A Simple Method to Overcome the Restriction of the SACK Blocks' Number in SACK TCP

Cui Lin and Choong Seon Hong
School of Electronic and Information of Kyung Hee University
cuilin@networking.khu.ac.kr, cshong@khu.ac.kr

Abstract

By definition of RFC 2018, each segments block of data queued at the data receiver is defined in the SACK option by two 32-bit unsigned integers in network byte order. Since TCP Options field has a 40-byte maximum length, when error bursts occur, we note that the limitation of maximum available option space may not be sufficient to report all blocks present in the receiver's queue and lead to unnecessarily force the TCP sender to retransmit packets that have actually been received but not carried related information in SACK option field. For overcoming this restriction, in this paper, a new solution is designed to further improve the performance of TCP SACK and prevent those unwanted retransmissions. Simulation result shows that the implementation of our proposal is effective.

1. Introduction

Multiple packet losses from a data window can have a catastrophic effect on TCP throughput. TCP uses a cumulative acknowledgment scheme and only can retransmits just one dropped packet per round-trip time, when multiple packets are lost from a window, this forces the sender to either wait a round trip time to find out about each lost packet, or unnecessarily retransmit segments which have been successfully received.

SACK is a strategy which corrects this behavior in the face of multiple dropped segments. With selective acknowledgments, the data receiver can inform the sender about all segments that have arrived successfully, thus the sender only need to retransmit the segments that have actually been lost. For this purpose, a selective acknowledgment (SACK) mechanism was defined in RFC 2018 [1].

However, in implementation of SACK TCP, each segments block of data queued at the data receiver is defined in the SACK option by two 32-bit unsigned integers as two edges' sequence number. Because TCP Options field has a length limit of 40 bytes, so the 40 bytes available for TCP options can specify a maximum of 4 blocks. Also, It is expected that SACK will often be used in conjunction with the Timestamp option used for RTTM [2], which takes an additional 10 bytes (plus two bytes of padding); thus a maximum of 3 SACK blocks will be allowed in this case (actually, it is usually the case and 3 is default value in ns2).

This restriction can, under error bursts' environment, cause unnecessary retransmission because of no enough space to convey all information about received segments. Thereby it will bring unnecessary shrinkage of TCP congestion window and degrade the performance of TCP connections.

In this paper, we propose a new solution to overcome this limitation. By implementation of our proposal, we can convey all information about lost segments with least bytes no matter how poor the environment is.

The rest of this paper is organized as follows: We introduce related works in Section II, present our proposal in Section III and simulation result in Section IV, finally we conclude in Section V.

2. Current SACK Option Format

The current SACK option format as defined in RFC 2018 [1] is as shown in Fig. 1. Its limitation has described in section I.

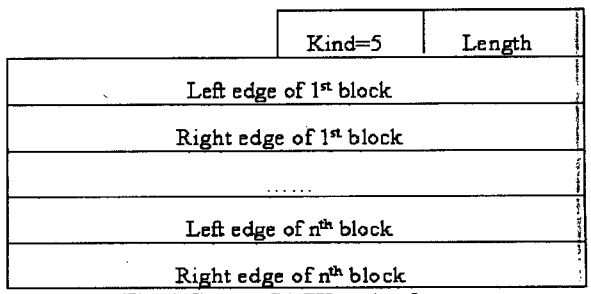

Fig. 1 Current SACK option format

This work was supported by MIC

3. Our Proposal

Since TCP does not change the maximum segment size (MSS) once the TCP connection is established, and all segments preceding the last one has MSS as their size, we can uniquely denote a segment by its left edge's sequence number, in other words, you can certainly know a segment's sequence number range as long as you know its left edge's sequence number, and so do we in this paper. Therefore, we well use this characteristic and present a very effective SACK mechanism. By implementation of our proposal, the sender can be able to effectively retransmit all segments which have actually been lost together with new segments in a burst of data sending and avoid any unnecessary retransmission. To carry it out, refer to Fig.2, in our proposal, we represent lost segments information by segment order's offset instead of 32-bit absolute sequence number. Thus, in order to let the sender wake up to whole gaps information in receiver's sequence space, since the sender has known the size of segment, and acknowledgment number field in ACK's TCP header has specified first gap's left edge by 32-bit absolute sequence number, the receiver just need to send other gaps information in sequence space. So, the following offset fields named offset1, offset2,…offsetn, only need respectively to represent the offset with respect to the segment specified by acknowledgment number field or previous offset field in segment order. Out of question, by 32-bit sequence number of first gap's left edge (acknowledge number field), size of segment and offsets in segment order, the sender can easily be aware of all 32-bit sequence numbers of all gaps in sequence space.

As shown in above Fig. 2, in our proposed SACK option format, each offset field consists of 1 byte. Wherein, first bit is multiple segments' gap flag (denoted it by M-flag in Fig.3), and other 7 bits represent real offset, 7 bits can represent up to 27-1=127. We think 7 bits are enough (even if each offset consist of 2 bytes, our proposal is still more effective and using least bytes). As for the interpretation of multiple segments' gap flag, "0" indicates that this offset corresponds to first segment of a gap no matter it consists of a single lost segment or multiple lost segments; but "1" just indicates that this offset corresponds to last segment of a gap consists of multiple successively lost segments. Obviously, environment condition becomes worse, the multiple segments' gap flag is more effective.

To understand more easily, we illustrate the proposed mechanism in Fig. 3. In this scenario, the sender can grasp lost segments' information by only 5 bytes, under the same scenario, for getting the same result, current SACK mechanism needs 26 bytes (2+8*3=26).

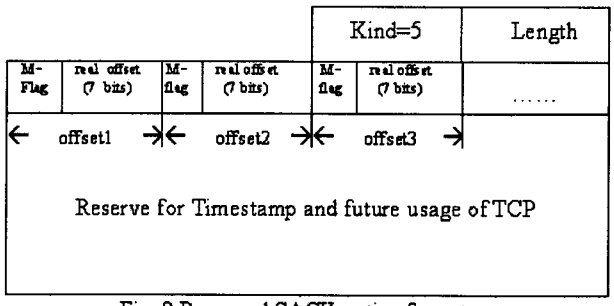

Fig. 2 Proposed SACK option format

Received Segment	ACK packet: Ack. Num	SACK option field: Offset1	Offset2	Offset3	How did the sender determine which segments have been lost?
5000	5500				5500 (Normal ACK)
5500 (Lost.)					
6000	5500				5500
6500 (Lost.)					
7000	5500	2			5500, 5500+500*2=6500
7500 (Lost.)					
8000 (Lost.)					
8500 (Lost.)					
9000 (Lost.)					
9500 (Lost.)					
10000	5500	2 (<128, next m_flag =0, single)	2 (<128, next m_flag =1, cont)	132 (>128, multiple)	A=5500, B=A+500*2=6500, C=B+500*2=7500, D=C+500*(132-128)=9500. because D>128, so lost: 5500,6500, 7500,8000,8500, 9000,9500

Fig. 3 Proposed SACK mechanism

By our proposal, n lost segments will need a length of (n+1) bytes at most, and 40 bytes are available for more than 39 (or 29 with timestamp) lost segments. Actually, it is clear that we do not need to represent so many lost segments' information by one ACK packet. Therefore, we can minimize the payload of ACK packet and save enough space to future usage of TCP. In one word, due to optimized SACK option's mechanism, the improvement of TCP performance and energy efficiency, all become possible.

4. Simulation

We did simulation in NS2 and observed the transmission of packets through the network. We used multiple error model in order to product error burst effect in 10M link and 2M link respectively, and the main parameters in the error model are listed in the following:

```
set tmp [new ErrorModel/Uniform 0 pkt]
set tmp1 [new ErrorModel/Uniform .05 pkt]
set m_periods [list 0.5 .0375]
set m_transmx { {0.95 0.05} {1 0} }
set m_trunit pkt
set m_sttype pkt
set m_nstates 2
set m_nstart [lindex $m_states 0]
```

By comparison between Fig.4 and Fig.5, we noted that packets with packet numbers 1314,1316 and 1317 are unnecessarily retransmitted because of the limitation of SACK in implementation of standard SACK TCP, while those packets are not retransmitted in implementation of proposed SACK TCP as shown in Fig5. This shows clearly that the proposed SACK does perform better than the present implementation.

Fig.6 shows that when TCP connections are established over a long period of the error condition environments, the proposed SACK performs far better than the existing implementation. We see that the new implementation of proposed SACK can avoid some unnecessary degradation (see the case at 60 sec) of performance and always gives a better throughput than another one.

5. Conclusion

The limitation of current SACK implementation indicates that it unfit to handle scenarios under error bursts' environment. In this paper, our proposed representation for TCP SACK can primely overcome this problem. Simulation results also showed that the implementation of our proposal for SACK is very effective by means of the least bytes and most robust mechanism in the high packet error rates' scenarios.

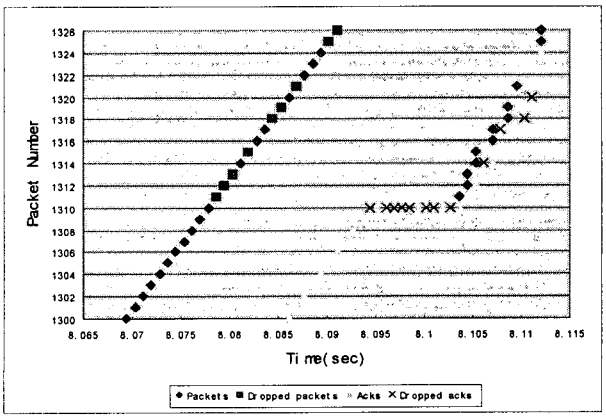

Fig. 4 Performance of Standard SACK TCP

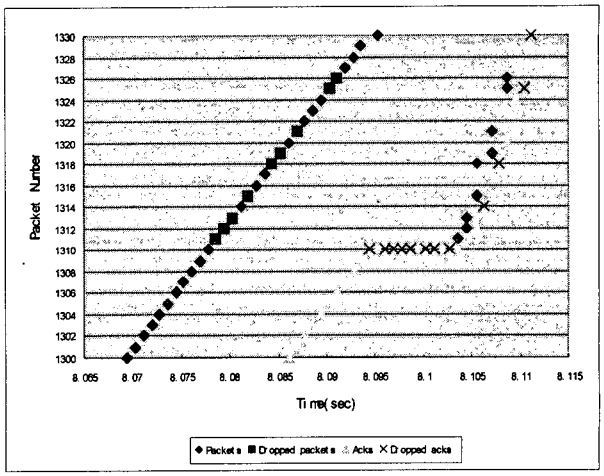

Fig.5 Performance of Proposed SACK TCP

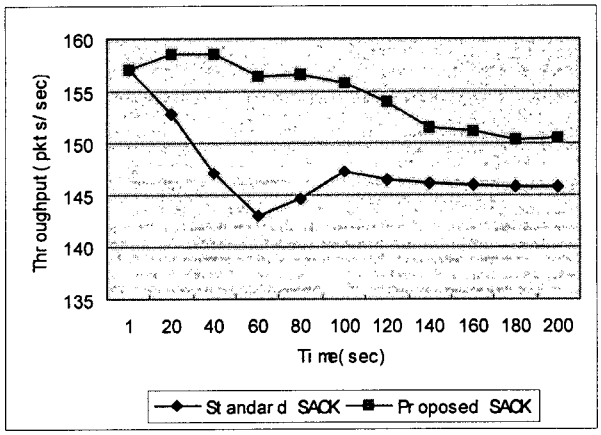

Fig. 6 Throughput vs. Time

References

[1] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, "TCP selective acknowledgment and options", RFC 2018, IETF, October 1996.

[2] V. Jacobson, R. Braden, D. Borman, "TCP Extensions for High Performance", RFC 1323, IETF, May 1992.

[3] S. Floyd, J. Mahdavi, M. Mathis, M. Podolsky, "An Extension to the Selective Acknowledgement (SACK) Option for TCP", RFC 2883, IETF, July 2000.

[4] K. Fall, S. Floyd, "Simulation based Comparisons of Tahoe, Reno, and SACK TCP", Computer Communications Review, vol.26, pp 5-21, 1996.

[5] M. Allman, D.Glover, NASA Lewis, L. Sanchez, "Enhancing TCP Over Satellite Channels Using Standard Mechanisms", RFC 2488, IETF, January 1999.

[6] S. Floyd, "Issues with TCP SACK", Technical Report, LBL Network Group, 1996.

[7] A. Romanow, S. Floyd, "Dynamics of TCP traffic over ATM Networks", IEEE Journal on Selected Areas in Communications, Vol. 13 No. 4, p 633-641, 1995.

[8] K.N. Srijith, Lillykutty Jacob, A.L. Ananda, "Worst-Case Performance Limitation of TCP SACK and a Feasible Solution", IEEE Journal on Selected Areas in Communications., 2002.