HEARTBEAT ACK chunks to acquire their states.

Naturally, it is considered whether all the paths to a multi-homed endpoint could be used for transmitting simultaneously. When the primary path's congestion window (cwnd) doesn't permit transmitting, the data load can be transmitted on other paths. By this kind of load sharing, the traffic can be shared among all the paths to a multi-homed endpoint. Because of employing more than one path, load sharing should improve the throughput of SCTP association. Original SCTP performs congestion control mechanism on one path. If the same congestion control mechanism is performed simultaneously on multiple paths simply, there are some side effects. The side effects are expressed in the paper. The SCTP load sharing scheme is proposed to eliminate the side effects.

The paper is organized as followings: Section 2 presents the basic idea of SCTP Load Sharing; section 3 introduces the LSFR algorithm overcoming the side effects of simple load sharing; section 4 presents the simulation results in NS-2; section 5 overviews the related works; section 6 gives the conclusion and a vision of future works on SCTP Load Sharing.

2. SIMPLE SCHEME OF LOAD SHARING

Our load sharing scheme is an improvement on the SCTP sender, and it does not have any restriction or requirement on the SCTP receiver. A SCTP endpoint with load sharing can communicate with any other SCTP receiver with or without load sharing scheme. It is to say, load sharing is compatible with the original SCTP.

2.1. Basic Scheme

Load sharing is designed for the instant transmission of application data. Only when the primary path's cwnd is not large enough for transmitting the application data, may other paths be selected for the transmission. In the case of lightweight load, the primary path itself may be sufficient for the transmission.

Not all paths are suitable for load sharing, e.g. inactive paths. A set of criteria is set up for selecting the eligible paths. In the course of transmission, when the primary path's cwnd is used up, load sharing starts. The sender will switch to another path and carry on the transmission until the current path uses up its cwnd or all the data is sent. If the current cwnd is used up too, the sender will continue to choose another path and perform the same procedure until all the paths use up their cwnds or there is no data to be sent.

2.2. Path Selection

For the purpose of implementing load sharing, firstly it should be determined which path(s) should be selected for transmission. We set up a transmission path set, which contains all the paths used for transmitting in a sending action. In SCTP Load Sharing there are three rules for justifying a path to be selected into the set:

1) The Primary Path is in the set;
2) It is active;
3) It is errorless;

Except for the first condition, the other two conditions have the AND relation, which means any path in the set should have the two attributes simultaneously. Of course the Primary Path should have the attributes as well. The second and third rules assure the selected

paths' availability.

The eligible path set is a logical conception, which means, an implementation may or may not build such a data structure. After a sending action on one path, the next path to be chosen is always the optimal path among the backup paths. Thus a high throughput may be obtained. The path should be cwnd maximum and round trip time (rtt) minimum. The selected path may have much smaller rtt than the previous path and thus results in loss-of-order. Such loss-of-order will cause severe side effects of simple load sharing, which is expressed in the following section.

3. SPLIT FAST RETRANSMIT FOR LOAD SHARING (LSFR)

3.1. Side Effects of Basic Scheme of Load Sharing

Using multiple paths is obviously better than using a single path because multiple paths can offer greater transmission ability. However, load sharing will bring some problem if only the basic scheme is deployed. The main point is the unnecessary fast retransmission. As the paths are not of the same quality, i.e. they may differ greatly in error rate, delay, etc. Packets sent at the same time on different paths may arrive at the receiver at different time. In some cases even later sent packets may arrive at the receiver earlier than the earlier sent packets. Consequently, the receiver will find holes in the received chunk sequence (TSN) and response with a SACK with Gap Ack Blocks. In fact, those TSNs in holes are just on the way coming. Because of the delay the missing report count at the sender is increased as it receives more SACKs with Gap Ack Blocks. Fast retransmission starts when the missing report count equals 4. [1] The delayed packets are regarded as loss and retransmitted. Actually such retransmission is unnecessary. The holes are very big under the circumstance of load sharing. The reordering is quite significant compared with the single path circumstance.

Evidently, unnecessary retransmission bring extra traffic load to the whole network. This is undesirable since it wastes network resources. On the other hand the fast retransmission reduces the size of the path's cwnd and limits the throughput of the association. We simulated a simple load sharing to observe the side effects. Experiments show in the case of good link qualities, SCTP with simple load sharing performs even worse than SCTP without it.

Here is an example of unnecessary fast retransmission.

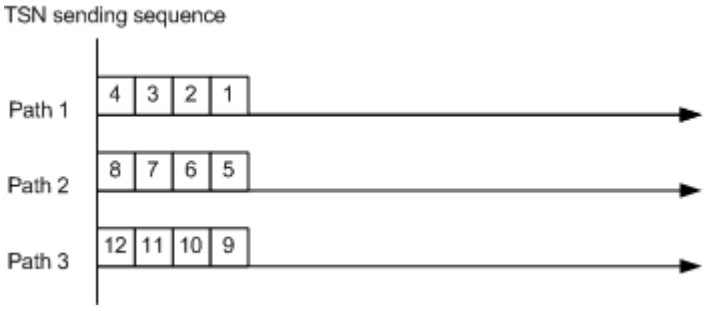

Figure 1. (a) Load sharing using 3 paths

Figure 1 (a) gives the scenario of SCTP load sharing. In the example, 3 paths are used simultaneously for transmission. Path 1 transmits TSN 1~4; Path 2 does TSN 5~8; Path 3 does TSN 9~12.

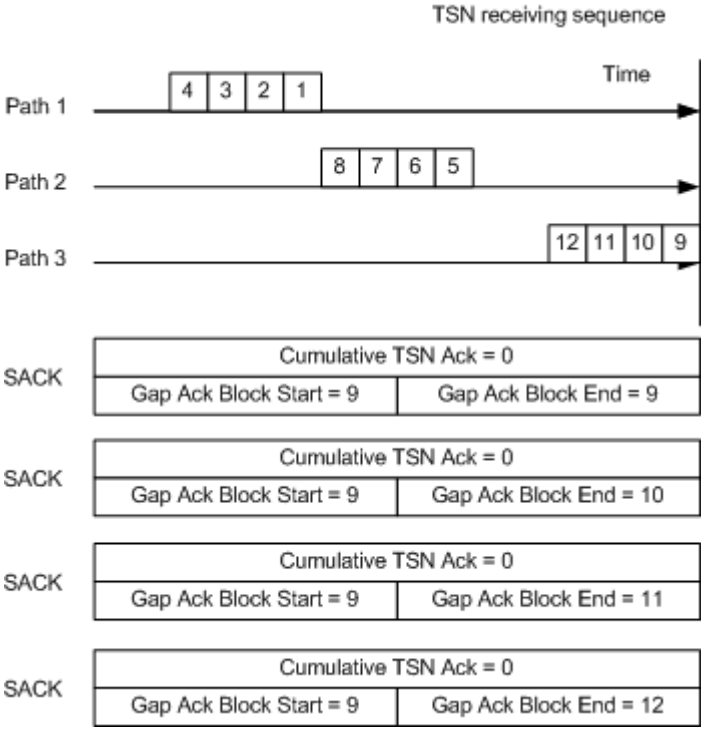

Figure 1. (b) Reordering during the transmission when multiple paths are used, 4 SACKs are triggered by TSN 9~12's reception.

The 3 paths have different qualities. During the transmission, TSN reordering occurs. Path 3 has a smaller delay than the other two paths and TSN 9~12 arrive at the receiver earliest. Here suppose each TSN is contained in a single SCTP packet. The receiver produces 4 SACKs with Gap Ack Blocks to the correspondent sender. Each SACK contains a hole ranging from TSN 1 to TSN 8 (Figure 1 (b)).

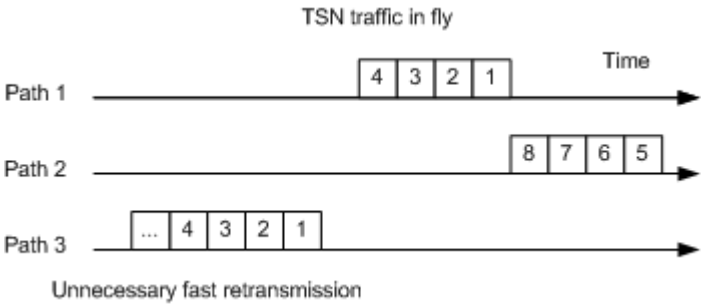

Figure 1. (c) The reordering of TSN causes the unnecessary fast retransmission, TSN 1~8 are falsely retransmitted by 4 SACKs' reception at the sender.

According to RFC2960, the 4 SACKs with Gap Ack Blocks increase the missing report counts of TSN 1~8 from 0 to 4, resulting in that the TSN 1~8 are marked as to be fast retransmitted (Figure 1 (c)). In fact, the TSNs are not lost, and they are just delayed. Such retransmissions are regarded as unnecessary.

3.2. LSFR Description

LSFR divides the transmitted chunks into different groups according to their transmitting paths. All the chunks transmitted on the same path belong to the same group. Refer to Figure 1 (a), TSN 1~4 belong to group 1, transmitted on path 1; TSN 5~8 belong to group 2, transmitted on path 2; TSN 9~12 belong to group 3, transmitted on path 3. The main idea in the LSFR algorithm is to maintain the state at the sender on a per-destination basis when a changeover happens. On the receipt of a SACK, the sender increases the missing report count for TSNs according to the state. In order to maintain the state that can be used for judging whether the missing report count to be increased, each path keeps a pending chunk queue that keeps the record of chunks transmitted on it. The record includes not only the chunks but also the necessary parameters dictated by RFC2960. LSFR is based on the following observation: the reordering observed during changeover happens while TSNs reaching the receiver are in-sequence within each group. Refer to Figure 1 (b) in section 3.1, the reordering happens inter-path, but not inner-path. On each path the TSNs are keeping sending sequence. While regarded from the whole sending sequence, they are reordered. From the example, the receiving sequence is: 9, 10, 11, 12, 5, 6, 7, 8, 1, 2, 3, 4.

Since the sequence is kept within each group, say, on each path, fast retransmit can be applied independently within each group. Refer to Figure 1 (b), the fast retransmit can be applied in Group 1 separately, where the scope is TSN 1~4, in Group 2 separately, where the scope is TSN 5~8, and in Group 3 separately, where the scope is TSN 9~12. This is called split fast retransmit.

In load sharing the status of the primary path is not significant. If there is no user assignment, the primary path has same chance as other ones in being selected for transmission. However, LSFR requires the sender keep the record of pending chunks in TSN sequence for each path. When the sender receives a SACK with Gap Ack Blocks, it will check each path with pending chunks on it. If there is new TSN acknowledged on the path, and there is hole in the GROUP (not the whole TSN sequence), the missing report count should be increased.

Consider the case in Figure 1 (b), as the SACK is not triggered by TSN transmitted on path 1 and 2, the pending chunks of the two paths do not have their missing report counts increased. There is another case described in Figure 2 (a).

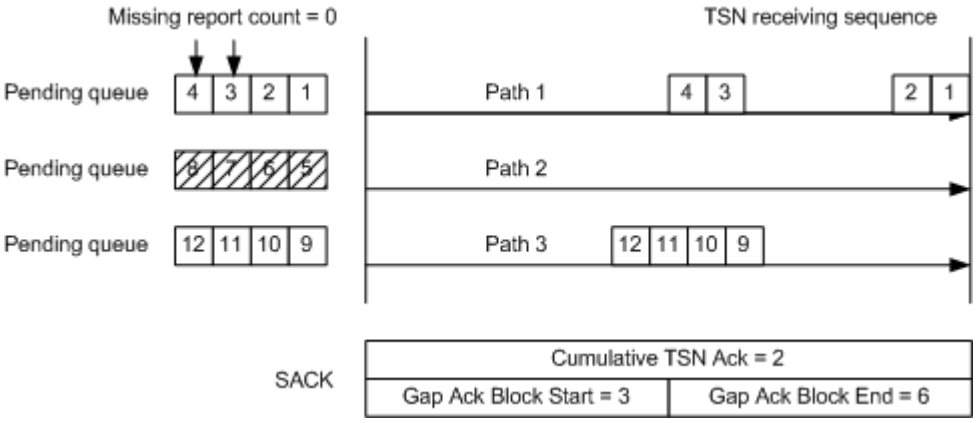

Figure 2. (a) A case that the missing report count should not be increased. TSN 5~8 have been acknowledged, while the reception of TSN 1 and 2 triggers SACK, which would increase missing report counts of TSN 3 and 4 according to original SCTP.

Look at the sending sequence in Figure 1 (a). The TSN 5~8 have been acknowledged before and are strip marked. At the time in Figure 2 (a), the receiver receives TSN 1 and 2. The SACK it sends to the sender still contains a gap. TSN 3 and 4 are in the hole of the whole TSN sequence. However, because there is no hole in Group 1, the missing report counts of TSN 3 and 4 should not be increased according to LSFR.

RFC2960 defines the circumstance where a SACK will be transmitted in no more than 200 ms after receiving an unacknowledged chunk. In LSFR, the SACK triggered by receiving chunks in other groups does not affect the missing report count of the chunk on a path. Figure 2 (b) gives the other circumstance where the missing report count is not increased in LSFR.

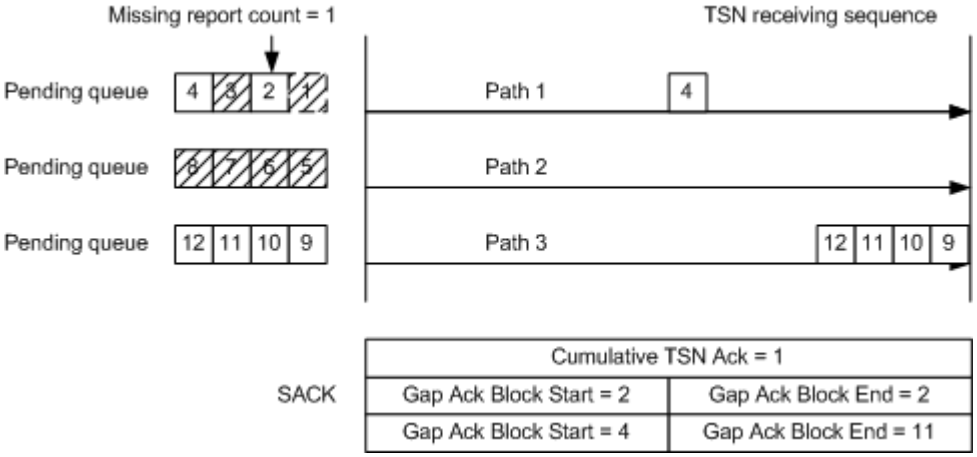

Figure 2. (b) Another case where the missing report count should not be increased. TSN 3, 5~8 have been acknowledged. TSN 2 is lost. The missing report counts of TSN 2 and 4 would be increased according to original SCTP.

Under the circumstance described by Figure 2 (b), the TSN 2 is really lost and it is really a hole in Group 1. The hole is presented in the coming SACK, which is triggered by the receiving of TSN 9~12. Because the SACK is not triggered by TSN in Group 1, the missing report count should not be increased and keeps the former value, 1. According to the circumstance described in Figure 2 (a), the missing report count of TSN 4 should not be increased either, because it is not a hole in Group 1. Here we can see only TSN greater than the hole and being in the same group with the hole can increase the missing report count of the hole. In any other cases the missing report count should not be increased.

4. SIMULATION RESULTS

Here are some typical simulation results of SCTP load sharing with LSFR. The simulation environment is NS-2.

4.1. Topology

Figure 3 shows the topology. We set N4 as the SCTP source endpoint, N1 as the sink. In order for comparison we set N0 as an UDP source to compete link sources with SCTP. N1 also serves as the UDP sink. We transfer files of different sizes using FTP that sits upon the

top of SCTP. UDP is employed to send CBR traffic as background.

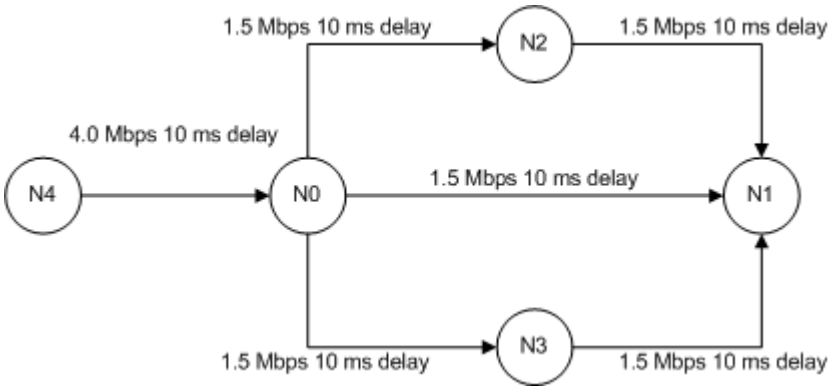

Figure 3. Simulation topology in NS-2

4.2. Performance Comparison of SCTP with and without Load Sharing

The results are of SCTP Load Sharing with LSFR enhanced. The time span is the whole time that is used for transferring the file data from beginning to the end. We also calculate the retransmission ratio of each transfer. The retransmission ratio reflects the transport efficiency of the transport. It is calculated as following:

Retransmission Ratio = Data size retransmitted / Data size

4.2.1. With no background traffic

Table 1

No.	PLR	Data size	Time Span with LS	Time Span without LS	Retransmission Ratio with LS	Retransmission Ratio without LS
1	0	1MB	2.999ms	6.842ms	0	0.023
2	0	5MB	14.434ms	28.176ms	0	0.005
3	0.001	1MB	2.726ms	7.219ms	0.003	0.010
4	0.001	5MB	10.950ms	30.397ms	0.002	0.013

(LS: Load Sharing; PLR: Packet Loss Rate; same in the next tables.)

As other transport protocol, such as TCP, in the phase of Congestion Control, some packets may encounter loss as the transmission rate keeps growing. As to SCTP Load Sharing, however, as all the eligible paths share the data load, the loss is not serious. (Table 1)

4.2.2. With UDP background traffic

Table 2

No.	PLR	Data size	Time Span with LS	Time Span without LS	Retransmission Ratio with LS	Retransmission Ratio without LS
5	0	1MB	4.329ms	11.152ms	0.002	0.047
6	0	5MB	18.695ms	61.630ms	0.001	0.009
7	0.001	1MB	4.014ms	10.734ms	0	0.038
8	0.001	5MB	18.825ms	53.464ms	0.006	0.009

In this case, there are CBR/UDP background-traffics on the links among N0, N1, N2, and N3. As the link source is stringent, SCTP Load Sharing still shows the good throughput and much less retransmissions. (Table 2)

Note: we didn't encounter retransmission in No.7 because this is a random case and the data size is not large.

4.3. Performance Comparison of SCTP load sharing with and without LSFR

The LSFR is the key of SCTP Load Sharing's success. It eliminates the side effects of simple load sharing on multiple paths, and gives a satisfying throughput. Table 3 shows the importance of LSFR in SCTP Load Sharing.

4.3.1. PLR (Packet Loss Rate) = 0, with no background traffic

Table 3

No.	Data Size	LS time span with LSFR	LS time span without LSFR	LS retransmission ratio with LSFR	LS retransmission ratio without LSFR
1	5MB	14.434ms	52.803ms	0.000	0.015
2	1MB	2.999ms	5.203ms	0.000	0.073
3	5MB	18.825ms	73.951ms	0.005	0.142
4	1MB	19.031	30.848	0.019	0.076

5. OTHER RELATED WORKS

SCTP Load Sharing is being intensively researched by the Internet society. Some IETF Draft gives discussion on how to implement it [5, 6]. A new Internet Draft, LS-SCTP [5], introduces two new types of chunk, LS-DATA and LS-SACK, and two new chunk parameters, Load-Sharing-Support for INIT/INIT-ACK and I-PSN for COOKIE-ECHO/COOKIE-ACK, to assist the load sharing. The communicating SCTP partners exchange path information at the association setup phase. The association maintains SCTP congestion control on each path chunk sequence defined by the LS-DATA. Though LS-SCTP applies the same principle as our approach, it requires both sides support the extensions. The Draft doesn't give any substantial implementation guide on how to maintain SCTP congestion control on each path. It needs further research.

6. CONCLUSION AND FUTURE WORKS

Our SCTP Load Sharing scheme has been proved an effective approach to improve SCTP association throughput. Load Sharing can utilize the network resources efficiently without bringing excess network congestion. It maintains SCTP congestion control on each eligible path separately. LSFR algorithm is the key point to implement the sender part Load Sharing. The approach does not require the SCTP receiver's extension at all. So, it has a good survivability and can be deployed conveniently.

SCTP Load Sharing may function as opening several TCP connections between the source and the destination. However, SCTP provides unified management for the connections to different network layer addresses. The overhead is less than maintaining multiple TCP connections at both sides simultaneously.

As SCTP Load Sharing is proved an efficient approach to improve SCTP throughput, it may be applied in other transmission fields. SCTP Load Sharing in mobile environments has been one of our research topics now. SCTP Load Sharing probably benefits in mobile transport, especially in soft handover phase between different access areas. Another possible field is how to associate the SCTP Load Sharing with the SCTP QoS. We expect PR-SCTP [7] Load Sharing may bring some attractive QoS features for multiple applications. That's a promising approach for future QoS warranted networks.

REFERENCES

1. R. Steward, et al, "Stream Control Transmission Protocol", Network Working Group of IETF, IETF RFC2960, Oct. 2000.
2. J. R. Iyengar, et al, "Preventing SCTP Congestion Window Overgrowth During Changeover", IETF Internet Draft, work in progress, Feb.22, 2002.
3. L. Coene, "Multirouting", IETF Internet Draft, work in progress, Feb.2002.
4. L. Coene, "Multihoming issues in the Stream Control Transmission Protocol", IETF Internet Draft, work in progress, Feb.2002.
5. A. Abd El Al, et al, "Load Sharing in Stream Control Transmission Protocol", IETF Internet Draft, work in progress, May 2003.
6. J. Kumar, et al, "Multihomed Loadsharing", IETF Internet Draft, work in progress, Feb. 2002.
7. R. Stewart, M. Ramalho, Stream Control Transmission Protocol (SCTP) Partial Reliability Extension, IETF RFC 3758, May 2004.
8. R. Stewart, et al, "Stream Control Transmission Protocol (SCTP) Implementers Guide", IETF Internet Draft, work in progress, Oct.1, 2002.