SCTP MULTI-STREAMING: STUDY OF TRANSMISSION OF PARTIALLY RELIABLE AND RELIABLE DATA USING DYNAMIC STREAM PRIORITIES

By

TASNEEM YUNUS KANPURWALA

Bachelor of Engineering

University of Madras

Madras, India

2002

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 2006

SCTP MULTI-STREAMING: STUDY OF

TRANSMISSION OF PARTIALLY RELIABLE AND

RELIABLE DATA USING DYNAMIC STREAM

PRIORITIES

Thesis Approved:

Dr.Venkatesh Sarangan
Thesis Advisor

Dr. Johnson Thomas

Dr. Debao Chen

Dr. A. Gordon Emslie
Dean of the Graduate College

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Table Page

CHAPTER I

INTRODUCTION

IP technology has successfully relied on both TCP and UDP for years as the workhorses of data transfer [2]. However as the desire for further exploring IP technology for a wider range of commercial application grows, researchers have started to feel that the data transfer services offered by TCP and UDP are inadequate.

One particular application that best exemplifies many of the shortcomings of TCP and UDP is the transportation of telephony signaling messages (SS7) over IP networks. TCP has several key weaknesses in dealing with telephone call control. The first realization came in 1991 when a network broke down while testing and many minutes transpired before the TCP socket gave an error indication. This was quiet unacceptable and thus directly motivated the development of the Stream Control Transmission Protocol (SCTP).

SCTP is a reliable, connection oriented transport protocol operating on top of the connectionless packet service, namely IP, designed to expand the scope beyond TCP and UDP. It is a proposed Internet Engineering Task Force standard (RFC 2960). Like TCP, SCTP provides a reliable, full-duplex connection with support for error-free non-duplicated transfer of messages. Unlike both TCP and UDP, an SCTP connection, called an association, provides novel services such as multi-homing, which allows the end points of a single association to have multiple IP addresses, and multi-streaming, which

allows for independent delivery of data in separate streams.

TCP Limitations

The following limitations of TCP make it hard to meet the rigid timing and reliability requirements of telephony signaling:

- TCP provides both reliable as well as strict order-of-transmission delivery of data. Telephony signaling applications require reliable message transfer with partial ordering of the data, i.e., maintaining an ordered sequence only within some sub-flows of the data. This strict sequence maintenance in TCP introduces unnecessary delay to the overall data transfer service, causing a single lost TCP segment to block delivery of all subsequent data in the stream, up until the lost TCP segment is delivered. This condition has been aptly named as head-of-line blocking, and such excessive delays in telephony signaling may cause service failures and thereby should be controlled.
- The byte-oriented nature of TCP is often an inconvenience to the message based telephony signaling. Applications must add their own record markings to delineate their messages, and they must make explicit use of the push facility to ensure that a complete message is transferred in a reasonable time.
- Providing highly available data transfer service is one of the primary requirements of telephony signaling network. TCP has no built-in support for multihomed hosts. Thus without link or path-level redundancy, the network is vulnerable to link failures.

- For telephony applications, security against malicious attacks that cause failure or interruptions to the service is a top priority. But TCP is known to be highly vulnerable to blind denial of service (DoS) attacks by SYN segments.

UDP Limitations

UDP has the following shortcomings when being considered for carrying telephony signaling data:

- UDP provides an unreliable data transfer service to the application, i.e., an application using UDP cannot know whether data sent to a peer application is received or not. Moreover, even if the data is received there is no guarantee on the ordering of the data, and the reception of duplicated copies.
- Also UDP has no built-in mechanism to detect path congestion and consequently throttle back its data transmission.

As UDP cannot meet the data reliability requirements it is unsuitable for telephony signaling applications. However as UDP is message-oriented and considered a lightweight protocol with small overhead, attempts have been made to make up in the application what is lacking in UDP in order to meet the stringent timing and data reliability requirements. This may not be a good solution as the added complexity may add additional burden on the application.

SCTP Enhancements over TCP and UDP

In order to address the limitations of TCP and UDP, the Signaling Transport (SIGTRAN) working group of IETF developed SCTP. While the development of SCTP

was motivated by the transportation of the Public Switched Telephone Network (PSTN) signaling messages across the IP network, SIGTRAN ensured that the design is also a good match for other applications with similar requirements.

The design of SCTP absorbed many of the strengths of TCP, such as error detection, retransmission and window-based congestion control. Nevertheless, SCTP has incorporated many new features that are otherwise not available in TCP. Two such new capabilities are:

- The support for multi-homed hosts which allows a single SCTP association to run across multiple paths thereby providing path redundancy which enable fast failover from one path over to another with minimal interruptions to the data transfer service.

Figure 1 - SCTP Multi-homing [3]

- The support for multi-streaming which alleviates the head-of line blocking problem of TCP. This feature can be used to divide the overall flow into independent sub-flows and to enforce ordering only within the sub-flows. Thereby preventing messages from different sub-flows from blocking one another.

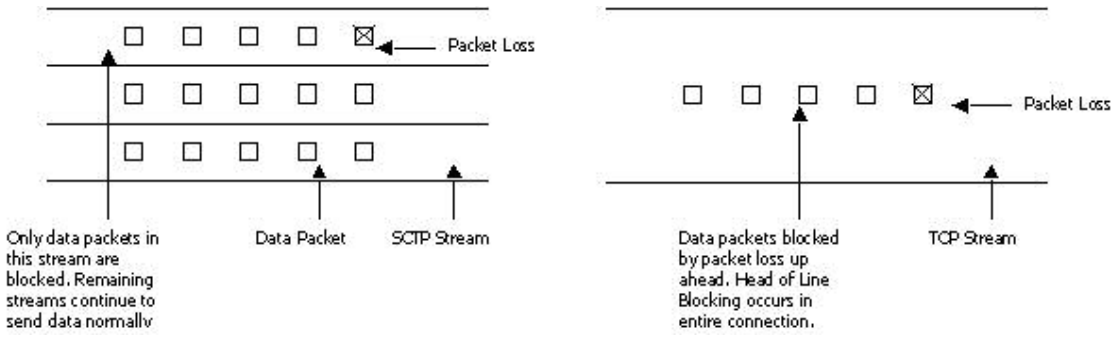


Figure 2 - SCTP Multi-streaming [3]

Besides these two major features there are other enhancements designed into SCTP. Table 1 gives a more detailed comparison between SCTP, TCP and UDP.

Protocol Features	**SCTP**	**TCP**	**UDP**
Full-duplex data transmission	yes	yes	yes
Connection oriented	yes	yes	no
Reliable data transfer	yes	yes	no
Partially reliable data transfer	optional	no	no
Ordered data delivery	yes	yes	no
Unordered data delivery	yes	no	no
Flow and congestion control	yes	yes	no
Explicit congestion notification support	yes	yes	no
Selective ACKs	yes	optional	no
Preservation of message boundaries	yes	no	yes
Path maximum transmission unit discovery	yes	yes	no

Application data fragmentation/bundling	yes	yes	no
Multi-streaming	yes	no	no
Multi-homing	yes	no	no
Protection against SYN flooding attack	yes	no	n/a
Reachability check	yes	yes	no
Half-closed connections	no	yes	n/a

Table I - Comparison between SCTP, TCP and UDP [3]

SCTP Packet Format

An SCTP packet is made up of an SCP common header of 12 bytes and building blocks called chunks [1].

The fields within the common header provide the following basic functions:

- Source and Destination Ports – These along with the IP addresses in the IP header help to uniquely identify the association to which an SCTP packet belongs.
- Verification Tag – This value helps to ensure that a particular packet belongs to the current incarnation of an association and provides protection against a blind attacker injecting data into an existing association.
- Checksum – This value helps to ensure the data integrity of the entire packet.

The building blocks or chunks constitute the rest of an SCTP packet. Chunks provide SCTP with the basic structure needed to carry information. They are classified into two types: control chunk and data chunk. SCTP control chunks transfer information needed

for association functionality, while data chunks carry application layer data. The current specification allows 256 different chunk types of which only 16 are currently defined in the base SCTP for association establishment, termination, data acknowledgement, destination failure detection, explicit congestion notification and error detecting, leaving an additional 240 chunk types that may be defined in the future by the IETF.

Each chunk has a chunk header that consists of three mandatory fields

- Chunk Type – This 8-bit field represents the type of chunk that is present. i.e., either data chunk or a type of control chunk.
- Chunk Flags- This 8-bit wide field defines any special flags that the chunk type may wish to use.
- Chunk Length – This 16-bit field indicates the length of the entire chunk (including chunk type and flags fields) in bytes.

SCTP has the flexibility to concatenate different chunk types into one data packet. The only restriction is that the packet size cannot exceed the path's Maximum Transmission Unit (MTU) size.

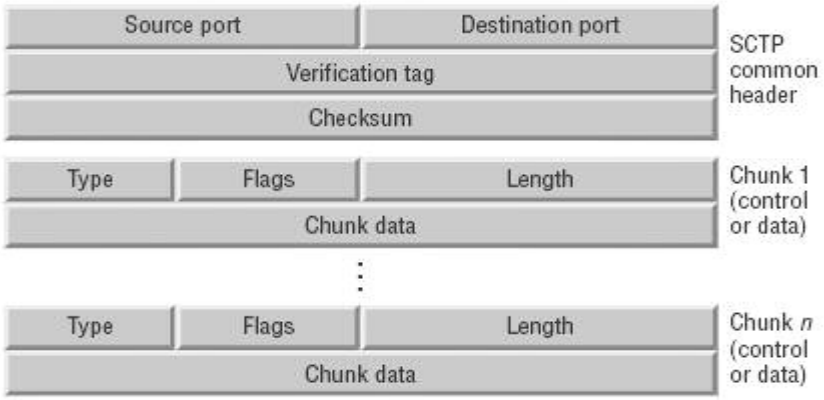


Figure 3 - SCTP Packet Format

The DATA chunk

The DATA chunk is the container for the user data transferred in SCTP. Figure – shows the format of the chunk [1]. The fields of the DATA chunk are described below:

- Chunk type – For the DATA chunk this field is set to 0x00
- A chunk flag – Out of the 8-bit length of this field, the lower 3-bits are used by the DATA chunk and are named the U, B and E bits. The upper 5-bits are reserved for future use. The U bit is used for ordered/unordered delivery options and the B and E bits are used to indicate the first and last part a fragmented user message.
- Chunk length – This field denotes the length of the user data. As a DATA chunk is required to have at least one byte of user data, this field should have a value equal to or greater than 17.
- TSN – This field represents the transmission sequence number for each data chunk. The TSN is used by both the sender and receiver to ensure that the chunk arrives at the destination and is also used to keep track of missing data chunks when a message is fragmented.
- Stream Identifier – This field indicates the stream number to which a data chunk belongs.
- Stream sequence number – This field helps to maintain message order within one stream. Stream sequence number remains the same for all the DATA chunks of a fragmented user message.
- Payload protocol identifier – This field is used by network monitors and packet filters for screening and viewing data.

- User data - This is the payload data. It is of variable length, up to the PMTU of the network for a particular destination. If the user message is larger than the PMTU, the sender fragments the message into multiple smaller parts and sends each part in a separate DATA chunk.

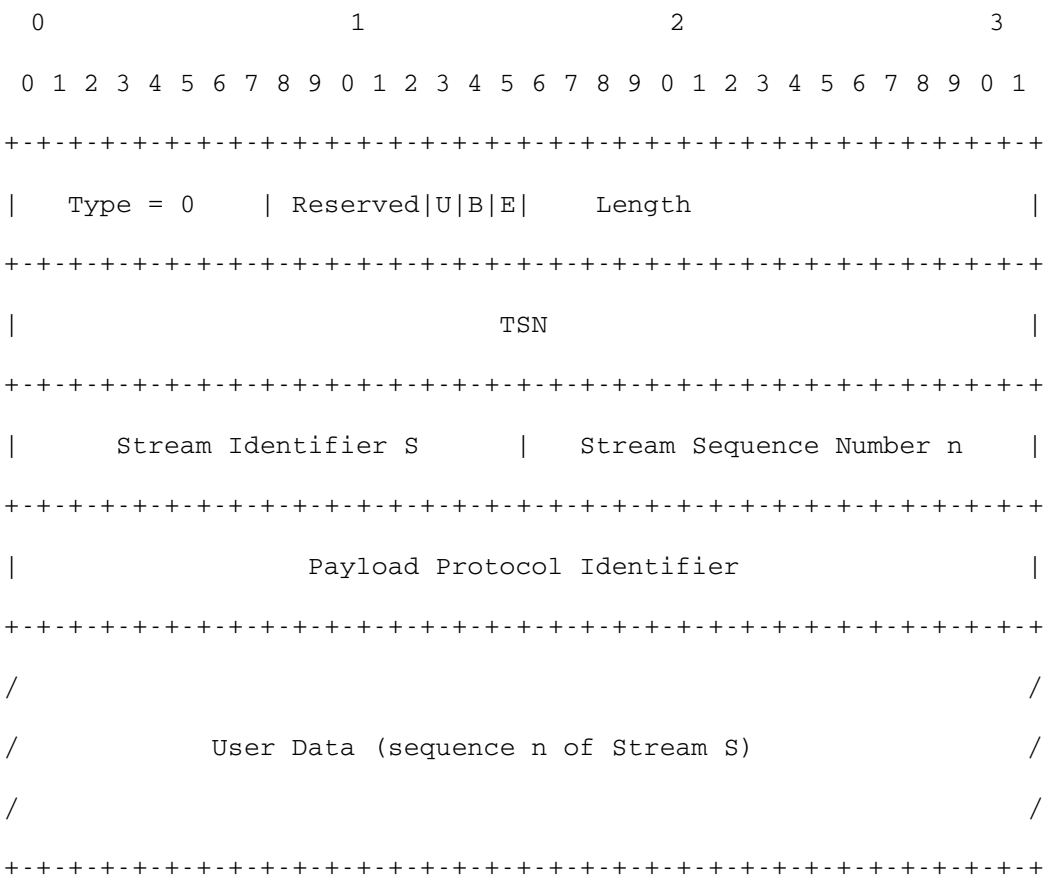


Figure 4 - SCTP Data Chunk Format

SCTP Association Phases

As a connection-oriented protocol, an SCTP association has three phases; association establishment, data transfer and association shutdown.

Association Establishment

SCTP, like TCP is a connection-oriented protocol. Therefore, setting up of an SCTP association between the two endpoints is the process that will always take place before any data can be exchanged between the two peers. This process involves the exchange of four SCTP packets between the endpoints. The exchange is robust enough to detect the classic TCP-type SYN flooding DOS attack.

The overhead of passing four SCTP packets may seem like a lot when compared to TCP's three-way handshake, but two of the of the SCTP packets can be piggy-backed with other types of information, such as user data. This helps minimize the delay burden for the application without compromising the improved security.

Figure depicts the typical four-way handshake between the two endpoints A and B. The process is detailed below in four steps.

1. When the application at host A has data to be transmitted to host B, the SCTP stack at A formulates an INIT chunk to send off to B and starts an INIT timer.
2. When host B receives the INIT chunk, it responds with an INIT-ACK chunk, without allocating any memory to maintain state for the requested association. This is unlike TCP, which is forced to maintain state at this point, making it highly susceptible to a blind SYN attack. Also host B embeds a cookie in the INIT-ACK chunk which contains information verifiable only by B regarding the legitimacy of host A.

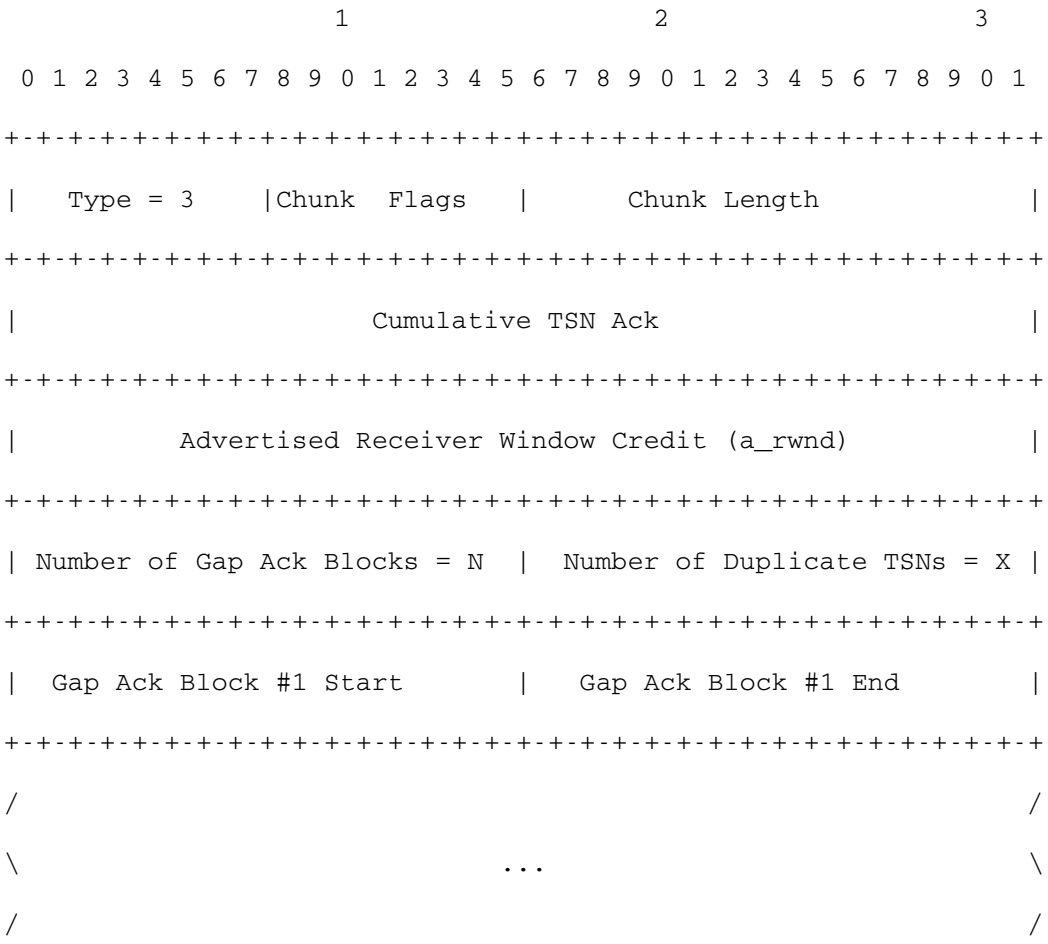
3. When host A receives the INIT-ACK, it stops the INIT timer, replies with a COOKIE-ECHO chunk; which essentially echoes the cookie that host B sent and starts the COOKIE timer.
4. On receiving the COOKIE-ECHO chunk, host B checks the validity of the cookie and on successful validation allocates resources and sends a COOKIE-ACK chunk to host A. Upon receiving the COOKIE-ACK, host A stops the COOKIE timer and an association is established between the two endpoints.

Data Transfer

Once the association is in the ESTABLISHED state normal data transfer can start. User messages passed from the application to the SCTP layer for transmission will first be converted into SCTP DATA chunks. This conversion process can take two different courses, depending on the size of the user message. If the user message is small enough, the conversion is simply to add a DATA chunk header to the message, forming a single DATA chunk. If the user message is bigger than the Path Maximum Transmission Unit (PMTU), it is fragmented into several small parts and then each part is converted into a separate DATA chunk. Each data chunk is also assigned the stream identifier of the outbound stream to which the message belongs, a stream sequence number to maintain the order of messages within each stream and a TSN to permit the receiving peer to acknowledge its receipt and detect duplicate deliveries. This data chunks is then bundled together with other data chunks or control chunks and passed to the IP layer for transmission over the network. After arriving at the SCTP receiver, the SCTP packets will be unbundled into DATA chunks as well as control chunks and delivered to the

application. Finally, the receiver will acknowledge the reception of all the DATA chunks by sending back SACKs to the message sender.

SCTP maintains reliability through SACKs, retransmissions and the CRC-32 checksum. SCTP acks carry cumulative (CumAck) and selective (GapAck) information. The CumAck indicates the last TSN successfully received in sequence. The GapAck blocks indicate TSN's received out of order beyond the CumAck. Figure – shows the packet format for an SCTP SACK control chunk. Network congestion prevention and packet loss detection and recovery is handled by flow and congestion control and time-out and fast retransmit mechanisms.



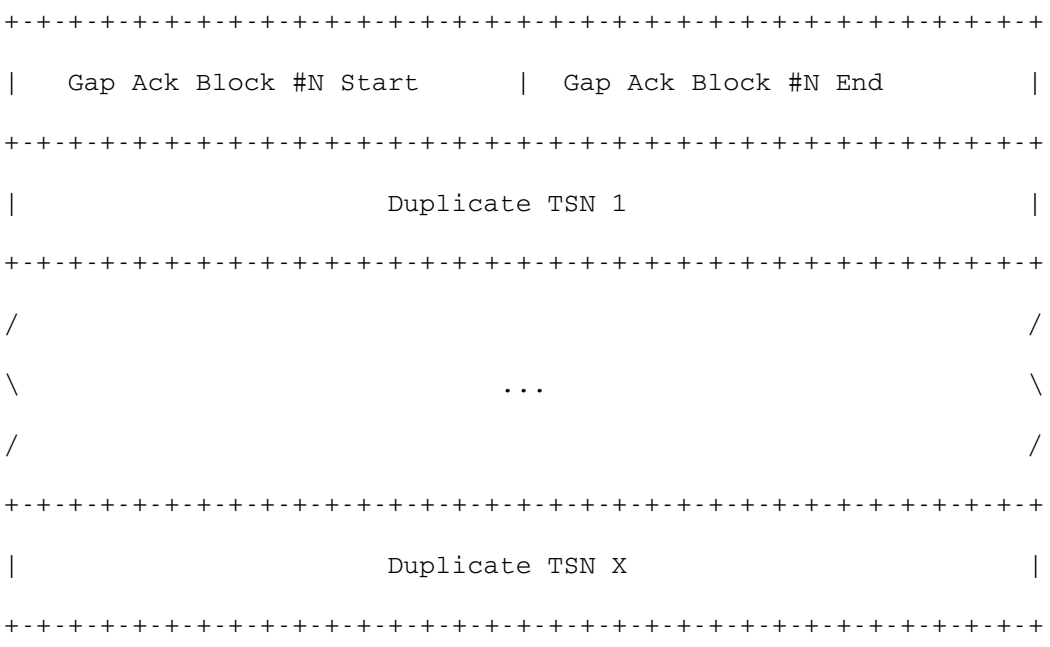


Figure 5 - SCTP SACK Chunk Format [1]

Association Shutdown

Any reliable protocol needs a methodology not only to bring up a communication but also to bring that communication to a close. SCTP has two methods for bringing an association to a CLOSED state, namely graceful and abortive shutdown.

In the graceful shutdown, each peer assures that all data in the queue is delivered and acknowledged. After this the association enters the CLOSED state. This is pretty much similar to TCP's down, with one major exception: while TCP supports a "half-closed" state, where one end is CLOSED and not accepting new data to transfer, and the other end is still open and able to send new data; SCTP does not. Figure illustrates SCTP's three-message handshake to gracefully close down an association.

The abortive shutdown is an unreliable best-effort attempt to tell a peer that the association is going away. This is simply carried out by the endpoint sending an ABORT chunk to its peer, removing its TCB and transitioning into the CLOSED state. The peer on receiving the ABORT chunk follows suit.

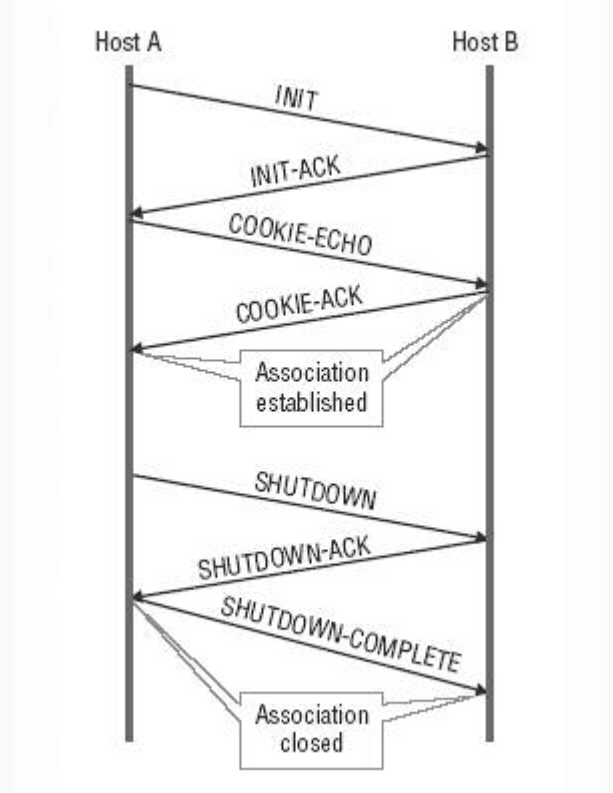


Figure 6 - SCTP Association Establishment and Shutdown [3]

Partially Reliable SCTP

Many Internet applications such as real-time multimedia traffic (e.g. VoIP), transmission of video and other time sensitive material, requires partial reliable transport of messages. A partially reliable transport service is defined as one which allows the user to specify the rules governing how persistent the transport service should be in attempting to transmit/retransmit the message to the receiver.

A new IETF draft RFC 3758 specifies the use of SCTP as a partial reliable transport protocol which can carry both traffic requiring partial reliability as well as traffic requiring full reliability [4]. PR-SCTP allows an SCTP sender to assign different levels of reliability to messages so that lost messages may be retransmitted only until the reliability threshold (or lifetime parameter) for that message is reached. If the reliability threshold is reached for unsent/un-ACKed messages, the sender abandons those messages and notifies the receiver to do the same.

In order to provide partial reliable service over an existing SCTP association, two new parameters have been added to the original protocol.

- A single new chunk type, FORWARD TSN that indicates that the receiver should move its cumulative ack point forward, possibly skipping past one or more DATA chunks that may not yet have been received and/or acknowledged.
- A single new parameter in the INIT/INIT-ACK chunk types that indicates whether the endpoint supports the new partial reliability extension.

CHAPTER II

REVIEW OF LITERATURE

In [6] stream priorities are introduced as a method of decreasing delays of important data during periods of low bandwidth availability. On-line multimedia experiences are often bandwidth intensive. They require high throughput connections to be comfortable for end users. While the number of broadband subscribers grows daily, the majority of Internet users still rely on slower dial-up connections, which are often insufficient for comfortable viewing of multimedia data. In addition, many pocket devices, such as mobile phones and Personal Digital Assistants (PDAs), now offer web browsing and streaming video over low bandwidth, wireless connections. The maximum throughput achieved by these devices fluctuates depending on signal strength. Moreover current end users want to request various types of data from application servers. Therefore, the servers must provide a way to transmit multiple data-types in parallel, and must effectively respond to periods of insufficient bandwidth

Traditionally, transmitting different types of data in parallel between endpoints relied on one of three approaches. In all three situations, Host A would like to send three types of data, labeled Data 1 through Data 3 to Host B. In the first approach, Host A opens three TCP connections to Host B – one connection per data type.

While this approach provides logical separation of data based on type, multiple connections defeat TCP-friendly congestion control by allowing an application to gain an unfair portion of available bandwidth at the expense of other data flows in the network.

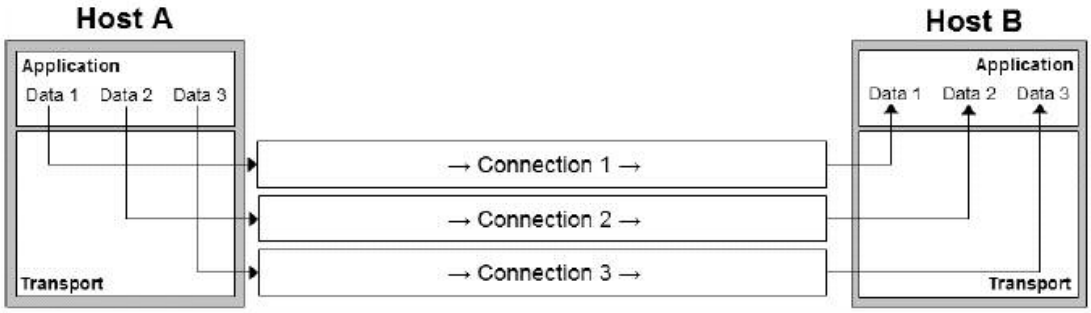


Figure 7 - Data Transmission using Multiple TCP Connections [6]

In the second approach, Host A multiplexes and demultiplexes the three types of data over a single connection. Applications using this approach maintain TCP-friendly congestion control; however, this approach increases complexity for the application programmer, since the application itself must handle the complicated task of efficiently and fairly managing data transmission scheduling.

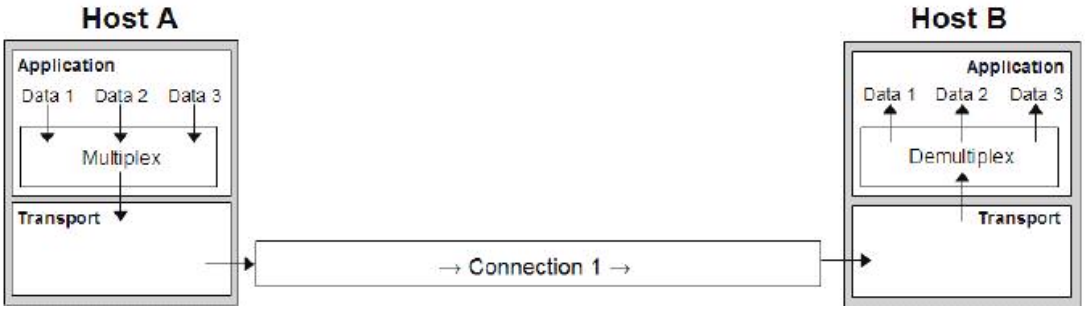


Figure 8 - Data Transmission using TCP with Application level Multiplexing/Demultiplexing [6]

The third approach has a multimedia application using UDP. This approach closely resembles the second approach; however application programmers must supply their own reliability service as well as their own multiplexing/demultiplexing due to UDP's unreliable, connectionless service.

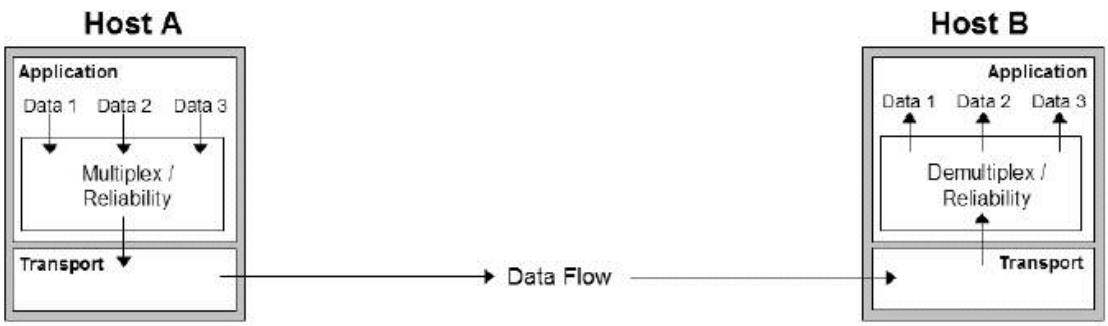


Figure 9 - Data Transmission using UDP [6]

With the introduction of the SCTP's concept of streams, applications are presented with a new transport layer solution to transmitting multiple types of data. This new approach combines advantages of multiple end-to-end connections and application multiplexing/demultiplexing.

SCTP's Multi-streaming Feature in Detail

In an SCTP association a stream is a unidirectional logical channel established from one endpoint to another. Multi-streaming aims to separate flows of logically different data within a single association. This logical separation of data using streams allows the transport layer to take up the responsibility of managing the flows, otherwise performed by the application layer. Within each stream messages are delivered in sequence, except for those messages that specify an unordered delivery service. During association setup SCTP end points negotiate the number of streams required at each end. Figure – shows a

multi-streamed association between hosts A and B. During this example's association setup, host A requested three outbound streams to host B (numbered 0 to 2) and host B requested only one outbound stream to host A (numbered 0).

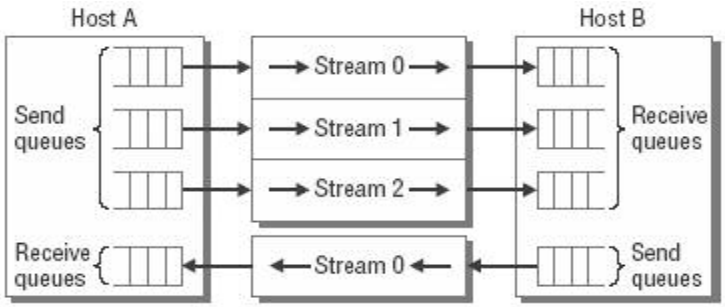


Figure 10 - A Multi-streamed Association [3]

To preserve the data order and reliability for each data chunk, within streams, SCTP uses stream sequence numbers (SSNs). The socket API extension for SCTP provide data structures and socket calls through which application can indicate or determine the stream number on which it intends to send or receive data. Between streams, no data order is preserved. This approach avoids TCP's head-of-line blocking problem, in which successfully transmitted segments must wait in the receiver's queue until a TCP sending end point retransmits any previously lost segments. This blockage delays delivery of received data to the receiving application, which is unnecessary and sometimes unacceptable in signaling and some multimedia applications. In case of SCTP, if data on stream 1 is lost, only stream 1 is blocked at the receiver while awaiting retransmissions. The receiving end point can immediately deliver data arriving on other streams to the application.

Adding Priorities to Streams

In order to transmit multiple types of data using SCTP multi-streaming, a scheduling algorithm that avoids stream starvation must be used [5]. Two of the most commonly used algorithms are first-come-first-serve and round-robin scheduling.

In the first-come-first-serve scheduling algorithm a Host A transmits each data chunk in the order in which it is received from the application, irrespective of the streams. In the round-robin scheduling approach a Host A would select a data chunk from each of the streams for transmission, for the life of the association.

These algorithms would be highly efficient during periods of high bandwidth availability. But during periods of poor network conditions, delays would be introduced between endpoints across all streams. Thus in order to alleviate this situation each stream must be assigned a priority by the application to specify the relative importance of the data carried in that stream. This will enable transmission of critical data to gain precedence during periods of low-quality of service. The SCTP strict-stream priority scheme can be defined as [6]:

"Data on stream i always have greater priority in relation to

data on stream j, where i < j"

This can be implemented in the SCTP Sockets API as

```
sctp_enablepriority ()
{
      for (i = 0; i < num_streams; i++)
      {
            streams[i].priority = i;
      }
```

```
}
```

After assigning priorities, [6] then introduces a fixed priority based scheduling algorithm which transmits data based on stream priorities. The running time of this algorithm depends on number of streams and amount of data to be transmitted. As today's multimedia applications over the internet involve the transmission of both reliable as well as partially reliable data such as audio, video and other time-sensitive data, the static priority algorithm would be inadequate for the stringent lifetime requirements of PR data. Thus a new scheme where the priorities of PR data dynamically change to accommodate for the time sensitive PR data is proposed in the next-section.

CHAPTER III

PROPOSED WORK

Dynamic Priority Streams

Previous work [6] introduces a static priority scheduling algorithm suitable for only reliable data. Several applications may benefit from this static-stream priority scheme. One such widely used general purpose application is Instant Messaging (IM). An IM software allows a user to connect to the messaging service and communicate with other users connected to the same service. Current instant messaging clients also support communication of various types of data such as voice, video and file transfers between two users. Among these data types audio and video can be grouped as requiring partial reliability. In other words this data type is time sensitive and thus if not delivered to the receiver within certain duration, set by the lifetime parameter, would become irrelevant. The static priority scheme if used for PR data would time-out most of the data and result in packet drops. In order to alleviate this problem, we propose a scheduling algorithm which dynamically changes priorities based on the timed-reliability factor of the PR data. The algorithm is presented below:

```
while (space exists in SCTP packet) {
      while (PR-stream has data to transmit AND PR-stream's lifetime
                                                        approaching) {
            swap PR data's priority with reliable data's;
```

```
        }
        for (j = 0; j < num of streams; j++) {
                if (current highest priority has data to transfer)
                        send all the data;
                else
                        move on to next priority;
        }
        if (priorities were swapped)
                revert priorities;
}
```

Before this algorithm is called, the priorities of the various streams are set by using the strict priority algorithm. Also checks for space in the congestion and receiver windows should be made. This algorithm checks if the lifetime of the PR data is approaching within the next round-trip time, while there is still enough space in the SCTP message. If yes, then the priority of the PR data is swapped with that of the higher priority data. If not then the packet is dropped and the sender sends a Forward-TSN informing the receiver to move its cumulative-ack forward. To prevent indefinite postponement of the reliable data, the algorithm also makes sure that the data is transmitted with acceptable delay. Based on the current priorities, the data for the highest priority stream is added to the SCTP message until space is available. If the highest priority has no more data to send then the next highest priority gets a turn at transmission. This algorithm depends on the number of streams between the end-points, the size of each stream's packet and the rate at which the application generates the data.

CHAPTER IV

SIMULATIONS AND FINDINGS

The dynamic priority algorithm proposed was applied to a simulation of an instant messaging scenario by using the popular SCTP module for ns-2, developed by the Protocol Engineering Lab at the University of Delaware [11]. Our simulation consisted of an SCTP association with 2 streams in one direction established between two nodes marked as sender and receiver.

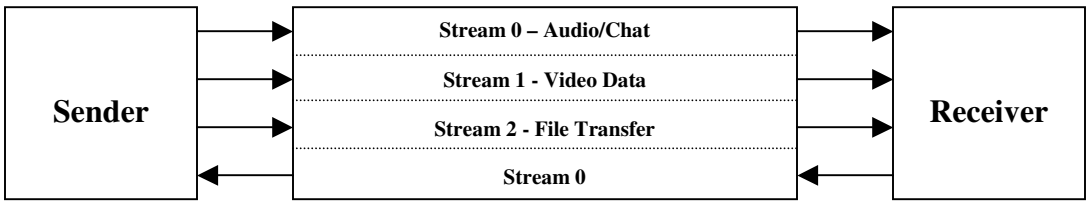


Figure 11 - An Instant Messaging Session

The sender's stream0 carried packets of 200bytes generated at 40packets/second representing a high priority, partially-reliable data, such as audio. The sender's stream1 carried partially reliable packets of 1000bytes generated at 15packets/second with a timed-reliability factor of 3 seconds, representing the video conferencing data through a web-camera. We then compared the performance of our algorithm against the static priority SCTP algorithm, over simulated dial-up (128Kbps)

links with a propagation delay of 250ms. Our performance criteria was based on the number of partially-reliable video packets dropped and the ratio of the number of packets of each stream transmitted. Our simulations did not consider any network losses due to congestion. For analysis let us consider the conditions given below. Let:

$R_{available}$ be the rate at which SCTP can send data to the receiver

R_0 be the rate of data submitted by the application for transmission over stream0

R_1 be the rate of data submitted by the application for transmission over stream1

R_2 be the rate of data submitted by the application for transmission over stream2

Condition1 - $R_0 + R_1 + R_2 < R_{available}$: During this time since enough bandwidth is available there is no queuing and data is transmitted without any packet drops.

Condition2 – $R_0 + R_1 + R_2 > R_{available}$: During this time queuing would definitely occur but the number of packet drops for the partially-reliable data using dynamic priorities should be lesser than the number of packet drops for the static prioritySCTP scheme.

Figure of Merit

Figure of merit (FOM) is defined as a numerical quantity based on one or more characteristics of a system that represents a measure of efficiency or effectiveness. To compare the performance of the dynamic priority algorithm with that of the static priority algorithm we compare the min ratios of the number of packets sent in each stream to the number of packets generated by each stream. The max of the values helps us prove the fairness of the scheme.

For static priority:

$$\min\left[\frac{\text{no. of stream0 packets sent}}{\text{no. of stream0 packets generated}}\ldots,\frac{\text{no. of stream n packets sent}}{\text{no. of stream n packets generated}}\right] \rightarrow [1]$$

For dynamic priorities:

$$\min\left[\frac{\text{no. of stream0 packets sent}}{\text{no. of stream0 packets generated}}\ldots,\frac{\text{no. of stream n packets sent}}{\text{no. of stream n packets generated}}\right] \rightarrow [2]$$

Fair scheme = algorithm with max ([1], [2])

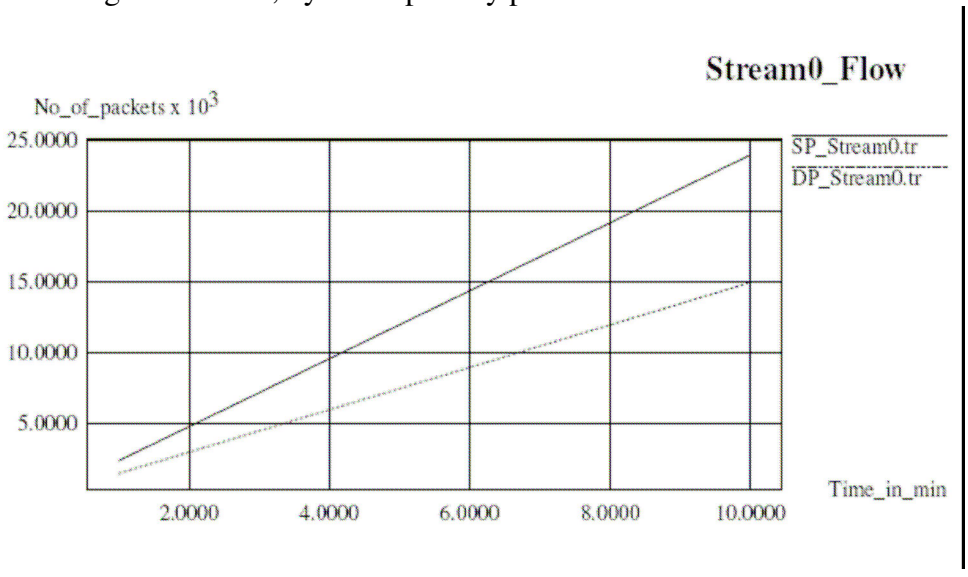
Simulations were performed with different delay rates. The graphs obtained are depicted below:

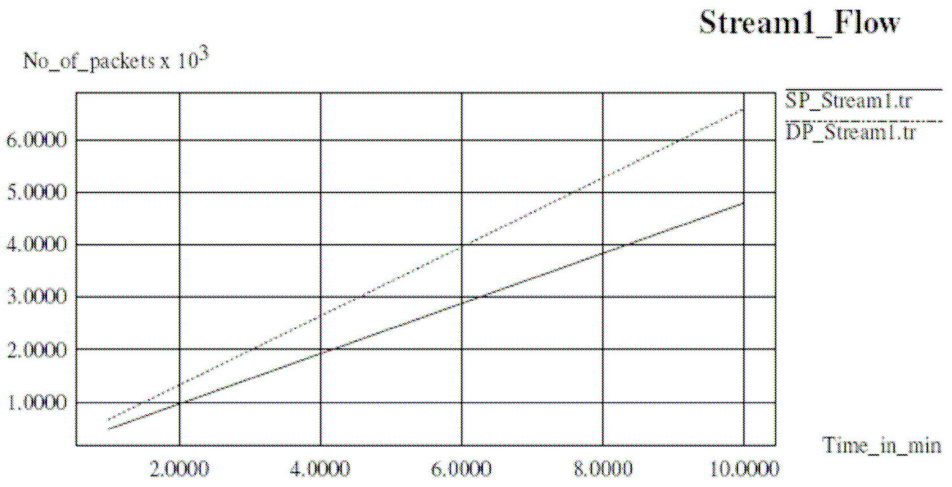
Scenario1: Bandwidth = 128kbps

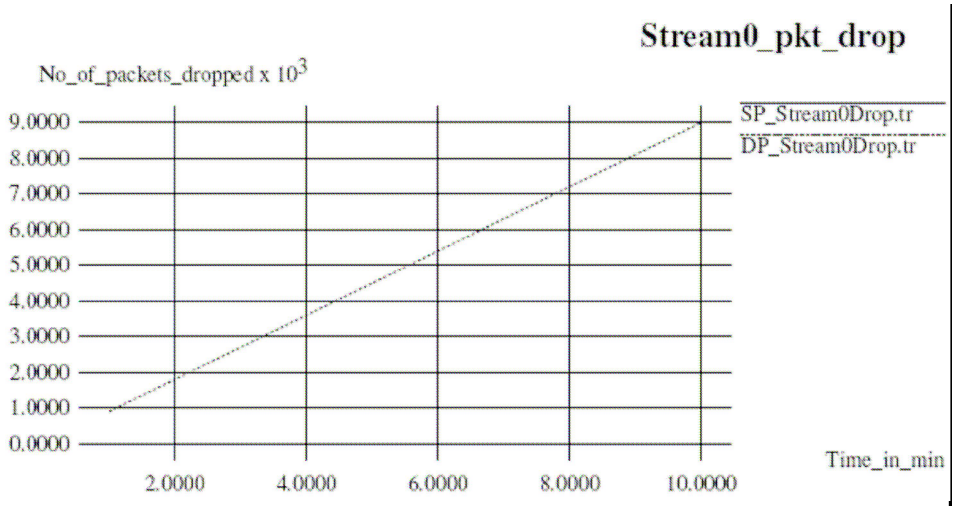
Stream	Delay	Bit rate	Total pkts in 10mins	No.of pkts sent in 10mins (SP)	No. of pkts dropped in 10mins (SP)	No.of pkts sent in 10mins (DP)	No.of pkts dropped in 10min(DP)
Stream0	200ms	64Kbps	24000	24000	0	14863	9137
Stream1	3sec	120Kbps	9000	4800	4200	6593	2407

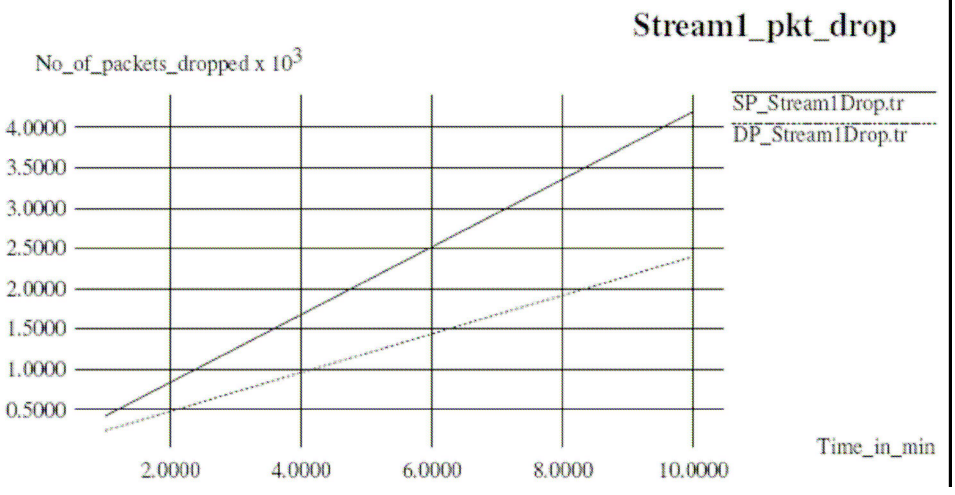
Table II - Simulation Scenario 1

Min. ratio for static priority = 0.53
Min. ratio for dynamic priority = 0.62
Based on our figure of merit, dynamic priority performs better.




Stream1_Flow
No_of_packets x 10^3
6.0000
5.0000
4.0000
3.0000
2.0000
1.0000
2.0000
4.0000
6.0000
8.0000
10.0000
Time_in_min
SP_Stream1.tr
DP_Stream1.tr


Stream0_pkt_drop
No_of_packets_dropped x 10^3
9.0000
8.0000
7.0000
6.0000
5.0000
4.0000
3.0000
2.0000
1.0000
0.0000
2.0000
4.0000
6.0000
8.0000
10.0000
Time_in_min
SP_Stream0Drop.tr
DP_Stream0Drop.tr

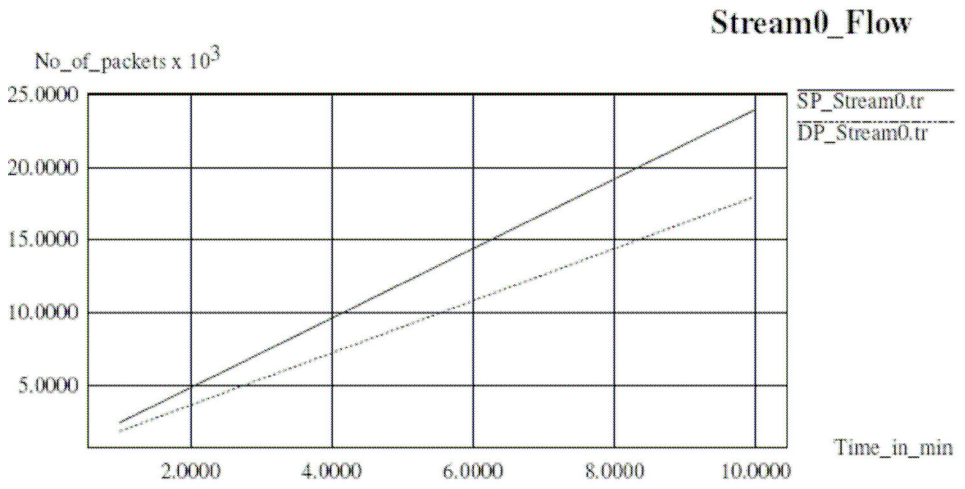

Stream1_pkt_drop
No_of_packets_dropped x 10^3
4.0000
3.5000
3.0000
2.5000
2.0000
1.5000
1.0000
0.5000
2.0000
4.0000
6.0000
8.0000
10.0000
Time_in_min
SP_Stream1Drop.tr
DP_Stream1Drop.tr

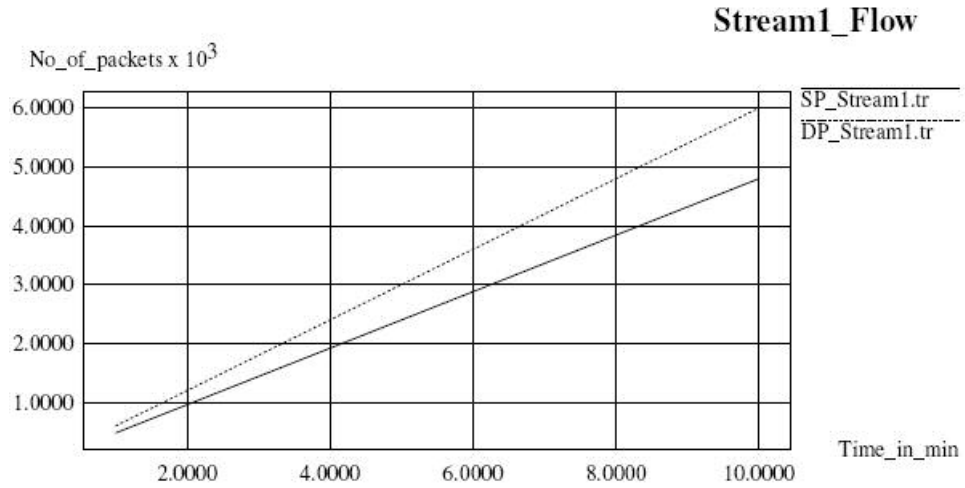
Scenario 2: Bandwidth = 128kbps

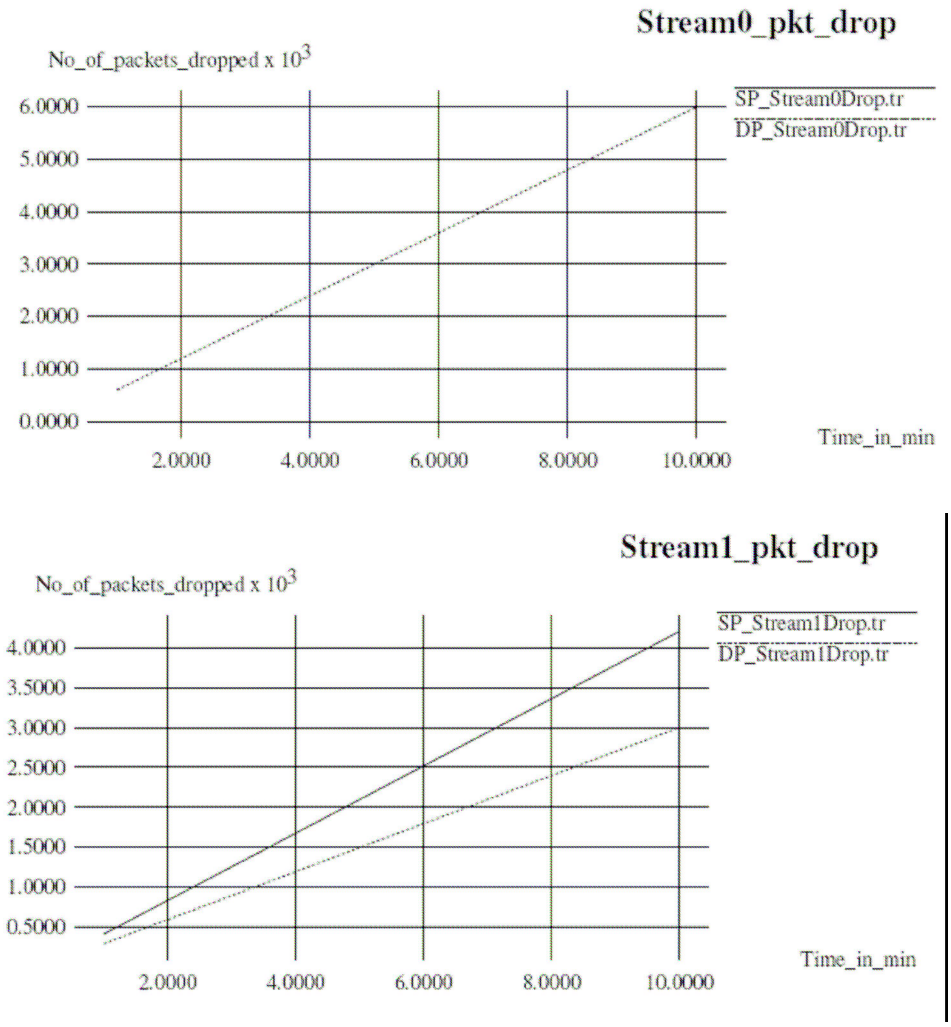
Stream	Delay	Bit rate	Total pkts in 10mins	No.of pkts sent in 10mins (SP)	No. of pkts dropped in 10mins (SP)	No.of pkts sent in 10mins (DP)	No.of pkts dropped in 10min(DP)
Stream0	250ms	64Kbps	24000	24000	0	17965	6035
Stream1	3sec	120Kbps	9000	4800	4200	5943	3057

Table III - Simulation Scenario 2

Min. ratio for static priority = 0.53
Min. ratio for dynamic priority = 0.66
Based on our figure of merit, dynamic priority performs better.





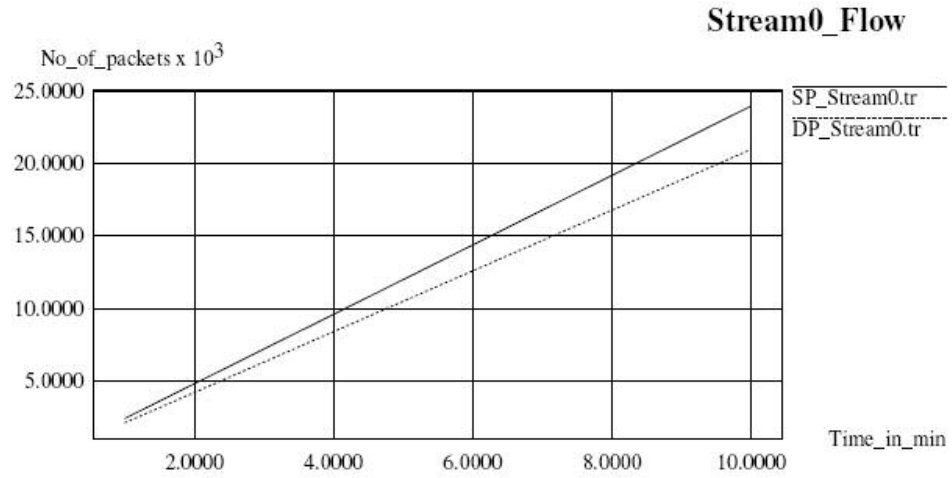


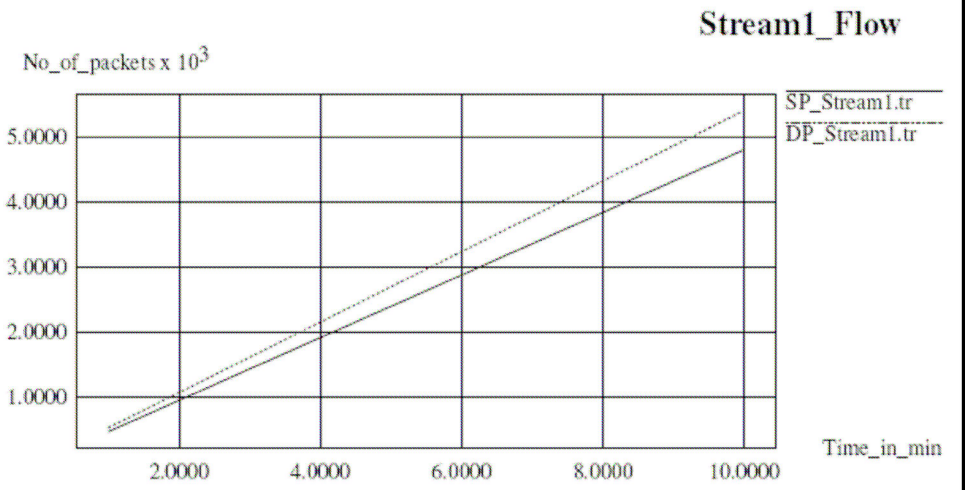
Scenario 3: Bandwidth = 128kbps

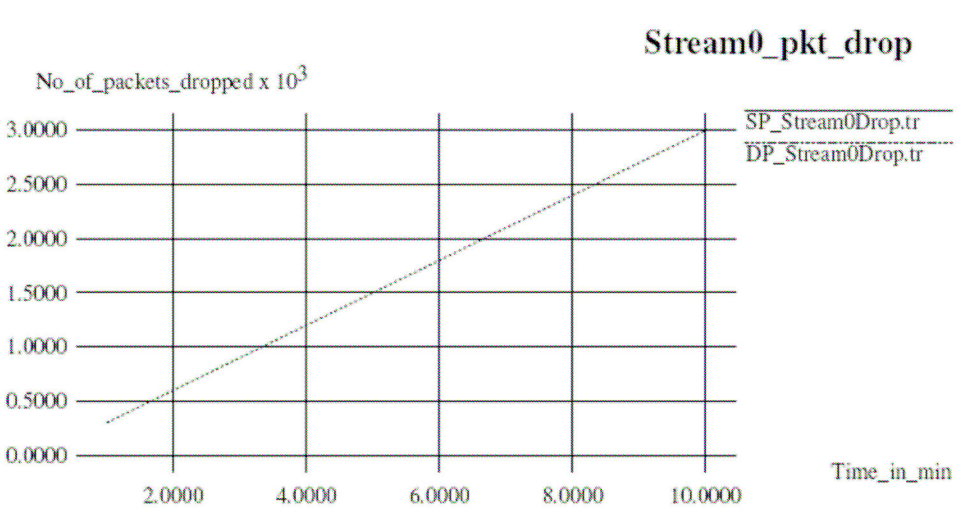
Stream	Delay	Bit rate	Total pkts in 10mins	No.of pkts sent in 10mins (SP)	No. of pkts dropped in 10mins (SP)	No.of pkts sent in 10mins (DP)	No.of pkts dropped in 10min(DP)
Stream0	300ms	64Kbps	24000	24000	0	20989	3011
Stream1	3sec	120Kbps	9000	4800	4200	5419	3581

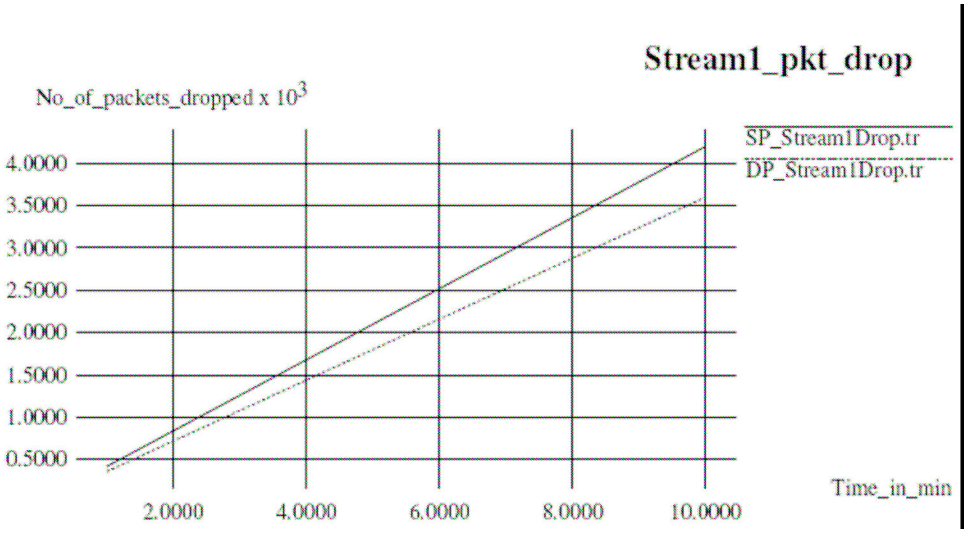
Table IV - Simulation Scenario 3

Min. ratio for static priority = 0.53
Min. ratio for dynamic priority = 0.60
Based on our figure of merit, dynamic priority performs better.


Stream0_Flow
No_of_packets x 10^3
25.0000
20.0000
15.0000
10.0000
5.0000
2.0000
4.0000
6.0000
8.0000
10.0000
Time_in_min
SP_Stream0.tr
DP_Stream0.tr


Stream1_Flow
No_of_packets x 10^3
5.0000
4.0000
3.0000
2.0000
1.0000
2.0000
4.0000
6.0000
8.0000
10.0000
Time_in_min
SP_Stream1.tr
DP_Stream1.tr


Stream0_pkt_drop
No_of_packets_dropped x 10^3
3.0000
2.5000
2.0000
1.5000
1.0000
0.5000
0.0000
2.0000
4.0000
6.0000
8.0000
10.0000
Time_in_min
SP_Stream0Drop.tr
DP_Stream0Drop.tr



Thus through the various simulation scenarios we are able to show that the dynamic priority algorithm performs better for streams transmitting partial-reliable data.

CHAPTER V

CONCLUSION AND FUTURE WORK

Initially, we have explained the basic architecture and working of the Stream Control Transmission Protocol. We have also explained SCTP's new provision for supporting partially reliable data. We have then summarized the drawbacks of the three traditional approaches used for transmitting different types of data in parallel between two end-points and have then detailed how SCTP's multi-streaming feature would help overcome these drawbacks. We have briefly explained the previous work [6] which uses a static priority schemes for only reliable data. We have then investigated a scenario where there is both partially reliable data for transmission and have come up with a scheduling algorithm which dynamically switches the priorities of the partially reliable data based on its timed-reliability factor. Finally through simulations we have shown that during periods of low bandwidth availability, the streams are given a fair chance to transmit data based on their timed-reliability factor, unlike the static priority scheme which gives preference to the higher priority stream

In this work although the streams are prioritized, because of the sharing of congestion information among streams, the stream receiving a lower level of service from the network may experience more losses. These losses in turn influence the entire transmission and the benefits of priorities among streams are thus lost. Future work

should investigate ways in which streams are treated differentially by the network based on their priorities. And should also explore the various other figures of merits for the dynamic priority algorithm in scenarios where the streams are of equal priority.

REFERENCES

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

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

[3] A. Caro Jr, J. Iyengar, P. Amer, S. Ladha, G. Heinz II, K. Shah. "SCTP: A Proposed Standard for Robust Internet Data Transport", 2003.

[4] R. Stewart, M. Ramalho, Q. Xie, M. Tuexen, P. Conrad. "RFC – 3758 Stream Control Transmission Protocol Partial Reliability Extension", May 2004

[5] L. Coene. "RFC – 3257 Stream Control Tranmission Applicability Statement", April 2002.

[6] G. Heinz II. "Priorities in Stream Transmission Control Protocol Multi-streaming". 2003.

[7] S. Ladha, P. Amer. "Improving File Transfers using SCTP Multi-streaming".

[8] S. Santani, J. Iyengar, M. Fecko. "SCTP Multi-streaming: Preferential Treatment among Streams" 2003

[9] A. Balk, M. Sigler, M. Gerla, M. Sanadidi. "Investigation of MPEG – 4 Video Streaming over SCTP".

[10] P. Conrad, G. Heinz, A. Caro Jr, P. Amer, J. Fiore. "SCTP in Battlefield Networks"

[11] A.Caro, J. Iyengar. ns-2 SCTP module. http://pel.cis.udel.edu.

VITA

Tasneem Yunus Kanpurwala

Candidate for the Degree of

Master of Science

Thesis: SCTP MULTI-STREAMING: STUDY OF TRANSMISSION OF PARTIALLY RELIABLE AND RELIABLE DATA USING DYNAMIC STREAM PRIORITIES

Major Field: Computer Science

Biographical:

Education: Received Bachelor of Engineering degree in Computer Science from University of Madras, Chennai, India in May 1998. Completed the requirements for Master of Science degree with major in Computer Science at Oklahoma State University in May, 2006

Experience: Employed by Oklahoma State University, College of Engineering Architecture and Technology as a Graduate Assistant, 2004 to Present

Professional Memberships: Association of Computing Machineries

Name: Tasneem Yunus Kanpurwala Date of Degree: May, 2006

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: SCTP MULTI-STREAMING: STUDY OF TRANSMISSION OF PARTIALLY RELIABLE AND RELIABLE DATA USING DYNAMIC STREAM PRIORITIES

Pages in Study: 34 Candidate for the Degree of Master of Science

Major Field: Computer Science

Scope and Method of Study: On-line multimedia experiences are often bandwidth-intensive. The majority of internet users still rely on dial-up modem or DSL connections which are often inadequate for comfortable viewing of real-time multimedia data. The application end-points must then provide a way to transmit multiple data-types in parallel and must effectively respond to periods of insufficient bandwidth. Using multiple TCP connections for transferring different data, defeats TCP-friendly congestion control. With the introduction of multi-streaming in the Stream Control Transmission Protocol, applications such as instant messaging are now presented with a new transport layer mechanism which is markedly superior to TCP and UDP for multimedia transmissions. Multi-streaming provides an aggregation mechanism with logical demarcation of data for transferring different objects belonging to the same application. Traditional SCTP makes use of round-robin or FIFO algorithm for scheduling the various streams. But during periods of low bandwidth availability, transmission of critical data should gain precedence over non-critical data. Previous work proposes a static priority scheme suitable for only reliable data. Our work investigates using dynamic priorities with both partially reliable (PR) and reliable data.

Findings and Conclusions: In this thesis we propose a scheduling algorithm to dynamically prioritize the streams depending on the timed-reliability factor of the PR-data. Through ns-2 simulations we compare the dynamic priority SCTP with static priority SCTP using an application with two streams, one sending a low bit-rate, short delay, partially reliable data and the second sending a high bit-rate partially reliable data. During periods of low-bandwidth availability we demonstrate that with this scheme, both the streams are given a fair chance to transmit data based on their timed-reliability factor, unlike the static priority scheme which gives preference to the higher priority stream.

ADVISOR'S APPROVAL: _____

Dr. Venkatesh Sarangan