)
               ttime=$2
               print $2, old_data1
        }
        }
         } wireless.tr > sctp_throughput.data
```

```
exec awk {
        {
         #  recieve     dest
        if ($1=="-" && $4==0 && $5=="sctp" && $6==1500) { {
         print $1 , $2 , $12
                          } }
         if ($1=="r" && $3=="_4_" && $4=="AGT" && $7=="sctp"
&& $8==1500){ {
              print $1 , $2 , $20
                           } }
             }
        } wireless.tr > sr_etoe_sctp.tr


exec awk {
      {

         if ($11=="cwnd:")
 {
        old_data4=$12/1468
        print $2 , old_data4

       }
}
         } trace.sctp > sctp_cwnd.data

 exec awk {
      {
         old_data8=old_data8+$2
         old_data9=old_data9+1
         print old_data8 , old_data9
      }
        } sctp_cwnd.data > avg_cwnd.data

#exec xgraph sctp_throughput.data &
#exec xgraph sctp_drop.data &
#exec  xgraph sctp_cwnd.data &
#exec nam wired-and-wireless.nam &
exit 0
}


$opt(mac) set dataRate_ 11Mb
$opt(mac) set basicRate_ 1Mb

$opt(mac) set bandwidth_ 22.0e6

#Mac/802_11 set dataRate_ 11Mb
```

```
#FHSS (IEEE802.11)
  $opt(mac) set SlotTime_          0.000020  ;#20microsec
  $opt(mac) set SIFS_              0.000010  ;#10microsec
  $opt(mac) set PreambleLength_      144     ;#144 bit
  $opt(mac) set PLCPHeaderLength_     48     ;#48 bits
  $opt(mac) set PLCPDataRate_          1.0e6  ;#1Mbps


    # frequency is 2.4 GHz
   $opt(netif) set freq_ 2.4e+9 ;
   # transmit power
   $opt(netif) set Pt_ 3.3962527e-2 ;
   # Receive sensitivity.
   $opt(netif) set RXThresh_ 6.309573e-12 ;
   $opt(netif) set CSThresh_ 6.309573e-12  ;



  set topo    [new Topography]
  $topo load_flatgrid $opt(x) $opt(y)
  # god needs to know the number of all wireless interfaces
  create-god [expr $opt(nn) + $num_bs_nodes]

  #create wired nodes
  set temp {0.0.0 0.0.1 0.0.2}
  for {set i 0} {$i < $num_wired_nodes} {incr i} {
      set W($i) [$ns_ node [lindex $temp $i]]
  }
  $ns_ node-config -adhocRouting $opt(adhocRouting) \
                   -llType $opt(ll) \
                   -macType $opt(mac) \
                   -ifqType $opt(ifq) \
                   -ifqLen $opt(ifqlen) \
                   -antType $opt(ant) \
                   -propInstance [new $opt(prop)] \
                   -phyType $opt(netif) \
                   -channel [new $opt(chan)] \
                   -topoInstance $topo \
                   -wiredRouting ON \
                   -agentTrace ON \
                   -routerTrace OFF \
                   -macTrace OFF

  set temp {1.0.0 1.0.1 1.0.2 }
  set BS(0) [$ns_ node [lindex $temp 0]]


  #configure for mobilenodes
```

```
  $ns_ node-config -wiredRouting OFF

  for {set j 0} {$j < $opt(nn)} {incr j} {
    set node_($j) [ $ns_ node [lindex $temp \
            [expr $j+1]] ]
    $node_($j) base-station [AddrParams addr2id [$BS(0) node-
addr]]
  }



$BS(0) random-motion 0

  $BS(0) set X_ 1.0
  $BS(0) set Y_ 2.0
  $BS(0) set Z_ 0.0

  $node_(0) set X_ 80.0
  $node_(0) set Y_ 30.0
  $node_(0) set Z_ 0.0

  $node_(1) set X_ 190.0
  $node_(1) set Y_ 30.0
  $node_(1) set Z_ 0.0

#create links between wired and BS nodes
  $ns_ duplex-link $W(1) $W(0) 5Mb 2ms DropTail
  $ns_ duplex-link $W(2) $W(0) 5Mb 2ms DropTail
  $ns_ duplex-link $W(0) $BS(0) 2Mb 2ms DropTail

#position in nam
  $ns_ duplex-link-op $W(1) $W(0) orient right-down
  $ns_ duplex-link-op $W(2) $W(0) orient left-down
  $ns_ duplex-link-op $W(0) $BS(0) orient down

set erormodule [new ErrorModel]
$erormodule unit pkt
$erormodule set rate_ 0.001
$erormodule ranvar [ new RandomVariable/Uniform]
$erormodule drop-target [new Agent/Null]
$ns_ lossmodel $erormodule $W(0) $BS(0)


# setup SCTP connections
  set sctp0 [new Agent/SCTP]
  $ns_ attach-agent $W(1) $sctp0
  $sctp0 set fid_ 1
  $sctp0 set debugMask_ 0x00303000
  $sctp0 set debugFileIndex_ 0
```

```
  $sctp0 set mtu_ 1500
  $sctp0 set dataChunkSize_ 1468
  $sctp0 set numOutStreams_ 1

  set trace_ch [open trace.sctp w]
  $sctp0 set trace_all_ 1
  $sctp0 trace cwnd_
  $sctp0 attach $trace_ch


  set sctp1 [new Agent/SCTP]
  $ns_ attach-agent $node_(0) $sctp1
  $sctp1 set debugMask_ -1
  $sctp1 set debugFileIndex_ 1
  $sctp1 set useDelayedSacks_ 1

  $ns_ connect $sctp0 $sctp1

set ftp1 [new Application/FTP]
  $ftp1 attach-agent $sctp0


  $ns_ at 2 "$ftp1 start"
  $ns_ at 90 "$ftp1 stop"




  for {set i 0} {$i < $opt(nn)} {incr i} {
      $ns_ initial_node_pos $node_($i) 30

   }

  for {set i } {$i < $opt(nn) } {incr i} {
      $ns_ at $opt(stop).5 "$node_($i) reset";
  }
  $ns_ at $opt(stop).7 "$BS(0) reset";

  $ns_ at $opt(stop).7 "finish"

  $ns_ at $opt(stop).9 "puts \"NS EXITING...\" ; $ns_ halt"



  puts "Starting Simulation..."
  $ns_ run
```

## 8.2 Raw generated d data

```
r 37.47155 0 3 sctp 1500 -------D 1 0.0.1.0 1.0.1.0 1 5828 8868 0 5827
r 37.4721 3 0 sctp 68 -------S 0 1.0.1.0 0.0.1.0 1 5827 8906 65535 65535
+ 37.4721 0 1 sctp 68 -------S 0 1.0.1.0 0.0.1.0 1 5827 8906 65535 65535
- 37.4721 0 1 sctp 68 -------S 0 1.0.1.0 0.0.1.0 1 5827 8906 65535 65535
r 37.47421 0 1 sctp 68 -------S 0 1.0.1.0 0.0.1.0 1 5827 8906 65535 65535
+ 37.47421 1 0 sctp 1500 -------D 1 0.0.1.0 1.0.1.0 1 5854 8907 0 5853
- 37.47421 1 0 sctp 1500 -------D 1 0.0.1.0 1.0.1.0 1 5854 8907 0 5853
+ 37.47421 1 0 sctp 1500 -------D 1 0.0.1.0 1.0.1.0 1 5855 8908 0 5854
r 37.47423 _4_ AGT --- 8868 sctp 1500 [13a 1 0 800] ------- [1:0 4194305:0
- 37.47555 0 3 sctp 1500 -------D 1 0.0.1.0 1.0.1.0 1 5830 8871 0 5829
- 37.47661 1 0 sctp 1500 -------D 1 0.0.1.0 1.0.1.0 1 5855 8908 0 5854
r 37.47755 0 3 sctp 1500 -------D 1 0.0.1.0 1.0.1.0 1 5829 8870 0 5828
r 37.47861 1 0 sctp 1500 -------D 1 0.0.1.0 1.0.1.0 1 5854 8907 0 5853
+ 37.47861 0 3 sctp 1500 -------D 1 0.0.1.0 1.0.1.0 1 5854 8907 0 5853
r 37.47991 _4_ AGT --- 8870 sctp 1500 [13a 1 0 800] ------- [1:0 4194305:0
s 37.47991 _4_ AGT --- 8909 sctp 48 [0 0 0 0] ------- [4194305:0 1:00
r 37.48101 1 0 sctp 1500 -------D 1 0.0.1.0 1.0.1.0 1 5855 8908 0 5854
+ 37.48101 0 3 sctp 1500 -------D 1 0.0.1.0 1.0.1.0 1 5855 8908 0 5854
- 37.48155 0 3 sctp 1500 -------D 1 0.0.1.0 1.0.1.0 1 5831 8873 0 5830
+ 37.48181 3 0 sctp 68 -------S 0 1.0.1.0 0.0.1.0 1 5829 8909 65535 65535
- 37.48181 3 0 sctp 68 -------S 0 1.0.1.0 0.0.1.0 1 5829 8909 65535 65535
r 37.48355 0 3 sctp 1500 -------D 1 0.0.1.0 1.0.1.0 1 5830 8871 0 5829
r 37.48408 3 0 sctp 68 -------S 0 1.0.1.0 0.0.1.0 1 5829 8909 65535 65535
+ 37.48408 0 1 sctp 68 -------S 0 1.0.1.0 0.0.1.0 1 5829 8909 65535 65535
- 37.48408 0 1 sctp 68 -------S 0 1.0.1.0 0.0.1.0 1 5829 8909 65535 65535
r 37.48591 _4_ AGT --- 8871 sctp 1500 [13a 1 0 800] ------- [1:0 4194305:0
r 37.48619 0 1 sctp 68 -------S 0 1.0.1.0 0.0.1.0 1 5829 8909 65535 65535
```

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| + | 37.48619 | 1 | 0 | sctp | 1500 | -------D | 1 | 0.0.1.0 | 1.0.1.0 | | 1 | | 5856 | 8910 | | 0 | 5855 | |
| - | 37.48619 | 1 | 0 | sctp | 1500 | -------D | 1 | 0.0.1.0 | 1.0.1.0 | | 1 | | 5856 | 8910 | | 0 | 5855 | |
| + | 37.48619 | 1 | 0 | sctp | 1500 | -------D | 1 | 0.0.1.0 | 1.0.1.0 | | 1 | | 5857 | 8911 | | 0 | 5856 | |
| - | 37.48755 | 0 | 3 | sctp | 1500 | -------D | 1 | 0.0.1.0 | 1.0.1.0 | | 1 | | 5832 | 8874 | | 0 | 5831 | |
| - | 37.48859 | 1 | 0 | sctp | 1500 | -------D | 1 | 0.0.1.0 | 1.0.1.0 | | 1 | | 5857 | 8911 | | 0 | 5856 | |
| r | 37.48955 | 0 | 3 | sctp | 1500 | -------D | 1 | 0.0.1.0 | 1.0.1.0 | | 1 | | 5831 | 8873 | | 0 | 5830 | |
| r | 37.49059 | 1 | 0 | sctp | 1500 | -------D | 1 | 0.0.1.0 | 1.0.1.0 | | 1 | | 5856 | 8910 | | 0 | 5855 | |
| + | 37.49059 | 0 | 3 | sctp | 1500 | -------D | 1 | 0.0.1.0 | 1.0.1.0 | | 1 | | 5856 | 8910 | | 0 | 5855 | |
| r | 37.49177 | _4_ | AGT | --- | 8873 | sctp | 1500 | [13a | | 1 | 0 | 800] | | ------- | [1:0 | | 4194305:0 | |
| s | 37.49177 | _4_ | AGT | --- | 8912 | sctp | 48 | [0 | | 0 | 0 | 0] | | ------- | [4194305:0 | | 1:00 | |
| r | 37.49299 | 1 | 0 | sctp | 1500 | -------D | 1 | 0.0.1.0 | 1.0.1.0 | | 1 | | 5857 | 8911 | | 0 | 5856 | |
| + | 37.49299 | 0 | 3 | sctp | 1500 | -------D | 1 | 0.0.1.0 | 1.0.1.0 | | 1 | | 5857 | 8911 | | 0 | 5856 | |
| + | 37.49351 | 3 | 0 | sctp | 68 | -------S | 0 | 1.0.1.0 | 0.0.1.0 | | 1 | | 5831 | 8912 | | 65535 | 65535 | |
| - | 37.49351 | 3 | 0 | sctp | 68 | -------S | 0 | 1.0.1.0 | 0.0.1.0 | | 1 | | 5831 | 8912 | | 65535 | 65535 | |
| - | 37.49355 | 0 | 3 | sctp | 1500 | -------D | 1 | 0.0.1.0 | 1.0.1.0 | | 1 | | 5833 | 8876 | | 0 | 5832 | |
| r | 37.49555 | 0 | 3 | sctp | 1500 | -------D | 1 | 0.0.1.0 | 1.0.1.0 | | 1 | | 5832 | 8874 | | 0 | 5831 | |
| r | 37.49578 | 3 | 0 | sctp | 68 | -------S | 0 | 1.0.1.0 | 0.0.1.0 | | 1 | | 5831 | 8912 | | 65535 | 65535 | |
| + | 37.49578 | 0 | 1 | sctp | 68 | -------S | 0 | 1.0.1.0 | 0.0.1.0 | | 1 | | 5831 | 8912 | | 65535 | 65535 | |
| - | 37.49578 | 0 | 1 | sctp | 68 | -------S | 0 | 1.0.1.0 | 0.0.1.0 | | 1 | | 5831 | 8912 | | 65535 | 65535 | |
| r | 37.49769 | _4_ | AGT | --- | 8874 | sctp | 1500 | [13a | | 1 | 0 | 800] | | ------- | [1:0 | | 4194305:0 | |

## 8.3 Post processed data.

### *8.3.1 Packet wise delay calculated*

```
1 14.840072
2 18.160072
3 15.059639
4 18.199639
5 21.859639
6 14.800206
7 18.380206
8 21.920206
9 14.779774
10 18.199774
11 21.739774
12 21.039774
13 24.579774
14 27.939774
15 24.256774
16 28.096774
17 31.616774
18 30.156774
20 30.916774
21 29.696774
22 33.716774
23 37.416774
24 35.856774
25 39.936774
26 43.476774
27 42.036774
28 45.376774
29 49.456774
30 52.496774
31 54.770774
32 53.410774
19 111.256774
33 30.190774
34 30.010774
35 29.890774
36 29.490774
37 30.270774
38 23.956774
39 27.276774
40 31.176774
```

### *8.3.2 Summary of Data*

```
Number of entries read     : 28959
Number of entries sent     : 14487
Number of entries received : 14472
Average delay of entries   : 211.499704
```

Variance of delay is : 47.665602