Enhancement of SCTP Throughput using Chunk Checksum

Cui Lin*, Seok J. Koh* and Yong-Geun Hong**
*Dept. of Computer Science, Kyungpook National University
**Protocol Engineering Center, ETRI

Abstract

Stream Control Transmission Protocol (SCTP) uses the 32-bit checksum in the common header, by which a corrupted SCTP packet will be regarded as a lost packet and then discarded. This may result in degradation of SCTP's throughput performance over wireless networks. This paper proposes a new chunk checksum scheme for SCTP, in which each data chunk contains its own checksum field and SACK chunk carry corresponding Transmission Sequence Number (TSN) with timestamp for every corruption event. The proposed chunk checksum scheme is introduced with the following three purposes: 1) to distinguish the chunk corruptions from the chunk losses; 2) to avoid the unnecessary halving of the congestion window (cwnd) in the case of chunk corruption; 3) to avoid the unwanted timeouts which can be induced in conventional SCTP in the case that the retransmitted data chunks are corrupted again in wireless networks. Simulation results show that the proposed chunk checksum scheme could improve the SCTP throughput in the wireless environments with a high bit error rate.

1. Introduction

The Stream Control Transmission Protocol (SCTP) is a new reliable transport layer protocol [1]. It is designed based on the window-based error and congestion control [2], as done in TCP. In SCTP, the corruption of a packet is regarded as a loss of the packet, which induces the retransmission request of the packet and the decrease of the congestion window at the sender side. This tends to result in the degradation of SCTP throughput performance, in particular, in the wireless networks with a high bit error rate.

It is noted that one SCTP packet which contains one or more data chunks carries only one checksum in its common header. On reception of an SCTP packet, if the checksum field indicates a corruption, the entire packet will be discarded. However, a corruption is different from the loss (or congestion), we may avoid adjustment of the congestion window in the corruption case, if we can determine that it is a corruption.

This paper proposes a data chunk-based checksum scheme which is purposed to extend the format of the SCTP data chunk by adding a 'header checksum'. The main purpose of the data chunk checksum is to handle the chunk corruption differently from the chunk loss for each data chunk, so that the congestion window is not decreased unnecessarily. In addition, this scheme is also purposed to prevent the SCTP retransmission timer from being expired by corruption.

The rest of this paper is organized as follows. Section 2 briefly reviews the existing related works on SCTP. In Section 3, we describe the proposed chunk checksum scheme.[1] Section 4 shows some simulation results for the proposed scheme using the ns-2 networks simulator. Section 5 concludes this paper.

This research was supported by the MIC, Korea, under the ITRC support program supervised by the IITA.

2. Related Works

Several works have been done to improve the SCTP throughput performance over wireless networks. The work in [3] introduces a MAC-Error-Warning (MEW) method with a new MAC packet type. The MEW method is similar to the Automatic Request (ARQ) mechanism [4], but a special MAC packet is generated at the MAC layer, before the data chunk is discarded, which contains some detailed information (including the stream ID and the sequence number) to notify the upper layer of the failed transmission. Arriving at the transport layer, the MEW packet will be analyzed by the SCTP source and all the useful information will be recovered so as to trigger fast retransmission of the data without degrading the congestion window.

The work in [5] utilizes the ECN [6] indications that will be marked by an ECN-capable router, which is used to detect the non-congestion packets loss. A packets loss with no ECN message in the same data window will be regarded as non-congestion errors, and then a simple loss-recovery procedure will be executed without applying the congestion control mechanism.

3. Proposed Chunk Checksum Scheme

3.1 Chunk-based Checksum

For compatibility with the standard SCTP, in the proposed chunk checksum scheme, the source and the destination will decide whether or not to use the proposed chunk-based checksum option in the association by exchanging the relevant information using the SCTP INIT and INIT-ACK chunks in the association establishment phase.

We assume in this paper that the SCTP endpoints agree to

use one additional 32-bit CRC checksum for each data chunk. The sender will construct and transmit each data chunk with the additional checksum, as shown in Figure 1.

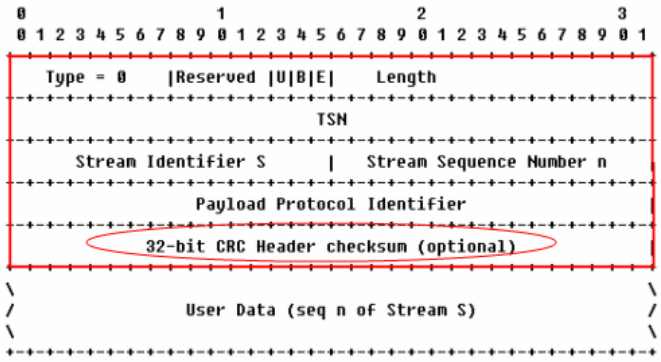

Fig.1. Data chunk format with 32-bit CRC checksum

As shown in the figure, we extend the format of SCTP data chunk by including the additional header checksum. At the sender side, the header checksum will be calculated by covering the first 20-byte header field of the data chunk (as indicated with the red lines in the figure). For ensuring the integrity of the packet's common header which contains port numbers and Verification Tag information, the header checksum in the first data chunk must be responsible for covering 12-byte packet's common header in addition to its own 20-byte header portion. With the additional header checksum, the receiver could recover useful information (e.g. TSN) from some, if not total, data chunks' headers which are not corrupted when the packet is corrupted.

3.2 Processing Procedure at Receiver Side

On reception of a SCTP packet, the receiver will first verify the integrity of the whole SCTP packet by checking the overall checksum in the common header. In case that the packet is corrupted, the receiver will then verify the integrity of each data chunk contained in the packet in turn by investigating the additional header checksums for each data chunk.

Once checking the header fails, if it is the first chunk, the receiver will discard this packet since the common header may contain wrong information; whereas if it is not the first one, the receiver only simply discards this data chunk and then continues to check the next one. The detailed checksum procedure is illustrated in Figure 2.

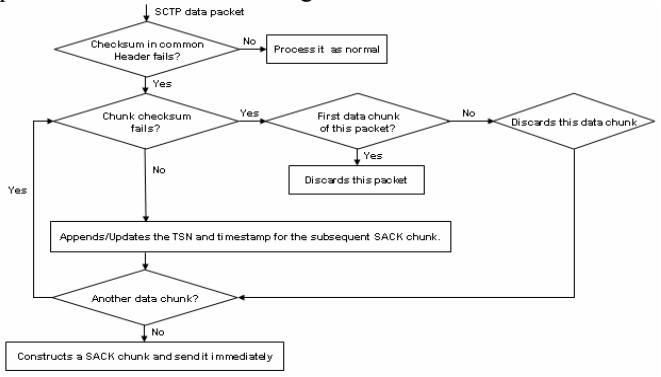

Fig.2. Procedures of the checksum checking by receiver

After inspecting the chunk checksum for each data chunk, the SCTP receiver will construct the subsequent SACK chunk, including corruption TSNs and corresponding timestamps. Herein, each timestamp indicates when the corresponding corruption event is detected at the receiver side. It is noted that each corruption event should be recorded in each subsequent SACK chunk for robustness. The detailed use of this timestamp will be described in the next section.

3.3 Retransmission Strategy by the Sender

Normally, the SCTP source uses the two different recovery schemes for a lost packet: namely timer-based retransmission and fast retransmission. In the proposed chunk checksum scheme, however, the 'corruption' event will be processed differently from the loss event, since the corruption itself indicates an explicit retransmission request.

By seeing the corruption TSN and corresponding timestamp enclosed in SACK chunk, the SCTP source can easily infer whether the corruption report is for a new corruption event. If the source determines that the corruption TSN and corresponding timestamp reports a new corruption event, it will retransmit the data chunk immediately. It is noted that the sender may retransmit the same data chunk multiple times before retransmission timer expires if it is corrupted repeatedly.

Figure 3 shows the procedure of processing a SACK chunk at the SCTP sender. In the figure, when the SCTP sender receives a SACK chunk, it first checks whether there are any corruption events' reports enclosed in this chunk. If not, the sender processes it as normal; If yes, it determines whether it is a report for a new corruption event or not. If it is a new corruption event's report, then the sender records the TSN and timestamp for retransmission. If not, the sender simply ignores it and continues to check the next corruption report.

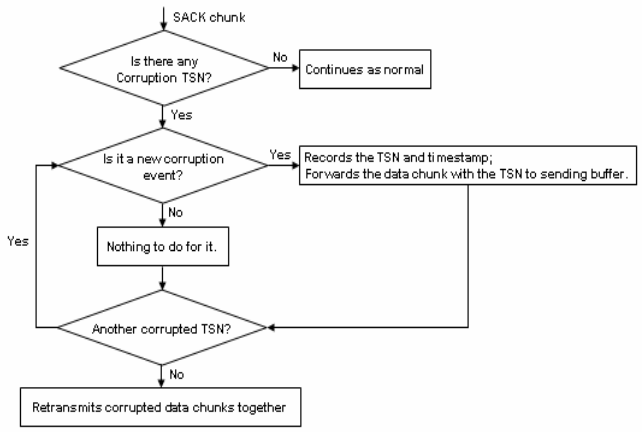

Fig.3. Flowchart of processing SACK chunk by sender

By means of the retransmission strategy, the proposed chunk checksum scheme can avoid unnecessary behaviors of both halving cwnd and unwanted timeouts which can be caused in conventional SCTP by corruption events and hence greatly improve the throughput performance of SCTP over wireless networks. The Figure 4 shows an example traced in our chunk checksum simulations which may illustrate the proposed retransmission strategy.

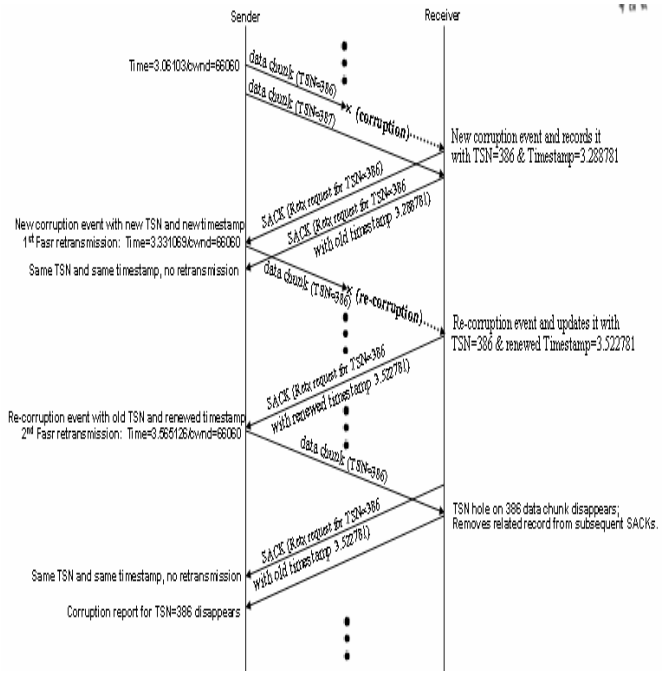

Fig.4. Sketch of retransmission strategy for proposal

4. Simulations

We evaluate the proposed chunk checksum scheme using the ns-2 network simulator (version 2.29) [7] over a simple test network, in which two endpoints communicate through either a single path (single-homing) or two non-overlapping paths (multi-homing), as shown in Figure 5.

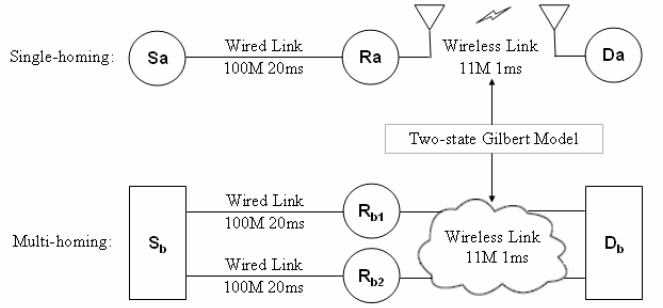

Fig.5. Simulation topology

In the figure, the end-to-end paths are in the wired-cum-wireless manner. Each wired link has the link bandwidth of 100 Mbps and the transmission delay of 20ms, whereas the wireless link has the bandwidth of 11 Mbps and the transmission delay of 1ms. For each experiment, we use the file transfer application using SCTP over the ns-2 simulator.

To emulate the corruptions, we employ Gilbert model [8] in the wireless environments, and we marked each corrupted packet with a corruption flag instead of dropping it in the experiments.

In the Gilbert model, two states of 'good' and 'bad' are expressed in terms of average error rates α and β, transition probabilities p and q, and average 'good' period of t1 seconds and 'bad' period of t2 seconds. If the link is in a 'good' state at present, it will continue to stay in the 'good' state with probability p, or transfer to the 'bad' state with a probability 1-p at the next instance. In the model, 'good' state means zero error probability. Also, if the link is in a 'bad' state at the current instance, then it will continue to stay in the 'bad' state with probability q, or transfer to the 'good' state with a probability 1-q at the next instance. When the link is in the 'bad' state, a SCTP packet experiences a packet corruption in the network with the probability β.

Table 1 summarizes the parameter values used for the simulations.

<Table 1> Parameter values used for experimentations

State	Average Period	Transition Probability	Packet corruption Rate
Good	t1=2.5 seconds	p=0.9, (1-p)=0.1	α= 0
Bad	t2=0.5 seconds	(1-q)=0.7, q=0.3	β =0.25

Furthermore, in receiver side, some corrupted packets are dropped intentionally by the proportion between header size (IP header + common header + data chunk header) and the size of user data payload in order to emulate the scenarios in which bit error occurs in header portion of SCTP packet.

For each ns-2 experiment in this paper, every packet size is 1500 bytes and only one data chunk is contained per packet. We performed a file transfer application for 100 seconds and compare the average throughputs (Kbytes/s) between the proposed scheme and standard SCTP for both the single-homing and multi-homing cases under the following various test scenarios:

4.1) for different packet corruption probability (β);
4.2) for different time period of two states (t1, t2);
4.3) for different packet loss rates as well as corruption.

4.1 Throughput for different corruption rates

Figure 6 shows the comparison of average throughput between chunk checksum scheme and standard SCTP for 100 seconds with the different corruption rates (β) from 0 to 0.5, given the other parameters as those in Table 1.

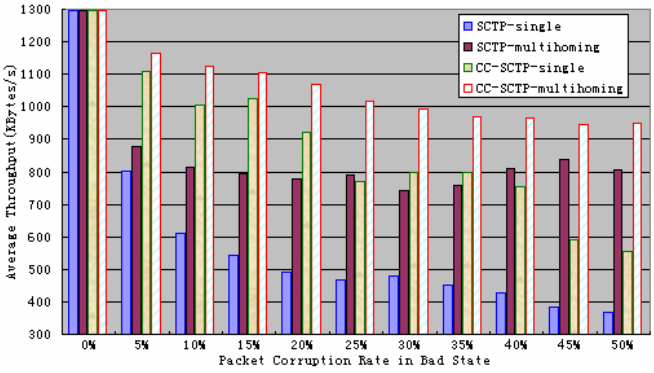

Fig.6. Average throughput for different β values

From the figure, we can see that the proposed chunk checksum scheme provides better throughputs over the standard SCTP scheme for both the single-homing and multi-homing cases. In particular, in the existing SCTP single-homing case, the average throughput of SCTP degrades drastically as the corruption rate (β) increases. On the contrary, the proposed chunk checksum scheme give better throughput, with the help of the retransmission strategy. This is because the proposed scheme can successfully avoid the behaviors of both halving the congestion window size incurred by fast retransmission due to corruption and slow start incurred by timeout due to corruption of retransmitted

packet.

By comparison of single-homing and multi-homing SCTP, the multi-homing SCTP provides better performance than the single-homing case, since the multi-homing SCTP can use the secondary path for retransmission of corrupted data chunks. Overall, the multi-homed chunk checksum scheme gives the best throughput over all the test cases.

4.2 Throughput for different time periods

Figure 7 shows the performance of chunk checksum scheme for different time periods of the two states: good (t1) and bad (t2). In the figure, given the base value of t1 = 2.5 second, the period of good state varies from 10s (4 x t1), 5s, 2.5s, 1.25s, 0.625s. 0.3125s, 0.15625s and 0.078125s, respectively, whereas given the base value of t2 = 0.5 second, the period of bad state varies from 2s (4 x t2), 1s, 0.5s, 0.25s, 0.125s, 0.0625s, 0.03125s and 0.015625s, respectively.

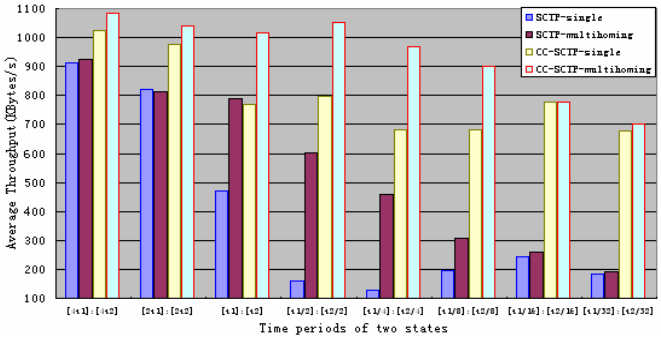

Fig.7. Average throughput for different time periods

From the figure, the results show that the standard SCTP is very sensitive to the time periods' variation of two states (t1 and t2), we can see that a shorter time period gives the worse performance in general. In the figure, it is noted that the proposed chunk checksum scheme outperform the existing SCTP in both single-homing and multi-homing cases. This is also because the corruption does not induce the shrink of the congestion window size in the proposed scheme.

4.3 Throughput of SCTP with packet losses

Furthermore, we compare the throughputs of chunk checksum scheme and standard SCTP with some packet 'losses' in addition to the packet corruptions. We apply a uniform error (loss) model for the primary wired-link to generate the packet losses, with the packet loss rate changed from 0 to 0.1.

Figure 8 shows the throughputs of the SCTP in the networks with the packet loss rates as well as the packet corruptions.

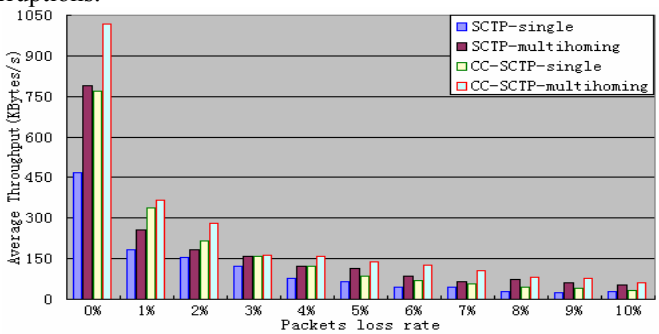

Fig.8. Average throughput with different packet loss rates

In the figure, we can see that the proposed chunk checksum scheme still provides the better throughputs than the standard SCTP for both the single-homing and multi-homing cases. In particular, when the packet loss rate is less than 2%, the proposed CC-SCTP scheme gives the performance gains of more than 40.1% and 43.9% respectively, compared to the standard SCTP.

5. Conclusions

In this paper, we propose the data chunk-based checksum scheme, which is designed to enhance the SCTP throughput even in the wireless network environments with a high bit error (or corruption) rate. From the simulation results, we can see that the proposed chunk checksum scheme provides far better throughput than the standard SCTP for both the single-homing and multi-homing cases in the networks with a high bit error or corruption rates.

It is noted that the performance gain of the proposed chunk checksum scheme comes from the following features: 1) the proposed scheme can distinguish the chunk corruptions from the chunk losses by using additional data chunk checksum; 2) hence, the chunk checksum scheme can avoid unnecessary halving of the congestion window in the case of packet corruption; 3) in particular, by enclosing the corruption TSN and corresponding timestamp in SACK chunk, the chunk checksum scheme may exploit a robust retransmission strategy to avoid the unwanted timeouts which can be induced in conventional SCTP in the case that the retransmitted data chunks are corrupted in wireless environments.

References

[1] R. Stewart et al., "Stream control transmission protocol," IETF, RFC 2960, Oct. 2000.

[2] M. Allman et al., "TCP congestion control," IETF, RFC 2581, Apr. 1999.

[3] D. Wang; S. Yang; W. Sun; "A Mac-Error-Warning Method for SCTP Congestion Control over High BER Wireless Network", Wireless Communications, Networking and Mobile Computing, 2005. Proceedings. 2005 International Conference on Volume 1, Sept. 23-26, 2005 Page(s):513 - 516.

[4] S. Lin, C1.J. Costello, and M.J. Miller, "Automatic-repeat-request errorcontrol schemes," IEEE Communicutions Muguzine, vol. 22, pp. 5-17, December 1984.

[5] G. Ye; S., T.N.; M. J Lee; "Improving Stream Control Transmission Protocol Performance Over Lossy Links", Selected Areas in Communications, IEEE Journal on Volume 22, Issue 4, May 2004 Page(s):727 – 736.

[6] K. Ramakrishnan et al., "The addition of explicit congestion notification (ECN) to IP," IETF, RFC 3168, Sept. 2001.

[7] Network Simulator (ns-2), available from http://www.isi.edu/nsnam/ns/.

[8] E. N. Gilbert, "Capacity of a burst-noise channel," Bell Systems Technical Journal, vol. 39, pp. 1253-1265,Sept. 1960.