Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Computer Communications 27 (2004) 1012–1024

computer communications

www.elsevier.com/locate/comcom

LS-SCTP: a bandwidth aggregation technique for stream control transmission protocol☆

Ahmed Abd El Al*, Tarek Saadawi, Myung Lee

Department of Electrical Engineering, City College and Graduate Center of City University of New York, New York, NY 10031, USA

Abstract

Stream Control Transmission Protocol (SCTP) specifications utilize the multiple paths capabilities between the sender and receiver for retransmission of lost data chunks and as a backup in case of primary path failure. Under normal conditions, all data chunks are sent on the primary path chosen by the SCTP user during the transport connection initiation. In this paper, we address in detail various aspects related to extending and engineering SCTP in order to utilize the available paths for simultaneous transmission of data chunks, while maintaining the SCTP congestion control on each path so as to ensure fair integration with other traffic in the network. The extended SCTP, referred to as Load-Sharing SCTP (LS-SCTP), is able to aggregate the bandwidth of all the active transmission paths between the communicating endpoints. LS-SCTP monitors the paths, and accordingly it chooses the paths that are suitable for load sharing. LS-SCTP retransmission mechanism accelerates the delivery of missing data to the receiver in order to prevent stalling the transport connection while waiting for missing data chunks. Simulation results show that LS-SCTP is extremely beneficial for networks with limited bandwidth, high loss rate and failure prone links.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Stream control transmission protocol; Bandwidth aggregation; Multi-homing

1. Introduction

For many years SS7 has been the dominant bearer of signaling traffic for telecommunication networks, but recently many proprietary solutions for transporting signaling traffic over IP have appeared. This approach promises tighter integration with Voice over IP (VoIP) solutions and ultimately the possibility of a common core network capable to transport signaling and media traffic. Driven by industry interest and general agreement on the unsuitability of TCP and UDP for signaling transport, the IETF Signaling Transport (SIGTRAN) group was formed in 1999 to standardize a suitable transport protocol for signaling traffic over IP. Stream control transmission protocol (SCTP) was the result of this work, and it was recently published as RFC 2960 [1] by the Internet society.

SCTP is a reliable, message-oriented data transport protocol that supports multiple streams within a single transport layer connection, an 'association' in SCTP terminology, and hosts with multiple network interfaces (multi-homed hosts). These properties make SCTP more suitable for signaling transport, as well as for providing transport benefits to other applications requiring additional performance and reliability. SCTP features will be described in detail in Section 3.

SCTP support for multi-homed hosts is intended to provide communication reliability for the hosts engaged in the association. Initially, two interfaces, one at each host, are chosen to form the primary path that is used for transmission of the data units, 'data chunks' in SCTP terminology. The other interfaces, which form the secondary paths, are only used for retransmission of lost data chunks or as a backup for the primary path. This means that although these paths exist, they are only utilized for

☆ Prepared through collaborative participation in the Communications and Networks Consortium sponsored by US Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011.

* Corresponding author. Tel.: +1-212-650-7158; fax: +1-212-650-8249.
E-mail addresses: amabdal@cs.com (A. Abd El Al); saadawi@ee1s0.engr.ccny.cuny.edu (T. Saadawi); mjlee@ees1s0.engr.ccny.cuny.edu (M. Lee).

0140-3664/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2004.01.021

retransmission or for failure recovery. In this paper, we propose extending SCTP to utilize the available paths for simultaneous transmission of data chunks, i.e. load sharing, while maintaining the SCTP congestion control on each path, in order to ensure fair integration with other traffic in the network. We believe that this form of bandwidth aggregation is extremely beneficial for networks with limited bandwidth and high loss rates. In order to aggregate the available bandwidth, taking in account the differences in the characteristics of the paths, in terms of bandwidth, latency and loss rates, we propose a separation between the association congestion control and flow control. In our SCTP extension, which we refer to as Load Sharing-SCTP (LS-SCTP), the congestion control is performed on a path basis, while the flow control is on association basis. Standard SCTP does not separate between the flow and the congestion control, as both mechanisms work together on a single path. As the failure of a transmission path or the increase in the path loss rate can affect the throughput of the whole association, LS-SCTP monitors the paths, and accordingly it dynamically determines the paths that are suitable for load sharing. In addition, LS-SCTP retransmission mechanism accelerates the delivery of missing data to the receiver in order to prevent stalling the association, while the receiver is waiting for a missing data chunk. As will be shown in our performance study, in Section 5, that these features make LS-SCTP robust to the variations in the characteristics of the transmission paths.

During the association initialization, LS-SCTP allows the user to enable/disable the load sharing capability, as well as controlling the number of interfaces that can be used simultaneously for load sharing. This feature is important for battery-powered hosts, as it enables the LS-SCTP user to conserve the battery power by controlling the simultaneous use of the interfaces.

This paper is organized as follows. Section 2 provides a review for similar works in bandwidth aggregation. Section 3 presents a quick overview on SCTP features and the differences between SCTP and TCP. Section 4 describes in detail LS-SCTP design. Section 5 provides a performance study for LS-SCTP. Finally, Section 6 concludes the paper.

2. Related work

The idea of resource aggregation in order to obtain higher performance has been used in different areas in the computer and communication fields. Ref. [2] introduces a striping technique for the disk subsystem, which is now a key aspect in Redundant arrays of inexpensive disks (RAID) architectures. In Ref. [3], authors provide an overview on the use of resource aggregation in the network subsystem, and introduce an evaluation criteria to judge the benefits provided from the resource aggregation in terms of: latency and buffering requirements, skew tolerance, scalability and complexity and finally the maximum aggregate bandwidth that can be supported. Using these criteria, they examined and evaluated resource aggregation at each layer of the protocol stack.

An example of link layer bandwidth aggregation is Bellcore's (currently Telcordia) effort, for their Aurora testbed [4], to obtain the equivalent bandwidth of an STS-12c (620 Mbps) by using four STS-3cs (155 Mbps) aggregated together into a trunk group. ATM cells are stripped across the links in an order determined by the trunk control algorithm. The algorithm places idle and active cells that allows the receiver to determine the order in which the cells were placed on the trunk group. References [5] and [6] introduce Inverse Multiplexing, which is a standard application-transparent method to provide higher end-to-end bandwidth by splitting traffic across multiple physical channels, creating a single logical channel. Most inverse multiplexing implementations assume physical transport mechanisms with constant bit rates and stable channel characteristics, in terms of bandwidth, latency and loss rates, such as found in circuit switched networks. For this reason they use a Round Robin approach to assign data fragments to the available channels. Round robin data striping is simple and can provide perfect fair distribution of fragments on the available channels when the fragments are roughly of the same size, and when the channels characteristics are similar and stable. Variations in the characteristics of any particular channel can have catastrophic impact on the performance of the whole bundle.

On the transport level, [7] introduces a bandwidth aggregation technique for mobile hosts called Reliable multiplexing transport protocol (R-MTP). R-MTP is a rate based transport protocol capable of multiplexing data from a single application data stream across multiple network interfaces. R-MTP tracks packets inter-arrival for discrimination between congestion based and transmission based losses. A packet-pair probing mechanism is used for estimating the available bandwidth over each channel. Inaccuracies in the bandwidth estimation, due to the fast variation in the network condition, can lead to out of order arrival of packets at the receiver and, in some cases, to re-sequencing buffer overflow. In addition, packet-pair probing mechanisms do not provide an accurate bandwidth estimate in networks with deep buffering.

On the application level, several work exist that use multiple TCP connections (sockets) in order to achieve higher throughput [8–10]. In Ref. [9] authors introduce a modified version of FTP, called XFTP, that use multiple TCP connections in order to simulate a virtual TCP connection with a large receiver window size. Limited receiver window size can throttle the TCP throughput on large delay-bandwidth product channels, such as satellite channels. In order for the XFTP to transfer a file using n connections, it divides the file into m 8 kbytes records (where $m \geqq n$). The sender of the file sends each record over a connection that has enough resources to accept the record. This is determined by using disjunctive wait (select) and

non-blocking writes. Ref. [10] describes a library of Application program interfaces (APIs) called Parallel Sockets (PSockets). The library allows developers of data intensive computing applications, such as data mining, running on a high-speed networks to stripe their data on different open sockets. As in Ref. [9], the application level striping and re-sequencing mechanisms initially target to relieve the network administrators from end-to-end window size tuning, by virtually increasing the receiver window size through aggregating the window sizes of multiple sockets. In Ref. [11], authors present an analytical model for the throughput of parallel sockets. The model is used to find the optimal number of sockets that can achieve higher throughput for the applications and at the same time do not lead to network congestion.

Implementing the bandwidth aggregation on the application level increases the complexity of the applications, as they are responsible for striping and re-sequencing the data. In addition, application level bandwidth aggregation mechanisms stripe the data on the different sockets without taking in consideration the differences in the paths characteristics. This can lead to a situation where a slow path can drag down the throughput of the whole bundle.

3. SCTP overview

SCTP is the fundamental member of a family of protocols designed by the SIGTRAN group to allow SS7 messages to be transported over an unreliable IP infrastructure [12,13]. In SCTP, data transfer between two hosts takes place in the context of an association, as shown in Fig. 1.

All data transferred between the hosts is encapsulated in SCTP packets. SCTP packet contains a common header and a sequence of structures called 'chunks'. 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 out-of-date or false messages into the SCTP

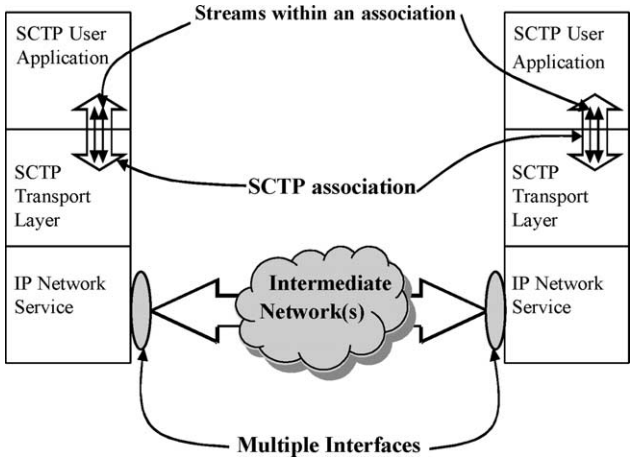

Fig. 1. SCTP association.

Chunk type	Chunk flags	Chunk length
Chunk data		

Fig. 2. The layout of a chunk.

association, and a 32-bit checksum for end-to-end error detection, which is more robust than the 16 bit checksum of TCP and UDP. The chunks can be either control chunks, such as selective acknowledgement (SACK) chunks, or data chunks. The layout of a chunk is shown in Fig. 2. Each chunk includes chunk type, chunk flags, chunk length and chunk value. Control chunks incorporate different flags and parameters depending on the chunk type. To improve transport efficiency, SCTP allows bundling of multiple data and control chunks in a single packet.

The data chunk is the container for the user data transferred in SCTP. Its format is shown in Fig. 3. Data chunks incorporate flags to control the segmentation and reassembly, in addition to the following parameters: Transmission sequence number (TSN), Stream ID, Stream sequence number (SSN), and Payload protocol identifier. Stream ID and SSN are used for multistreaming support, while the Payload Protocol Identifier is included for future flexibility.

Data chunks arriving at the SCTP receiver are acknowledged by transmitting SCTP packet with a SACK control chunk. The format of SACK chunk is shown in Fig. 4.

During association initialization the SCTP end points exchange the size of their receiver window (an indication of the available space in their inbound buffer) and the initial TSN of the data chunks to be exchanged during the association.

SCTP has multiple features that make it different from TCP. The most attractive features are multi-streaming and multi-homing support.

Multi-streaming allows data to be partitioned into multiple streams that have the property of being independently delivered to the application at the receiver. This means that the loss of a data chunk that belongs to a certain stream will only affect the delivery within that stream, without affecting the delivery of other streams. This feature prevents head-of-line blocking problem that can occur in TCP, as TCP supports only a single data stream. Within

Type	Chunk Flags =UBE	Chunk Length = Variable
ASN		
Stream Identifier		Stream Sequence Number
Payload Protocol Identifier		
User data		

Fig. 3. The Data chunk.

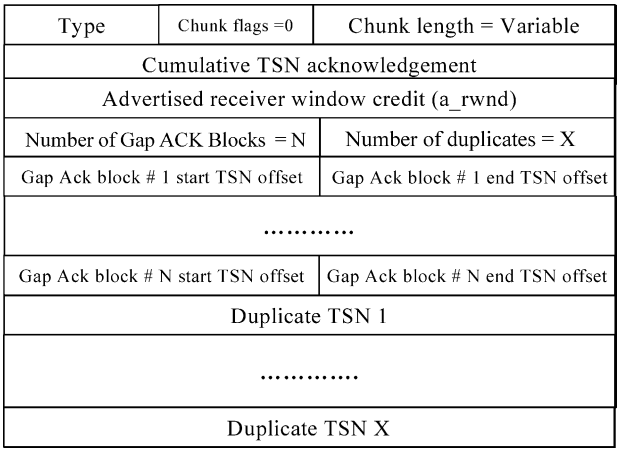

Fig. 4. The SACK chunk.

the association, the stream ID parameter in the data chunk header determines the stream to which the data chunk belongs. Normally, data chunks are ordered within a stream using the SSN, although transfer of unordered data is also supported.

Multi-homing allows a single SCTP endpoint to support multiple IP addresses. In its current form, SCTP multi-homing support is only for redundancy. A single address is chosen as the 'primary' address, which is the destination address for all the data chunks during normal transmission. Retransmitted data chunks use the alternate address(es), to improve the probability of reaching the remote endpoint. Continues failure to send to the primary address ultimately results in the decision to transmit all data chunks to an alternate destination address until the primary address becomes reachable again.

SCTP includes mechanisms for path and peer monitoring. An SCTP instance monitors all the transmission paths to the peer instance of an association. To this end, Heartbeat control chunks are sent over all the idle paths, which are paths that are not currently used for data chunks transmission, regardless whether they are active or inactive. Each Heartbeat chunk has to be acknowledged by a Heartbeat-Ack chunk. A path is considered active if it has been used in the recent past to transmit an SCTP chunk that has been acknowledged by the peer. If transmissions on a certain path fail repeatedly more than a certain configurable limit, *Path.Max.Retrans*, the path is regarded as *inactive*. The number of events in which Heartbeats were not acknowledged within a certain time or retransmission occurred are counted on a per association basis. When a certain configurable limit, *Association.Max.Retrans*, is exceeded the peer endpoint is considered unreachable and the association terminated.

SCTP congestion control is an amalgamation of current best practice for TCP implementations with extensions to deal with multi-homing aspect of SCTP and modifications due to the message rather than stream-based nature of the protocol [14]. The SCTP standard specifies an adaptive sliding window control with adapted versions of the well known TCP slow-start, congestion avoidance, fast retransmit and fast recovery mechanisms [15]. SCTP congestion control mechanisms include two major differences with the equivalent TCP mechanisms. First, the direct dependence of SCTP on the number of bytes acknowledged, rather than the number of acknowledgements received, to increase the congestion window. Secondly, the implicit dependence of SCTP on SACK messages for acknowledging the received data chunks.

4. LS-SCTP design

In this section we present our extension for SCTP (LS-SCTP) to support aggregating the available bandwidth of the transmission paths between the transport endpoints. First, we will discuss the reasons that make SCTP, in its current form, not suitable for load sharing. Then we present solutions to make SCTP load sharing capable.

Our motivations to extend SCTP for load sharing are:

(i) The increase in the number of multi-homed devices. For instance, laptops commonly come with a built in infrared and multiple PCMCIA slots, allowing for multiple communication cards.
(ii) By aggregating the available bandwidth, applications can gain higher throughput, as will be shown in Section 5, which is beneficial for networks with limited bandwidth and high error rate.
(iii) SCTP is more attractive for load sharing than other transport protocols, such as TCP, due to the fact that SCTP inherently support multi-homing, which allows a single SCTP connection to be able to deal with different interfaces. This has the following advantages: (1) It reduces the overhead of initiating and terminating multiple transport connections, (2) There is no need for an upper layer to glue the transport connections and control the data striping, (3) Facilitates the access to the congestion control information of the paths, which allows intelligent data striping [16].
(iv) SCTP attractive features, which are shown to be beneficial for different applications [17].

4.1. Is SCTP suitable for load sharing?

Transmission paths can have different characteristics, in terms of round trip time and congestion state Consequently, if SCTP simply stripes the load on the different paths, data chunks can arrive out of order at the receiver. Out of order arrival of data chunks can unnecessary initiate SACKs transmission to the sender, to report the gaps in the TSN, which increase the network load. In addition, the arrival of four SACKs, as oppose to 3 SACKs in TCP, at the sender reporting the same TSNs will be interpreted

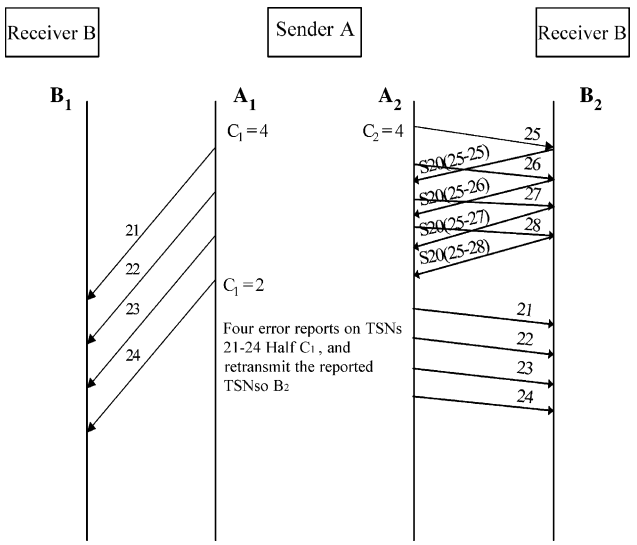

Fig. 5. Effect of distributing SCTP packets on different paths.

as loss of the TSNs. This can lead to unnecessary retransmission of data chunks, as well as halving the congestion window (*cwnd*) of the path on which the TSNs were originally sent. Similar problem was reported in Ref. [18]. To illustrate this problem, we assume two multi-homed hosts *A* and *B*. Host *A* has interfaces A_1 and A_2, and host *B* has interfaces B_1 and B_2, and all the four addresses are bound to an SCTP association. For one of several possible reasons (e.g. path diversity, policy based routing), we assume that the data traffic from *A* to B_1 is routed through A_1, and from *A* to B_2 is routed through A_2. Fig. 5 shows the timeline of events. The vertical lines represent interfaces B_1, A_1, A_2 and B_2. Each arrow depicts the departure of a packet from one interface and its arrival at the destination. The labels on the arrows are either SCTP TSNs or labels $ST_C(T_{GS} - T_{GE})$ which represents a packet carrying SACK chunk with cumulative ACK T_C, and gap ACK for TSNs T_{GS} through T_{GE}. C_1 is the *cwnd* at A for destination B_1, and C_2 is the *cwnd* at *A* for destination B_2. C_1 and C_2 are denoted in terms of path maximum transfer units (PMTUs). We assume that the round trip time (*RTT*) to $B_1 > 2^*$ *RTT* to B_2, and at a certain time the *cwnd* of both destination addresses is 4 PMTU. Also we assume that data chunks with TSNs 21–24 were sent to B_1, and data chunks with TSNs 25–28 were sent to B_2. Because of the difference in the *RTT*s, data chunks sent to B_2 arrive earlier to the receiver end point initiating 4 SACKs that report the loss of TSNs 21–24, this will falsely lead the sender to retransmit TSNs 21–24 to B_2, and cutting the *cwnd* of B_1 to one half of its value.

Techniques that depend on the differentiation between the transmission and retransmission in order to avoid congestion window overgrowth [18], as well as techniques for adapting the fast transmission threshold with the degree of reordering in the network [19,20], can partially solve the problem. Although these techniques adapts the sender with the reordering in the network, the receiver is still not able to differentiate between the reordering that is due to the difference in the delay of the paths and that is due to packet losses. Thus the receiver continues to generate SACKs to report the out of order arrival of the data chunks, which consume the network resources. Delaying the generation of the SACK from the receiver, in order to account the out of order arrival of data chunks due to the difference in the paths delays, will work fine in the case of lossless networks, but in the case of networks with losses, the delayed SACK will delay reporting packet losses to the sender, which can eventually timeout.

Our goal is to design a bandwidth aggregation technique in SCTP that has the following features:

1. Robust to the variations in the paths characteristics without, unnecessary, consuming the network resources.
2. Adaptable to the failure/restoration of the paths within the association.
3. Backward compatible with standard SCTP.

4.2. Architectural overview of LS-SCTP

LS-SCTP is based on the idea of separating the association flow control from congestion control. In LS-SCTP, the flow control is on association basis, thus both the sender and receiver endpoints use their association buffer to hold the data chunks regardless their transmission paths. On the other hand, congestion control is performed on per path basis, thus the sender has a separate congestion control for each path, and these congestion control mechanisms can be used simultaneously for data chunks transmission. This provides the sender endpoint with a virtual congestion window (*cwnd*) size equal to the aggregate of the *cwnd*s of all the paths within the association that are used for load sharing. Standard SCTP does not separate between the flow and congestion control mechanisms, as it was designed to deal with a single path, and both mechanisms use the same transmission sequence number. In LS-SCTP, the congestion control on each path is following RFC2960 [1], so as to insure fair integration with other traffic in the network. An architectural view of LS-SCTP is shown in Fig. 6 [21].

In order to separate the flow control from the congestion control, LS-SCTP uses two different sequence numbers.

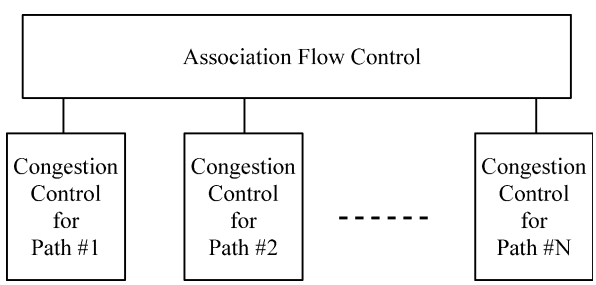

Fig. 6. Architectural view of LS-SCTP.

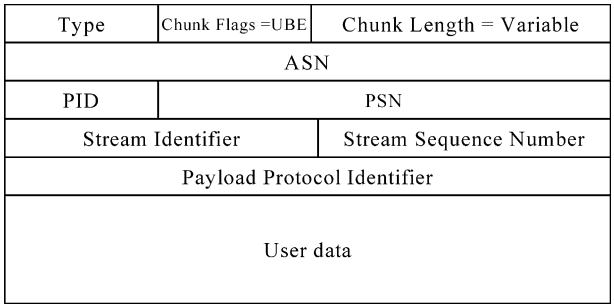

Fig. 7. Load sharing data chunk.

The first sequence number is the association sequence number (ASN), that is a per association sequence number, and is used to reorder the received data chunks at the receiver association buffer, regardless the path from which they have been received. The second sequence number is the path sequence number (PSN), which is a per path sequence number, used for reliability and congestion control on each path. In addition, LS-SCTP continues to use the stream sequence number (SSN) for ordering the data chunks within the association streams.

To support load sharing, we defined a new data chunk 'load sharing data chunk' by adding two new parameters to the standard SCTP data chunk. The first parameter is a 4 bits Path Identifier (PID), which identifies the path used for the data chunk transmission. The second parameter is 12 bits PSN, which is a monotonically increasing sequence number for the data chunks transmitted over the same path. Fig. 7 shows the load sharing data chunk.

For acknowledging the received data chunks, LS-SCTP uses a new SACK chunk 'Load Sharing SACK (LS-SACK)' shown in Fig. 8.

As data chunks are received and buffered in the receiver association buffer, the receiver endpoint decrements the Advertised receiver window credit (a_rwnd), which represents the available space in the buffer, by the number of bytes received and buffered. As data chunks are delivered to the upper layer application (ULA) and released from the receiver association buffer, the receiver increments the a_rwnd by the number of bytes delivered to the upper layer. When the receiver sends a LS-SACK, it places the current value of a_rwnd into the a_rwnd field, and it sets the Cumulative ASN ACK, which represents the highest ASN delivered to the ULA, taking into account that the data sender will not retransmit data chunks that have been acknowledged via the Cumulative ASN ACK (i.e. will drop them from its association buffer). In addition, the receiver sets the Cumulative PSN, which represents the highest in-sequence PSN received on the path over which the LS-SACK will be sent, as well as the missing and duplicate PSNs. The receiver of the LS-SACK uses the PSN related values to update the congestion control variables of the corresponding path.

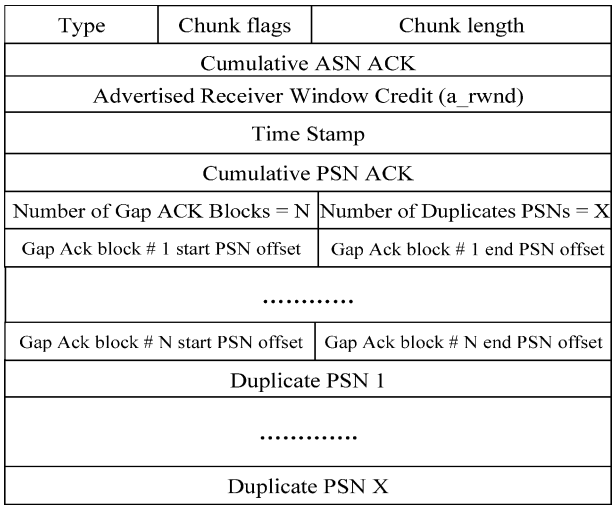

Fig. 8. Load sharing SACK chunk.

As the LS-SACKs are received on all the paths that are used for load sharing within the association, and due to the different delays of these paths, LS-SACKs can be received out of order. Out of order arrival of the LS-SACKs can cause the sender to develop an incorrect view of the receiver's buffer space, as the LS-SACK includes the receiver's free buffer space at the time it was sent. For this reason, the LS-SACK includes a time stamp field that is used by the sender of the LS-SACK to indicate when it was sent. The receiver of the LS-SACKs uses the Time Stamps to determine their order. The receiver of an old LS-SACK, does not update its view of the peer's free buffer space, based on the values in the LS-SACK, but on the other hand, it uses the remaining LS-SACK information to update the congestion control variables for the path from which the LS-SACK was received.

The congestion control variables for each path, including congestion window size (*cwnd*), slow-start threshold (*ssthresh*), Round Trip Time (*RTT*), retransmission time out (*RTO*), outstanding data count, and partial bytes acknowledged (*partial_bytes_acked*), are maintained in a data structure called the 'Virtual Buffer'. In addition, the virtual buffer keeps track of the PSNs sent or received on each path. The receiver window size (*rwnd*), which represents the available space in the peer's association buffer, as well as the total outstanding data count, are kept on an association basis.

Figs. 9 and 10 illustrate the inbound and outbound message passage in LS-SCTP.

As the sender's application layer passes a data packet to the LS-SCTP, the data packet is fragmented, if required, and assigned an in-sequence ASN as well as a Stream ID and SSN, before it is placed in the association buffer. The path assignment module selects the transmission path for the data chunks and accordingly the PID and PSN are assigned to each data chunk. Before passing the data chunks to the network layer, for transmission on the selected paths, the LS-SCTP can bundle the data chunks with control chunks for transport efficiency [1]. At the receiver, data chunks are unbundled from control chunks, and the congestion control

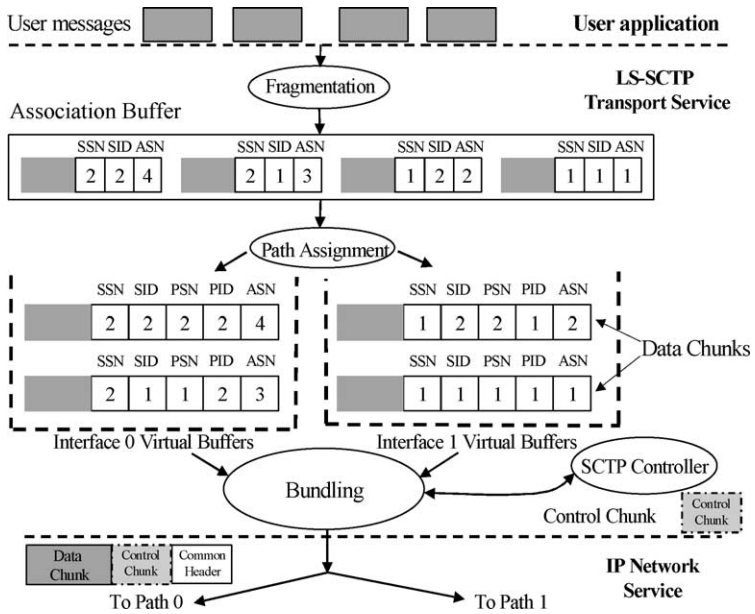

Fig. 9. Outbound message passage in LS-SCTP.

variables, in the virtual buffer corresponding to the path from which the data chunks were received, are updated. Then the data chunks are placed in the association buffer, until they are delivered to the application, in the order of the stream they belong to.

4.3. Path assignment module

The path assignment module in LS-SCTP, is responsible for assigning transmission paths for the data chunks. The assignment can be based on a criteria requested by the ULA during the association initiation. For example, the ULA may request the LS-SCTP to assign different data streams, within the association, to different paths. As far as the data chunks in the different streams are ordered by their SSN and delivered independently to the ULA, there will not be any problem at the receiver if the streams are assigned to different paths. The ULA request may also be in the form of quality of service (QoS) requirement for each stream. Based on the QoS requirements for the streams, as well

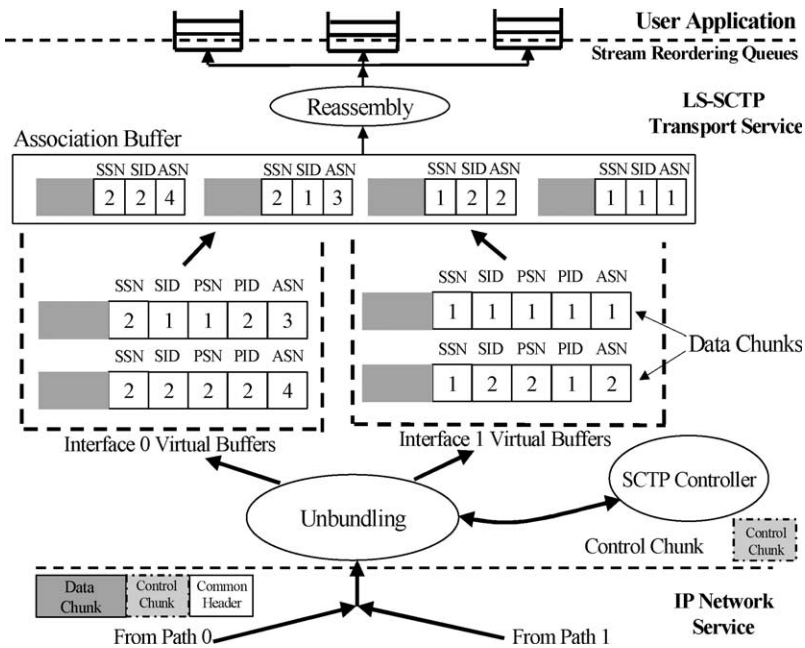

Fig. 10. Inbound message passage in LS-SCTP.

as the current characteristic of the transmission paths, the path assignment module can dynamically change the streams paths assignment, to fulfill the streams QoS. If no assignment criteria were specified, the Path Assignment module assigns the data chunks based on the bandwidth availability of the paths.

Round robin path assignment mechanism, similar to that used in Ref. [6], is not suitable when the transmission paths have different and variable characteristics. Such mechanisms can limit the throughput of the association to the throughput of the slowest path. Thus it is important that the path assignment be based on the ratio of the bandwidth of the paths within the association.

LS-SCTP uses the current congestion window (*cwnd*) of each path, as an estimate of its current bandwidth-delay product [22]. The path assignment module assigns data chunks to the paths according to the *cwnd*/RTT of each path. After assigning a data chunk to a path, the data chunk is assigned the PID that identifies the path, as well as an in-sequence PSN.

At the receiver, the data chunks do not compete on association buffer, as the sender controls the amount of data injected on all the paths, based on its view of the free space in the receiver's association buffer (*rwnd*), as well as the total outstanding data on all the *active* paths. In order to compensate the differences in the paths *RTT*, the minimum receiver association buffer should be based on the summation of the bandwidth of all the paths and the maximum *RTT* [7], as shown in Eq. (1).

$$\text{Minimum receiver association buffer} = \left(\sum_{i}^{N} B_i\right)\text{RTT}_{\max} \tag{1}$$

where, B_i is the bandwidth of path i, N is the number of *active* path within the association and RTT_{Max} is the maximum *RTT*.

To improve the probability that a retransmitted data chunk(s) reaches the peer endpoint, the path assignment module retransmits the data chunk(s) on a different path than the one used for the original transmission. Delaying the arrival of a data chunk can stall the association, because data chunks that were sent on other paths will be queued at the receiver waiting the missing chunk. At a certain point the receiver's association buffer space will limit the sender from sending more data chunks, until it receives the missing chunk. The PSN that was assigned to the data chunk(s) on the original path is returned to the PSN pool of this path, and can be used for new data chunk(s). The retransmitted data chunk(s) will be assigned PID and PSN on the retransmission path. In addition, the outstanding data counts on the old and new paths are adjusted according to the size of the retransmitted data chunk(s).

Similarly, when time-out occurs on one of the *active* paths within the association, the path assignment module assigns the timed-out data chunks to different paths than the one they were originally timed-out on, that have enough capacity to transmit them. On the contrary of standard SCTP, that uses a new data chunk for probing the timed-out path, the Path Assignment module assigns a copy of one of the retransmitted data chunks for probing the path. The reason behind this, that there is a probability that the timed-out path can continue to fail, which delays the re-transmission of the new data chunk until the path time out again. Mean while the receiver buffer will start to fill up with data chunks, following the delayed one, that are received from other *active* paths within the association, and this can eventually lead the whole association to stall. The situation can be worse if the path fails for a long period. As the *RTO* of a path increases exponentially with consecutive time-outs, the association can repeatedly stall for exponentially increasing periods, as will be shown in Section 5. This can continue until either the path recovers or it is marked as *inactive* by the Path Monitor, as will be described in Section 4.4.

In a wired domain it is possible that multiple paths can be sharing the same bottleneck. This makes an LS-SCTP association that is using these paths more aggressive, to other flows passing through the bottleneck, than a single SCTP association. One solution to this problem is to use schemes similar to that described in Ref. [23] in order to detect the shared bottleneck between the paths. If it can thus be inferred that more than one path within the association are sharing a bottleneck, only one of them is considered for load sharing. Also approaches like [16] can be employed which allows an ensemble of concurrent flows to share the bandwidth at intermediate nodes and obtain consistent performance, without adversely affecting other network flows.

4.4. Path monitoring

As LS-SCTP utilizes multiple paths for transmission, a failure of a single path, or the increase in a path loss rate can affect the throughput of the whole association. For this reason, LS-SCTP includes a path monitoring mechanism referred to as the "Path Monitor" that is responsible for updating the *active* paths list, which includes all the paths that can be used for load sharing. The Path Monitor is not only monitoring the availability of the paths but also monitoring their quality, in terms of loss rate, delay, etc. It can update the *active* paths list based on a network feedback, if available, regarding the failure/recovery of paths within the association. In addition, the Path Monitor removes a path from the *active* paths list when the number of consecutive retransmission time-outs on the path exceeds *Path.Max.Retrans*, which is set to 5, or when the path quality deteriorate to a limit that can affect the performance of the whole association. Currently, we are using a reasonable default threshold for the average loss rate on each path, and basing the path membership in the *active*

paths list on the threshold value. *Inactive* paths remain as a part of the association, and the sender keeps monitoring them through Heartbeat control chunks. As soon as a path recovers, the Path Monitor adds it again to the *active* paths list. As will be shown in the experimental results, this technique prevents stalling the association while waiting for a missing data chunk.

In SCTP the end point monitors an *inactive* path i by sending a Heartbeat chunk every H_i seconds, which is determined using the following formula:

$$H_i = RTO.\text{Initial} + HB.\text{Interval} \times (1 + \delta) \quad (2)$$

where *RTO.Initial* is the initial *RTO*, *HB.Interval* is a configurable parameter to control the Heartbeat chunks generation rate, and δ is random value between -0.5 and 0.5 designed to add some jitter into the timing of each individual Heartbeat. RFC2960 [1] recommends $RTO.Initial = 5$ s, $HB.Interval = 30$ s. These values were recommended in SCTP, as it only considers the alternative paths between the sender and receiver for retransmission of lost data chunks or as a backup for the primary path. But as LS-SCTP is considering using all the paths within the association for simultaneous transmission of data chunks, it is important for LS-SCTP to detect the recovery of the *inactive* paths as soon as possible. For this reason, we recommend using short *HB.Interval*, in order to increase the probing rate for the *inactive* paths.

4.5. Fragmentation and reassembly

The sender should track the association Path MTU (PMTU), which is the smallest MTU discovered for all of the peer's destination addresses When fragmenting messages into multiple parts, the association PMTU is used to calculate the size of each fragment. This allows retransmissions to be seamlessly sent to an alternate address, without encountering IP fragmentation. In addition, LS-SCTP sender assigns all the fragments of a given message to the same path, so as to ensure that the message fragments will be received in order at the receiver. This reduces the time required to reassemble the original message. The sender endpoint assigns the same ASN and SSN to each of the data chunks in the fragments series. After assigning the fragments data chunks to a transmission path, they are assigned, in sequence, PSN on this path. As in SCTP [1], LS-SCTP receiver recognizes fragmented data chunks, by examining the flags in the data chunks header, and queue the fragmented data chunks for re-assembly. Once the user message is reassembled, LS-SCTP passes the re-assembled user message to the message stream for possible reordering and final dispatching.

4.6. Backward compatibility of LS-SCTP

To insure backward compatibility, LS-SCTP uses a similar technique to that used for supporting Explicit Congest Notification (ECN) [24]. During the association initiation both end points have to exchange a Load-Sharing-Supported parameter, in order to start using LS-SCTP over the association. If any of the end points is not load sharing capable or is load sharing capable but not willing to use load sharing over the new association, in order to conserve the battery power for example, it does not include this parameter. In this case, standard SCTP is used over the new association.

5. Performance study

In order to examine the performance of LS-SCTP, we extended our SCTP implementation, in OPNET network simulation software [25]. In our simulation, we used the network topology shown in Fig. 11. We assume that an SCTP association is already initiated between the two hosts, and the association is unidirectional, which means that data chunks will only be sent from the host A (sender) to host B (receiver). In addition, we use only one stream within the association.

Nodes 1-N are used to configure the characteristics of the transmission paths in terms of bandwidth, loss rate, delay and delay jitter. In addition, the nodes can be configured to represent a short term or a long term path failures, as well as intermittent failures. As we need to examine the performance of LS-SCTP under diverse paths characteristics, and in order to have full control on the configuration of the paths during our performance study, we assumed that the paths do not share a common bottleneck. Techniques to detect and deal with shared bottlenecks are described in Section 4.3.

The application packet size is set to 1 Kbytes. We set the application packet inter-arrival time to insure that the application will always have packets for transmission. During all our simulations, the application starts generating packets after 0.1 s from starting the simulation.

The association buffer sizes (both sender and receiver) were set according to Eq. (1). Unless specified otherwise, the bandwidth of all the paths between host A and host B were set to 1.5 Mbps, with $RTT = 100$ ms.

In our performance study we used the association throughput as a performance metrics, which is defined as the number of bytes delivered to the receiver's application layer per second.

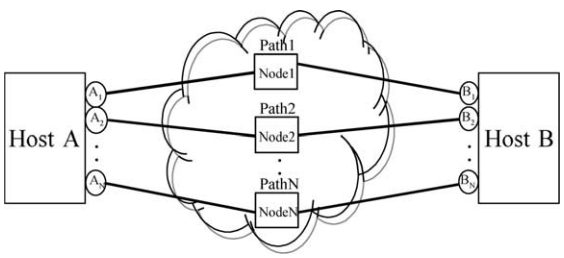

Fig. 11. Network model for the simulation study.

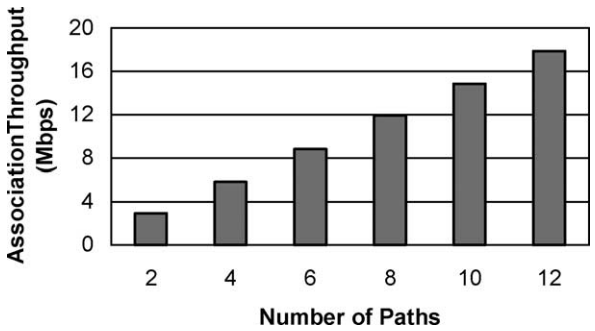

Fig. 12. Association throughput versus the number of transmission paths.

Specific questions we looked at in our performance study:

(1) Does LS-SCTP scale well with the number of transmission paths?
(2) What is the effect of packet losses on LS-SCTP throughput?
(3) How does LS-SCTP perform with bandwidth variation?
(4) How does LS-SCTP react to short term and long term path failures?

5.1. Scalability with the number of transmission paths

We examined the scalability of LS-SCTP with increasing the number of transmission paths. As far as we are examining the maximum throughput that can be achieved by LS-SCTP, we assumed that there are no losses on the paths. Fig. 12 shows the LS-SCTP association throughput as we increase the number of *active* paths within the association. It can be observed that the performance of LS-SCTP scales well with increasing the number of *active* paths, and it is able to aggregate their bandwidth efficiently.

5.2. Effect of packet losses

In this experiment, we examined the performance of LS-SCTP under different packet loss rates. We assumed that the association includes two paths, namely path 1 and path 2.

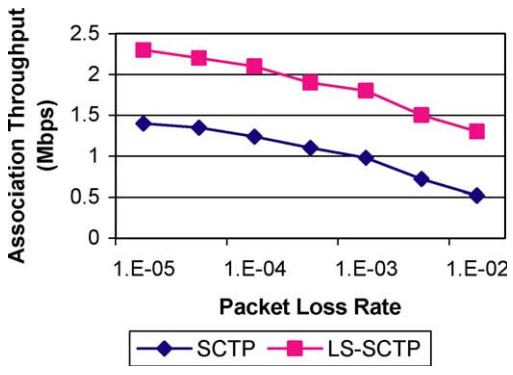

Fig. 13. Total throughput versus packet loss rate.

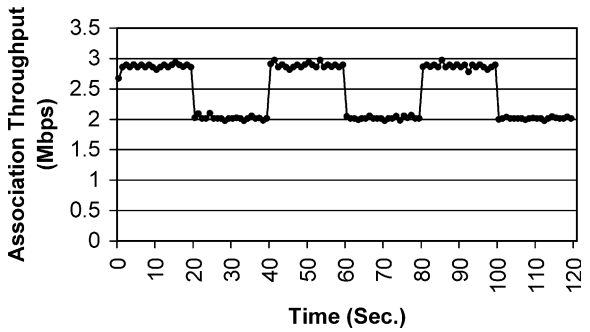

Fig. 14. Effect of bandwidth fluctuation on the association throughput.

We configured node 1, on path 1, to drop packets randomly in the range between 0.001 and 1%, while we assumed that there is no losses on path 2. Fig. 13 shows the total throughput of SCTP and LS-SCTP under different packet loss rates. Because of the randomization in this experiment, each point in the graph represents the average of 10 samples with different seeds. It can be shown that the although the paths that are used for load sharing have different packet loss characteristics, LS-SCTP still achieves higher throughput than SCTP, due to the packet retransmission technique used in LS-SCTP.

5.3. Effect of bandwidth fluctuation

In this experiment, we studied the effect of bandwidth fluctuation on LS-SCTP performance. We used two paths between host A and host B, namely path 1 and path 2. We represented the bandwidth fluctuation on path 1, by using a square-wave background traffic with amplitude 1 Mbps, and period $t = 40$ s. While we assumed that there is no background traffic on path 2. We run the simulation for 120 s.

Fig. 14 shows the association throughput. It can be seen that LS-SCTP is able to adapt with the bandwidth fluctuation on the transmission paths, as the sender is splitting the load on the paths based on an estimate of their bandwidths. Also LS-SCTP is able to utilize the available bandwidth between the communicating endpoints. The difference between the association throughput and the available bandwidth is mostly due to the overhead from various protocol levels.

5.4. Short term path failure

We examined the effect of short term path failure on the LS-SCTP. By short term path failure we mean that the path will recover before it is marked as *inactive* by the sender, i.e. before the number of consecutive path timeouts exceed *Path.Max.Retrans*, which is set to 5. We assumed that there are two *active* paths within the association, path 1 and path 2. Path 1 failed for 10 s, after 5 s from initiating the association. During our simulation, we compared two techniques for handling timed-out paths. The first technique, which we refer to as SCTP Time-out Handling Technique, is

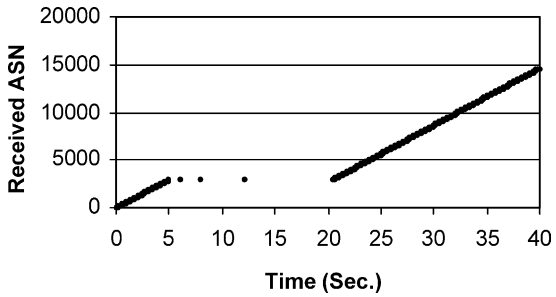

Fig. 15. Received ASNs progression under SCTP time-out handling technique.

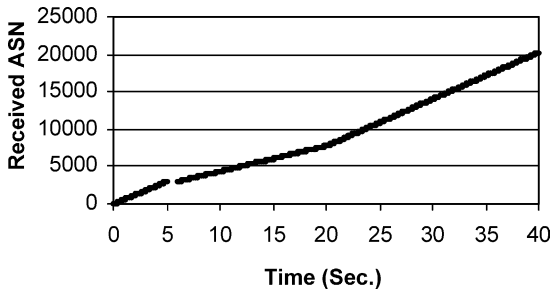

Fig. 17. Received ASNs progression under LS-SCTP timeout handling technique.

similar to that used in SCTP, where the timed-out data chunks are retransmitted on an alternative path, and a new data chunk is used for probing the bandwidth of the timed-out path. The second technique, which we refer to as LS-SCTP Time-out Handling Technique, is the technique we proposed in Section 4.3, where the timed-out data chunks are retransmitted on alternative path, other than the one they already timed-out on, and a copy of one of the retransmitted data chunks is used for probing the timed-out path. After the receiver sends a LS-SACK to acknowledge the data chunk copy, the sender starts to use the path again for transmission of data chunks.

Fig. 15, shows the received ASNs progression, under the SCTP time-out handling technique. As can be seen that the whole association stalled, for around 15 s, after path 1 timed-out. The reason behind this behavior, that after the first time-out of path 1, the sender retransmitted all the timed-out data chunks on path 2, then started to probe path 1 with a new data chunk, ASN = 2944. At the same time, the sender continued to send the data chunks following ASN = 2944 on path 2. This caused the receiver association buffer to fill up, as it is waiting for ASN = 2944, preventing the sender from sending more data chunks. This caused the whole association to stall, until ASN = 2944 is timed-out and retransmitted on path 2, allowing the receiver to deliver all the data chunks in the association buffer to the application. At this time, the sender resumed the transmission again. This cycle repeated for 3 times. Also, as we can notice from the figure, that even after path 1 is recovered at $t = 15$ s, the association continued to stall, until path 1 is timed-out again at $t = 20$ s. The reason behind this that path 1 *RTO* increased exponentially after each time-out, which led the association to stall for exponentially increasing periods of time.

Fig. 16 shows the association throughput, with SCTP time-out handling technique. As can be seen that the association throughput dropped to zero most of the interval between $t = 5$ to $t = 20$.

We repeated the simulation using LS-SCTP time-out handling technique. Received ASN progression is shown is Fig. 17. As can been seen, that using a copy of a retransmitted packet to probe path 1 *cwnd*, led the association to resume using path 2. This had the effect of reducing the association throughput, as shown in Fig. 18, but with minimum interruption to the association progress. After the recovery of path 1, the sender started to utilize it for transmission. This consequently increased the association throughput. The reduction of the association throughput after path 1 failure, occurred for a duration approximately equal path 1 *RTO*, before the sender stopped considering path 1 for transmitting new data chunks. This throttling interval can be decreased by increasing the receiver association buffer, as this will allow the receiver buffer to absorb more sender's transmission before filling up.

5.5. Long term path failure

We examined the effect of long term path failure on the LS-SCTP. With long term path failure, we mean that the path will recover after the sender's Path Monitor

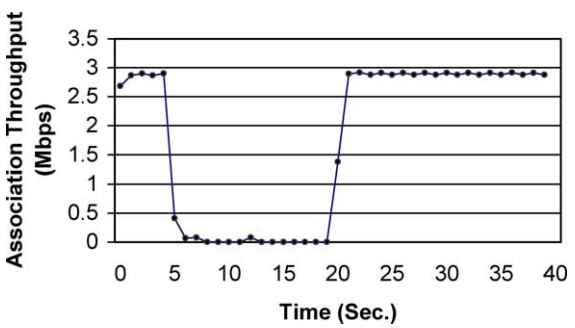

Fig. 16. Association throughput under SCTP time-out handling technique.

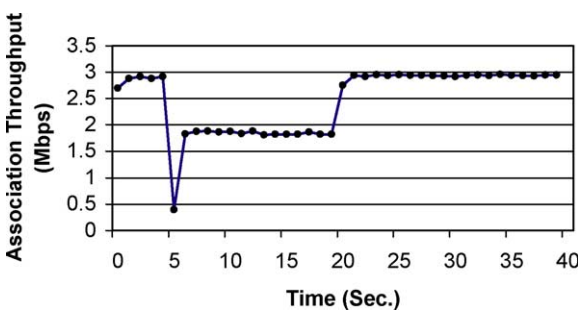

Fig. 18. Association throughput under LS-SCTP time-out handling technique.

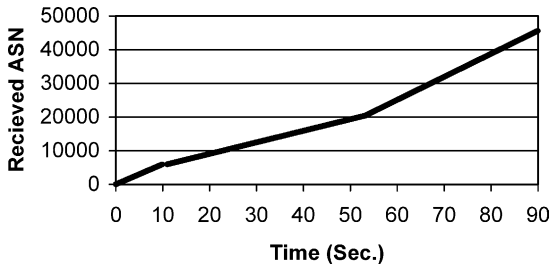

Fig. 19. Received ASNs progression.

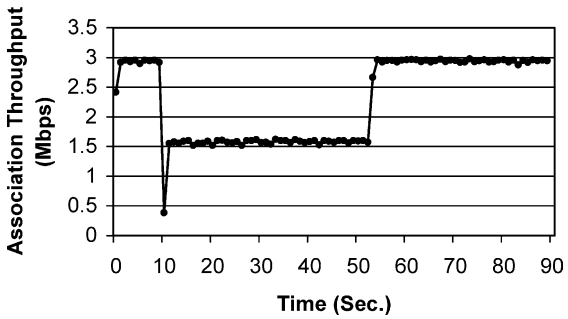

Fig. 20. Association throughput.

marks the path as *inactive*, i.e. after the number of consecutive path time-outs exceed *Path.Max.Retrans*. Again we assumed that there are two *active* paths within the association, path 1 and path 2, and path 1 failed for 40 s, after 5 s from initiating the association. The failure duration was selected to insure that the sender's Path Monitor would set path 1 as *inactive* early enough before its recovery. After path 1 is set *inactive*, the sender kept monitoring the path using Heartbeat control chunks. During our simulation we set the HB.Interval = 1 s, so as to increase the probing rate, as discussed in Section 4.4.

As shown from Figs. 19 and 20, after the failure of path 1 the sender continued to use path 2 for transmission of new data chunks. The data chunk copy on path 1 continued to time-out, until path 1 is set as *inactive* by the Path Monitor at $t = 33$ s. The LS-SCTP sender stopped probing path 1 *cwnd*, but at the same time it continued to monitor the status of path 1 using the Heartbeats chunks. After path 1 recovery at $t = 50$ s, the first Heartbeat sent by the sender, at $t = 52$ s, is acknowledged by a Heartbeat-Ack. Since then, the sender's Path Monitor set path 1 as *active*, and continued to use it for data chunks transmission.

6. Conclusions and future research

In this paper, we propose an extension for SCTP, LS-SCTP, which aggregates the available bandwidth on the paths between the sender and receiver. We first showed that SCTP in its current form is not suitable for load sharing. Then, we suggested remedies for SCTP to make it load sharing capable. We proposed separating the association flow control and congestion control. The congestion control is performed per path while the flow control is performed per association. Also we proposed modifying the sender to control the distribution of data on the available paths based on an estimate of the bandwidth of each path. In addition, we presented a technique that allows the sender to react to path failure and path quality deterioration without stalling the whole association. LS-SCTP performance study shows that it is scalable with the number of transmission path. Also it shows that LS-SCTP is able to deal efficiently with short term as well as long term path failures. We believe that our proposed bandwidth aggregation mechanism is extremely beneficial for networks with limited bandwidth and high loss rates, especially with the increase of multi-homed devices that can be simultaneously connected to different networks through different interfaces.

Work is in progress in many directions. We are investigating techniques for achieving end-to-end service differentiation by mapping different streams within the association, to different paths. We are studying the different QoS parameters for the association streams, as well as examining techniques for accurate monitoring to the congestion status of transmission paths. This will allow the sender endpoint to perform accurate mapping between the streams and the paths, in order to fulfill the streams QoS requirements. In addition, we are extending the SCTP Linux implementation with our proposal for load sharing, in order to further examine the performance of LS-SCTP in wire line as well as in wireless networks.

7. Disclaimer

The views and conclusions in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research laboratory or the US Government.

References

[1] R. Stewart, Q. Xie, et al., Stream Control Transmission Protocol, IETF (2000) 2960 Request for Comments.
[2] R. Katz, G. Gibson, D. Pattersan, Disk system architecture for high performance computing, Proceedings of the IEEE 77 (12) (1989).
[3] C. Brenden, S. Traw, M. Smith, Striping Within the Network Subsystem, IEEE Network 9 (4) (1995) 22–32.
[4] Computer Staff, Gigabit network testbeds, IEEE Computer 23 (9) (1990) 77–80.
[5] J. Duncanson, Inverse Multiplexing, IEEE Communications Magazine 32 (4) (1994) 34–41.
[6] A. Snoeren, Adaptive inverse multiplexing for wide-area wireless networks, Proceedings of the IEEE GLOBECOM'99 (1995).
[7] L. Magalhaes, R. Kravets, Transport level mechanisms for bandwidth aggregation on mobile hosts, Proceedings of the IEEE ICNP'01 (2001).

[8] J. Lee, D. Gunter, et al., Applied techniques for high bandwidth data transfers across wide area networks, Proceedings of the Computers in High Energy Physics CHEP'01 (2001).
[9] M. Allman, H. Kruse, S. Ostermann, An application-level solution to TCP's satellite inefficiencies, Proceedings of the Workshop on Satellite-Based Information Services WOSBIS'96 (1996).
[10] H. Sivakumar, S. Bailey, R. Grossman, PSockets: the case for application-level network striping for data intensive applications using high speed wide area networks, Proceedings of the IEEE Supercomputing SC'00 (2000).
[11] T. Hacker, B. Athey, The end-to-end performance effects of parallel tcp sockets on a lossy wide-area network, Proceedings of the IEEE IPDPS'02 (2002).
[12] L. Ong, I. Rytina, et al., Framework Architecture for Signaling Transport, IETF (1999) 2719 Request for Comments.
[13] H. Elsayed, A. Abd El Al, T. Saadawi, M. Lee, Synchronization algorithm for SCTP network, Proceedings of the International Workshop on Multimedia Network Systems and Applications MNSA'03 (2003).
[14] R. Brennan, T. Curran, SCTP congestion control: initial simulation studies, Proceedings of the International Teletraffic Congress CFP'01 (2001).
[15] M. Allman, V. Paxson, W. Stevens, TCP Congestion control, IETF (1999) 2581 Request for Comments.
[16] H. Balakrishnan, H. Rahul, S. Seshan, An integrated congestion management architecture for Internet host, Proceedings of the ACM/SIGCOMM'99 (1999).
[17] Balk, M. Sigler, M. Gerla, M. Sanadidi, Investigation of MPEG4 Video Streaming over SCTP, Sixth World Multiconference on Systemics, Cybernetics, and Informatics (SCI 2002) July 14–18, Orlando, FL, USA (2002).
[18] J. Iyengar, A. Caro, et al., SCTP Congestion Window Overgrowth During Changeover, Proceedings of the Systemics, Cybernetics and Informatics Conference SCI'02 (2002).
[19] E. Blanton, M. Allman, On making TCP more robust to packet reordering, ACM Computer Communication Review (2002).
[20] M. Zhang, B. Karp, S. Floyd, L. Peterson, RR-TCP: a reordering robust TCP with DSACK, Technical Report International Computer Science Institute ICSI'02 (2002).
[21] Abd El Al, T. Saadawi, M. Lee, Load Sharing in stream control transmission protocol, IETF draft, draft-ahmed-lssctp-00.txt, 5/03.
[22] J. Semke, J. Mahdavi, Automatic TCP buffer tuning, computer communication review, ACM/SIGCOMM 28 (4) (1998).
[23] D. Rubenstein, J. Kurose, D. Towsley, Detecting shared congestion flows via end-to-end measurements, Proceedings of the ACM/SIGMETRICS'00 (2000).
[24] K. Ramakrishnan, S. Floyd, A proposal to add explicit congestion notification (ECN) to IP, IETF (1999) 2481 Request for Comments.
[25] Opnet Modular, http://www.opnet.com.