Network Working Group
Request for Comments: 2883
Category: Standards Track

S. Floyd
ACIRI
J. Mahdavi
Novell
M. Mathis
Pittsburgh Supercomputing Center
M. Podolsky
UC Berkeley

July 2000

An Extension to the Selective Acknowledgement (SACK) Option for TCP

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

This note defines an extension of the Selective Acknowledgement (SACK) Option [RFC2018] for TCP. RFC 2018 specified the use of the SACK option for acknowledging out-of-sequence data not covered by TCP's cumulative acknowledgement field. This note extends RFC 2018 by specifying the use of the SACK option for acknowledging duplicate packets. This note suggests that when duplicate packets are received, the first block of the SACK option field can be used to report the sequence numbers of the packet that triggered the acknowledgement. This extension to the SACK option allows the TCP sender to infer the order of packets received at the receiver, allowing the sender to infer when it has unnecessarily retransmitted a packet. A TCP sender could then use this information for more robust operation in an environment of reordered packets [BPS99], ACK loss, packet replication, and/or early retransmit timeouts.

1. Conventions and Acronyms

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in [B97].

2. Introduction

The Selective Acknowledgement (SACK) option defined in RFC 2018 is used by the TCP data receiver to acknowledge non-contiguous blocks of data not covered by the Cumulative Acknowledgement field. However, RFC 2018 does not specify the use of the SACK option when duplicate segments are received. This note specifies the use of the SACK option when acknowledging the receipt of a duplicate packet [F99]. We use the term D-SACK (for duplicate-SACK) to refer to a SACK block that reports a duplicate segment.

This document does not make any changes to TCP's use of the cumulative acknowledgement field, or to the TCP receiver's decision of *when* to send an acknowledgement packet. This document only concerns the contents of the SACK option when an acknowledgement is sent.

This extension is compatible with current implementations of the SACK option in TCP. That is, if one of the TCP end-nodes does not implement this D-SACK extension and the other TCP end-node does, we believe that this use of the D-SACK extension by one of the end nodes will not introduce problems.

The use of D-SACK does not require separate negotiation between a TCP sender and receiver that have already negotiated SACK capability. The absence of separate negotiation for D-SACK means that the TCP receiver could send D-SACK blocks when the TCP sender does not understand this extension to SACK. In this case, the TCP sender will simply discard any D-SACK blocks, and process the other SACK blocks in the SACK option field as it normally would.

3. The Sack Option Format as defined in RFC 2018

The SACK option as defined in RFC 2018 is as follows:

```
                         +--------+--------+
                         | Kind=5 | Length |
       +--------+--------+--------+--------+
       |      Left Edge of 1st Block       |
       +--------+--------+--------+--------+
       |      Right Edge of 1st Block      |
       +--------+--------+--------+--------+
       |                                   |
       /            . . .                  /
       |                                   |
       +--------+--------+--------+--------+
       |      Left Edge of nth Block       |
       +--------+--------+--------+--------+
       |      Right Edge of nth Block      |
       +--------+--------+--------+--------+
```

The Selective Acknowledgement (SACK) option in the TCP header contains a number of SACK blocks, where each block specifies the left and right edge of a block of data received at the TCP receiver. In particular, a block represents a contiguous sequence space of data received and queued at the receiver, where the "left edge" of the block is the first sequence number of the block, and the "right edge" is the sequence number immediately following the last sequence number of the block.

RFC 2018 implies that the first SACK block specify the segment that triggered the acknowledgement. From RFC 2018, when the data receiver chooses to send a SACK option, "the first SACK block ... MUST specify the contiguous block of data containing the segment which triggered this ACK, unless that segment advanced the Acknowledgment Number field in the header."

However, RFC 2018 does not address the use of the SACK option when acknowledging a duplicate segment. For example, RFC 2018 specifies that "each block represents received bytes of data that are contiguous and isolated". RFC 2018 further specifies that "if sent at all, SACK options SHOULD be included in all ACKs which do not ACK the highest sequence number in the data receiver's queue." RFC 2018 does not specify the use of the SACK option when a duplicate segment is received, and the cumulative acknowledgement field in the ACK acknowledges all of the data in the data receiver's queue.

4. Use of the SACK option for reporting a duplicate segment

This section specifies the use of SACK blocks when the SACK option is used in reporting a duplicate segment. When D-SACK is used, the first block of the SACK option should be a D-SACK block specifying the sequence numbers for the duplicate segment that triggers the acknowledgement. If the duplicate segment is part of a larger block of non-contiguous data in the receiver's data queue, then the following SACK block should be used to specify this larger block. Additional SACK blocks can be used to specify additional non-contiguous blocks of data, as specified in RFC 2018.

The guidelines for reporting duplicate segments are summarized below:

(1) A D-SACK block is only used to report a duplicate contiguous sequence of data received by the receiver in the most recent packet.

(2) Each duplicate contiguous sequence of data received is reported in at most one D-SACK block. (I.e., the receiver sends two identical D-SACK blocks in subsequent packets only if the receiver receives two duplicate segments.)

(3) The left edge of the D-SACK block specifies the first sequence number of the duplicate contiguous sequence, and the right edge of the D-SACK block specifies the sequence number immediately following the last sequence in the duplicate contiguous sequence.

(4) If the D-SACK block reports a duplicate contiguous sequence from a (possibly larger) block of data in the receiver's data queue above the cumulative acknowledgement, then the second SACK block in that SACK option should specify that (possibly larger) block of data.

(5) Following the SACK blocks described above for reporting duplicate segments, additional SACK blocks can be used for reporting additional blocks of data, as specified in RFC 2018.

Note that because each duplicate segment is reported in only one ACK packet, information about that duplicate segment will be lost if that ACK packet is dropped in the network.

4.1 Reporting Full Duplicate Segments

We illustrate these guidelines with three examples. In each example, we assume that the data receiver has first received eight segments of 500 bytes each, and has sent an acknowledgement with the cumulative acknowledgement field set to 4000 (assuming the first sequence number is zero). The D-SACK block is underlined in each example.

4.1.1. Example 1: Reporting a duplicate segment.

Because several ACK packets are lost, the data sender retransmits packet 3000-3499, and the data receiver subsequently receives a duplicate segment with sequence numbers 3000-3499. The receiver sends an acknowledgement with the cumulative acknowledgement field set to 4000, and the first, D-SACK block specifying sequence numbers 3000-3500.

```
      Transmitted    Received    ACK Sent
      Segment        Segment     (Including SACK Blocks)

      3000-3499      3000-3499   3500 (ACK dropped)
      3500-3999      3500-3999   4000 (ACK dropped)
      3000-3499      3000-3499   4000, SACK=3000-3500
                                            ---------
```

4.1.2. Example 2: Reporting an out-of-order segment and a duplicate segment.

Following a lost data packet, the receiver receives an out-of-order data segment, which triggers the SACK option as specified in RFC 2018. Because of several lost ACK packets, the sender then retransmits a data packet. The receiver receives the duplicate packet, and reports it in the first, D-SACK block:

```
      Transmitted    Received    ACK Sent
      Segment        Segment     (Including SACK Blocks)

      3000-3499      3000-3499   3500 (ACK dropped)
      3500-3999      3500-3999   4000 (ACK dropped)
      4000-4499      (data packet dropped)
      4500-4999      4500-4999   4000, SACK=4500-5000 (ACK dropped)
      3000-3499      3000-3499   4000, SACK=3000-3500, 4500-5000
                                            ---------
```

4.1.3. Example 3: Reporting a duplicate of an out-of-order segment.

Because of a lost data packet, the receiver receives two out-of-order segments. The receiver next receives a duplicate segment for one of these out-of-order segments:

```
Transmitted    Received    ACK Sent
Segment        Segment     (Including SACK Blocks)

3500-3999      3500-3999   4000
4000-4499      (data packet dropped)
4500-4999      4500-4999   4000, SACK=4500-5000
5000-5499      5000-5499   4000, SACK=4500-5500
               (duplicated packet)
               5000-5499   4000, SACK=5000-5500, 4500-5500
                                ---------
```

4.2. Reporting Partial Duplicate Segments

It may be possible that a sender transmits a packet that includes one or more duplicate sub-segments--that is, only part but not all of the transmitted packet has already arrived at the receiver. This can occur when the size of the sender's transmitted segments increases, which can occur when the PMTU increases in the middle of a TCP session, for example. The guidelines in Section 4 above apply to reporting partial as well as full duplicate segments. This section gives examples of these guidelines when reporting partial duplicate segments.

When the SACK option is used for reporting partial duplicate segments, the first D-SACK block reports the first duplicate sub-segment. If the data packet being acknowledged contains multiple partial duplicate sub-segments, then only the first such duplicate sub-segment is reported in the SACK option. We illustrate this with the examples below.

4.2.1. Example 4: Reporting a single duplicate subsegment.

The sender increases the packet size from 500 bytes to 1000 bytes. The receiver subsequently receives a 1000-byte packet containing one 500-byte subsegment that has already been received and one which has not. The receiver reports only the already received subsegment using a single D-SACK block.

```
        Transmitted    Received    ACK Sent
        Segment        Segment     (Including SACK Blocks)

        500-999        500-999     1000
        1000-1499      (delayed)
        1500-1999      (data packet dropped)
        2000-2499      2000-2499   1000, SACK=2000-2500
        1000-2000      1000-1499   1500, SACK=2000-2500
                       1000-2000   2500, SACK=1000-1500
                                               ---------
```

4.2.2. Example 5: Two non-contiguous duplicate subsegments covered by the cumulative acknowledgement.

After the sender increases its packet size from 500 bytes to 1500 bytes, the receiver receives a packet containing two non-contiguous duplicate 500-byte subsegments which are less than the cumulative acknowledgement field. The receiver reports the first such duplicate segment in a single D-SACK block.

```
         Transmitted    Received    ACK Sent
         Segment        Segment     (Including SACK Blocks)

         500-999        500-999     1000
         1000-1499      (delayed)
         1500-1999      (data packet dropped)
         2000-2499      (delayed)
         2500-2999      (data packet dropped)
         3000-3499      3000-3499   1000, SACK=3000-3500
         1000-2499      1000-1499   1500, SACK=3000-3500
                        2000-2499   1500, SACK=2000-2500, 3000-3500
                        1000-2499   2500, SACK=1000-1500, 3000-3500
                                                ---------
```

4.2.3. Example 6: Two non-contiguous duplicate subsegments not covered by the cumulative acknowledgement.

This example is similar to Example 5, except that after the sender increases the packet size, the receiver receives a packet containing two non-contiguous duplicate subsegments which are above the cumulative acknowledgement field, rather than below. The first, D-SACK block reports the first duplicate subsegment, and the second, SACK block reports the larger block of non-contiguous data that it belongs to.

```
        Transmitted    Received    ACK Sent
        Segment        Segment     (Including SACK Blocks)

        500-999        500-999     1000
        1000-1499      (data packet dropped)
        1500-1999      (delayed)
        2000-2499      (data packet dropped)
        2500-2999      (delayed)
        3000-3499      (data packet dropped)
        3500-3999      3500-3999   1000, SACK=3500-4000
        1000-1499      (data packet dropped)
        1500-2999      1500-1999   1000, SACK=1500-2000, 3500-4000
                       2000-2499   1000, SACK=2000-2500, 1500-2000,
                                           3500-4000
                       1500-2999   1000, SACK=1500-2000, 1500-3000,
                                              ---------
                                           3500-4000
```

4.3. Interaction Between D-SACK and PAWS

RFC 1323 [RFC1323] specifies an algorithm for Protection Against Wrapped Sequence Numbers (PAWS). PAWS gives a method for distinguishing between sequence numbers for new data, and sequence numbers from a previous cycle through the sequence number space. Duplicate segments might be detected by PAWS as belonging to a previous cycle through the sequence number space.

RFC 1323 specifies that for such packets, the receiver should do the following:

> Send an acknowledgement in reply as specified in RFC 793 page 69, and drop the segment.

Since PAWS still requires sending an ACK, there is no harmful interaction between PAWS and the use of D-SACK. The D-SACK block can be included in the SACK option of the ACK, as outlined in Section 4, independently of the use of PAWS by the TCP receiver, and independently of the determination by PAWS of the validity or invalidity of the data segment.

TCP senders receiving D-SACK blocks should be aware that a segment reported as a duplicate segment could possibly have been from a prior cycle through the sequence number space. This is independent of the use of PAWS by the TCP data receiver. We do not anticipate that this will present significant problems for senders using D-SACK information.

5. Detection of Duplicate Packets

This extension to the SACK option enables the receiver to accurately report the reception of duplicate data. Because each receipt of a duplicate packet is reported in only one ACK packet, the loss of a single ACK can prevent this information from reaching the sender. In addition, we note that the sender can not necessarily trust the receiver to send it accurate information [SCWA99].

In order for the sender to check that the first (D)SACK block of an acknowledgement in fact acknowledges duplicate data, the sender should compare the sequence space in the first SACK block to the cumulative ACK which is carried IN THE SAME PACKET. If the SACK sequence space is less than this cumulative ACK, it is an indication that the segment identified by the SACK block has been received more than once by the receiver. An implementation MUST NOT compare the sequence space in the SACK block to the TCP state variable snd.una (which carries the total cumulative ACK), as this may result in the wrong conclusion if ACK packets are reordered.

If the sequence space in the first SACK block is greater than the cumulative ACK, then the sender next compares the sequence space in the first SACK block with the sequence space in the second SACK block, if there is one. This comparison can determine if the first SACK block is reporting duplicate data that lies above the cumulative ACK.

TCP implementations which follow RFC 2581 [RFC2581] could see duplicate packets in each of the following four situations. This document does not specify what action a TCP implementation should take in these cases. The extension to the SACK option simply enables the sender to detect each of these cases. Note that these four conditions are not an exhaustive list of possible cases for duplicate packets, but are representative of the most common/likely cases. Subsequent documents will describe experimental proposals for sender responses to the detection of unnecessary retransmits due to reordering, lost ACKS, or early retransmit timeouts.

5.1. Replication by the network

If a packet is replicated in the network, this extension to the SACK option can identify this. For example:

```
Transmitted    Received     ACK Sent
Segment        Segment      (Including SACK Blocks)

500-999        500-999      1000
1000-1499      1000-1499    1500
               (replicated)
               1000-1499    1500, SACK=1000-1500
                                       ---------
```

In this case, the second packet was replicated in the network. An ACK containing a D-SACK block which is lower than its ACK field and is not identical to a previously retransmitted segment is indicative of a replication by the network.

WITHOUT D-SACK:

If D-SACK was not used and the last ACK was piggybacked on a data packet, the sender would not know that a packet had been replicated in the network. If D-SACK was not used and neither of the last two ACKs was piggybacked on a data packet, then the sender could reasonably infer that either some data packet *or* the final ACK packet had been replicated in the network. The receipt of the D-SACK packet gives the sender positive knowledge that this data packet was replicated in the network (assuming that the receiver is not lying).

RESEARCH ISSUES:

The current SACK option already allows the sender to identify duplicate ACKs that do not acknowledge new data, but the D-SACK option gives the sender a stronger basis for inferring that a duplicate ACK does not acknowledge new data. The knowledge that a duplicate ACK does not acknowledge new data allows the sender to refrain from using that duplicate ACKs to infer packet loss (e.g., Fast Retransmit) or to send more data (e.g., Fast Recovery).

5.2. False retransmit due to reordering

If packets are reordered in the network such that a segment arrives more than 3 packets out of order, TCP's Fast Retransmit algorithm will retransmit the out-of-order packet. An example of this is shown below:

```
Transmitted    Received    ACK Sent
Segment        Segment     (Including SACK Blocks)

500-999        500-999     1000
1000-1499      (delayed)
1500-1999      1500-1999   1000, SACK=1500-2000
2000-2499      2000-2499   1000, SACK=1500-2500
2500-2999      2500-2999   1000, SACK=1500-3000
1000-1499      1000-1499   3000
               1000-1499   3000, SACK=1000-1500
                                      ---------
```

In this case, an ACK containing a SACK block which is lower than its ACK field and identical to a previously retransmitted segment is indicative of a significant reordering followed by a false (unnecessary) retransmission.

WITHOUT D-SACK:

With the use of D-SACK illustrated above, the sender knows that either the first transmission of segment 1000-1499 was delayed in the network, or the first transmission of segment 1000-1499 was dropped and the second transmission of segment 1000-1499 was duplicated. Given that no other segments have been duplicated in the network, this second option can be considered unlikely.

Without the use of D-SACK, the sender would only know that either the first transmission of segment 1000-1499 was delayed in the network, or that either one of the data segments or the final ACK was duplicated in the network. Thus, the use of D-SACK allows the sender to more reliably infer that the first transmission of segment 1000-1499 was not dropped.

[AP99], [L99], and [LK00] note that the sender could unambiguously detect an unnecessary retransmit with the use of the timestamp option. [LK00] proposes a timestamp-based algorithm that minimizes the penalty for an unnecessary retransmit. [AP99] proposes a heuristic for detecting an unnecessary retransmit in an environment with neither timestamps nor SACK. [L99] also proposes a two-bit field as an alternate to the timestamp option for unambiguously marking the first three retransmissions of a packet. A similar idea was proposed in [ISO8073].

RESEARCH ISSUES:

The use of D-SACK allows the sender to detect some cases (e.g., when no ACK packets have been lost) when a a Fast Retransmit was due to packet reordering instead of packet loss. This allows the TCP sender to adjust the duplicate acknowledgment threshold, to prevent such unnecessary Fast Retransmits in the future. Coupled with this, when the sender determines, after the fact, that it has made an unnecessary window reduction, the sender has the option of "undoing" that reduction in the congestion window by resetting ssthresh to the value of the old congestion window, and slow-starting until the congestion window has reached that point.

Any proposal for "undoing" a reduction in the congestion window would have to address the possibility that the TCP receiver could be lying in its reports of received packets [SCWA99].

5.3. Retransmit Timeout Due to ACK Loss

If an entire window of ACKs is lost, a timeout will result. An example of this is given below:

```
Transmitted    Received    ACK Sent
Segment        Segment     (Including SACK Blocks)

500-999        500-999     1000 (ACK dropped)
1000-1499      1000-1499   1500 (ACK dropped)
1500-1999      1500-1999   2000 (ACK dropped)
2000-2499      2000-2499   2500 (ACK dropped)
(timeout)
500-999        500-999     2500, SACK=500-1000
                                 --------
```

In this case, all of the ACKs are dropped, resulting in a timeout. This condition can be identified because the first ACK received following the timeout carries a D-SACK block indicating duplicate data was received.

WITHOUT D-SACK:

Without the use of D-SACK, the sender in this case would be unable to decide that no data packets has been dropped.

RESEARCH ISSUES:

For a TCP that implements some form of ACK congestion control [BPK97], this ability to distinguish between dropped data packets and dropped ACK packets would be particularly useful. In this case, the connection could implement congestion control for the return (ACK) path independently from the congestion control on the forward (data) path.

5.4. Early Retransmit Timeout

If the sender's RTO is too short, an early retransmission timeout can occur when no packets have in fact been dropped in the network. An example of this is given below:

```
Transmitted    Received    ACK Sent
Segment        Segment     (Including SACK Blocks)

500-999        (delayed)
1000-1499      (delayed)
1500-1999      (delayed)
2000-2499      (delayed)
(timeout)
500-999        (delayed)
               500-999     1000
1000-1499      (delayed)
               1000-1499   1500
...
               1500-1999   2000
               2000-2499   2500
               500-999     2500, SACK=500-1000
                                 --------
               1000-1499   2500, SACK=1000-1500
                                 ---------
               ...
```

In this case, the first packet is retransmitted following the timeout. Subsequently, the original window of packets arrives at the receiver, resulting in ACKs for these segments. Following this, the retransmissions of these segments arrive, resulting in ACKs carrying SACK blocks which identify the duplicate segments.

This can be identified as an early retransmission timeout because the ACK for byte 1000 is received after the timeout with no SACK information, followed by an ACK which carries SACK information (500-999) indicating that the retransmitted segment had already been received.

WITHOUT D-SACK:

If D-SACK was not used and one of the duplicate ACKs was piggybacked on a data packet, the sender would not know how many duplicate packets had been received. If D-SACK was not used and none of the duplicate ACKs were piggybacked on a data packet, then the sender would have sent N duplicate packets, for some N, and received N duplicate ACKs. In this case, the sender could reasonably infer that some data or ACK packet had been replicated in the network, or that an early retransmission timeout had occurred (or that the receiver is lying).

RESEARCH ISSUES:

After the sender determines that an unnecessary (i.e., early) retransmit timeout has occurred, the sender could adjust parameters for setting the RTO, to prevent more unnecessary retransmit timeouts. Coupled with this, when the sender determines, after the fact, that it has made an unnecessary window reduction, the sender has the option of "undoing" that reduction in the congestion window.

6. Security Considerations

This document neither strengthens nor weakens TCP's current security properties.

7. Acknowledgements

We would like to thank Mark Handley, Reiner Ludwig, and Venkat Padmanabhan for conversations on these issues, and to thank Mark Allman for helpful feedback on this document.

8. References

[AP99] Mark Allman and Vern Paxson, On Estimating End-to-End Network Path Properties, SIGCOMM 99, August 1999. URL "http://www.acm.org/sigcomm/sigcomm99/papers/session7-3.html".

[BPS99] J.C.R. Bennett, C. Partridge, and N. Shectman, Packet Reordering is Not Pathological Network Behavior, IEEE/ACM Transactions on Networking, Vol. 7, No. 6, December 1999, pp. 789-798.

[BPK97] Hari Balakrishnan, Venkata Padmanabhan, and Randy H. Katz, The Effects of Asymmetry on TCP Performance, Third ACM/IEEE Mobicom Conference, Budapest, Hungary, Sep 1997. URL "http://www.cs.berkeley.edu/~padmanab/index.html#Publications".

[F99] Floyd, S., Re: TCP and out-of-order delivery, Message ID <199902030027.QAA06775@owl.ee.lbl.gov> to the end-to-end-interest mailing list, February 1999. URL "http://www.aciri.org/floyd/notes/TCP_Feb99.email".

[ISO8073] ISO/IEC, Information-processing systems - Open Systems Interconnection - Connection Oriented Transport Protocol Specification, Internation Standard ISO/IEC 8073, December 1988.

[L99] Reiner Ludwig, A Case for Flow Adaptive Wireless links, Technical Report UCB//CSD-99-1053, May 1999. URL "http://iceberg.cs.berkeley.edu/papers/Ludwig-FlowAdaptive/".

[LK00] Reiner Ludwig and Randy H. Katz, The Eifel Algorithm: Making TCP Robust Against Spurious Retransmissions, SIGCOMM Computer Communication Review, V. 30, N. 1, January 2000. URL "http://www.acm.org/sigcomm/ccr/archive/ccr-toc/ccr-toc-2000.html".

[RFC1323] Jacobson, V., Braden, R. and D. Borman, "TCP Extensions for High Performance", RFC 1323, May 1992.

[RFC2018] Mathis, M., Mahdavi, J., Floyd, S. and A. Romanow, "TCP Selective Acknowledgement Options", RFC 2018, April 1996.

[RFC2581] Allman, M., Paxson,V. and W. Stevens, "TCP Congestion Control", RFC 2581, April 1999.

[SCWA99] Stefan Savage, Neal Cardwell, David Wetherall, Tom Anderson, TCP Congestion Control with a Misbehaving Receiver, ACM Computer Communications Review, pp. 71-78, V. 29, N. 5, October, 1999. URL "http://www.acm.org/sigcomm/ccr/archive/ccr-toc/ccr-toc-99.html".

Authors' Addresses

Sally Floyd
AT&T Center for Internet Research at ICSI (ACIRI)

Phone: +1 510-666-6989
EMail: floyd@aciri.org
URL: http://www.aciri.org/floyd/

Jamshid Mahdavi
Novell

Phone: 1-408-967-3806
EMail: mahdavi@novell.com

Matt Mathis
Pittsburgh Supercomputing Center

Phone: 412 268-3319
EMail: mathis@psc.edu
URL: http://www.psc.edu/~mathis/

Matthew Podolsky
UC Berkeley Electrical Engineering & Computer Science Dept.

Phone: 510-649-8914
EMail: podolsky@eecs.berkeley.edu
URL: http://www.eecs.berkeley.edu/~podolsky

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.