Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 16, 2012

N. Ekiz
P. Amer
University of Delaware
P. Natarajan
Cisco Systems
R. Stewart
Researcher
J. Iyengar
Franklin & Marshall College

August 15, 2011

Non-Renegable Selective Acknowledgements (NR-SACKs) for SCTP
draft-natarajan-tsvwg-sctp-nrsack-08.txt

Abstract

Stream Control Transmission Protocol (SCTP) [RFC4960] specifies Selective Acknowledgements (SACKs) to allow an SCTP data receiver to acknowledge DATA chunks which arrive out-of-order. In SCTP, SACK information is advisory -- though SACKs notify a data sender about the reception of specific out-of-order data, the SCTP data receiver is permitted to later discard the data, a.k.a reneging. Since delivery of a SACKed out-of-order DATA chunk is not guaranteed, a copy of this DATA chunk MUST be kept in the data sender's retransmission queue until this DATA chunk is cumulatively acked.

By definition, data that has been delivered to the application is non-renegable by the SCTP data receiver. (Recall that, in SCTP, out-of-order data can sometimes be delivered.) Also, SCTP implementations can be configured such that the SCTP data receiver is not allowed to, and therefore, never reneges on out-of-order data. With SCTP's current SACK mechanism, non-renegable out-of-order data is selectively acked, and is (wrongly) deemed renegable by the SCTP data sender.

This document specifies an extension to SCTP's acknowledgment mechanism called Non-Renegable Selective Acknowledgements (NR-SACKs.) NR-SACKs enable a data receiver to explicitly inform the data sender of non-renegable out-of-order data. As opposed to renegable data, a data sender can consider non-renegable data as never requiring retransmission, and therefore can remove non-renegable data from the retransmission queue.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

In providing end-to-end reliable data transfer, SCTP specifies Cumulative Acknowledgements (ACKs), Selective ACKs (SACKs), and Duplicate Selective ACKs (D-SACKs). These three types of acks are carried in the following fields of the SACK chunk, respectively: Cumulative TSN Ack, Gap Ack Block, and Duplicate TSN. In this document, we refer to the Cumulative TSN Ack as the "cum-ack", the selective Gap Ack Blocks as "gap-acks", and the Duplicate TSN selective acks as "dup-TSN reports".

Gap-acks acknowledge DATA chunks that arrive out-of-order to a transport layer data receiver. A gap-ack in SCTP is advisory, in that, while it notifies a data sender about the reception of indicated DATA chunks, the data receiver is permitted to later discard DATA chunks that it previously had gap-acked. Discarding a previously gap-acked DATA chunk is known as "reneging." Because of the possibility of reneging in SCTP, any gap-acked DATA chunk MUST NOT be removed from the data sender’s retransmission queue until the DATA chunk is later cum-acked.

Situations exist when a data receiver knows that reneging on a particular out-of-order DATA chunk will never take place, such as (but not limited to) after an out-of-order DATA chunk is delivered to the receiving application. This document describes an extension to SCTP to allow for Non-Renegable Selective Acknowledgments (NR-SACKs). A new NR-SACK chunk type is described that allows this extension to be implemented.

The NR-SACK chunk is an extension of the existing SACK chunk. Several fields are identical, including the Cumulative TSN Ack, the Advertised Receiver Window Credit (a_rwnd), and Duplicate TSNs. These fields have the same semantics as described in [RFC4960].

NR-SACKs extend SACKs by also identifying out-of-order DATA chunks that a receiver either: (1) has delivered to its receiving application, or (2) takes full responsibility to eventually deliver to its receiving application. These out-of-order DATA chunks are "non-renegable." Non-Renegable data are reported in the NR Gap Ack Block field of the NR-SACK chunk as described in Section 4.1. We refer to non-renegable selective acknowledgements as "nr-gap-acks."

When an out-of-order DATA chunk is nr-gap-acked, the data sender no longer needs to keep that particular DATA chunk in its retransmission queue, thus allowing the data sender to free up its buffer space sooner than if the DATA chunk were only gap-acked. NR-SACKs have been shown to better utilize the data sender’s memory and improve throughput, at the trade-off of generating and processing additional acknowledgement information [Natarajan], [Yilmaz].

An SCTP message is encapsulated within a single DATA chunk or within multiple DATA chunks in case of fragmentation. In this document without loss of generality, each application message maps to a single transport layer DATA chunk, and delivering a DATA chunk to a receiving application means delivering the message carried within the DATA chunk to a receiving application.

SCTP divides an end-to-end association into independent logical data streams (a.k.a. multistreaming.) A DATA chunk that arrives in-sequence within a stream can be delivered to the receiving application even if the DATA chunk is out-of-order relative to the association’s overall flow of data. These out-of-order DATA chunks are "deliverable." By definition, a DATA chunk marked for unordered delivery also is "deliverable" to the receiving application immediately upon reception, regardless of its position within the overall flow of data.

With current SACKs in SCTP, it is not possible for a data receiver to inform a data sender if or when a particular out-of-order "deliverable" DATA chunk has been "delivered" to the receiving application. Thus the data sender MUST keep a copy of every gap-acked out-of-order DATA chunk(s) in the data sender’s retransmission queue until the DATA chunk is cum-acked. This use of the data sender’s retransmission queue is wasteful. Given the receiving application has received the data, the data sender has no reason to keep this data in its retransmission queue. Yet, the sending transport layer keeps the data because no mechanism currently exists to indicate which out-of-order DATA chunks have been delivered. (Note: once a DATA chunk is delivered to the receiving application, it is impossible for the data receiver to renege on that DATA chunk.)

If NR-SACKs are used, the data receiver MAY include the TSN of a delivered out-of-order DATA chunk in an NR-SACK to inform the data sender that the delivery has occurred, allowing the data sender to remove the copy of the delivered DATA chunk from the data sender’s retransmission queue even before the DATA chunk is cum-acked.

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Negotiation

Before sending/receiving NR-SACKs, both peer endpoints MUST agree on using NR-SACKs. This agreement MUST be negotiated during association establishment. NR-SACK is an extension to the core SCTP, and SCTP extensions that an endpoint supports are reported to the peer endpoint in Supported Extensions Parameter during association establishment (see Section 4.2.7 of [RFC5061].) The Supported Extensions Parameter consists of a list of non-standard Chunk Types that are supported by the sender.

An endpoint supporting the NR-SACK extension MUST list the NR-SACK chunk in the Supported Extensions Parameter carried in the INIT or INIT-ACK chunk, depending on whether the endpoint initiates or responds to the initiation of the association. If the NR-SACK chunk type ID is listed in the Chunk Types List of the Supported Extensions Parameter, then the receiving endpoint MUST assume that the NR-SACK chunk is supported by the sending endpoint.

Both endpoints MUST support NR-SACKs for either endpoint to send an NR-SACK. If an endpoint establishes an association with a remote endpoint that does not list NR-SACK in the Supported Extensions Parameter carried in INIT chunk, then both endpoints of the association MUST NOT use NR-SACKs. After association establishment, an endpoint MUST NOT renegotiate the use of NR-SACKs.

Once both endpoints indicate during association establishment that they support the NR-SACK extension, each endpoint SHOULD acknowledge received DATA chunks with NR-SACK chunks, and not SACK chunks. That is, throughout an SCTP association, both endpoints SHOULD send either SACK chunks or NR-SACK chunks, never a mixture of the two.

4. The New Chunk Type: Non-Renegable SACK (NR-SACK)

Table 1 illustrates a new chunk type that will be used to transfer NR-SACK information.

Chunk Type	Chunk Name	
0x10	Non-Renegable Selective Acknowledgment	(NR-SACK)

Table 1: NR-SACK Chunk

As the NR-SACK chunk replaces the SACK chunk, many SACK chunk fields are preserved in the NR-SACK chunk. These preserved fields have the same semantics with the corresponding SACK chunk fields, as defined in [RFC4960], Section 3.3.4. The Gap Ack fields from RFC4960 have been renamed as R Gap Ack to emphasize their renegable nature. Their semantics are unchanged. For completeness, we describe all fields of the NR-SACK chunk, including those that are identical in the SACK chunk.

Similar to the SACK chunk, the NR-SACK chunk is sent to a peer endpoint to (1) acknowledge DATA chunks received in-order, (2) acknowledge DATA chunks received out-of-order, and (3) identify DATA chunks received more than once (i.e., duplicate.) In addition, the NR-SACK chunk (4) informs the peer endpoint of non-renegable out-of-order DATA chunks.

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |   Type = 0x10 |  Chunk Flags  |         Chunk Length          |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                      Cumulative TSN Ack                       |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |          Advertised Receiver Window Credit (a_rwnd)           |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |Number of R Gap Ack Blocks = N |Number of NR Gap Ack Blocks = M|
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     | Number of Duplicate TSNs = X  |           Reserved            |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     | R Gap Ack Block #1 Start      |   R Gap Ack Block #1 End      |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     /                                                               /
     \                              ...                              \
     /                                                               /
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |  R Gap Ack Block #N Start     |  R Gap Ack Block #N End       |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |  NR Gap Ack Block #1 Start    |   NR Gap Ack Block #1 End     |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     /                                                               /
     \                              ...                              \
     /                                                               /
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |   NR Gap Ack Block #M Start   |  NR Gap Ack Block #M End      |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                       Duplicate TSN 1                         |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     /                                                               /
     \                              ...                              \
     /                                                               /
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                       Duplicate TSN X                         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type: 8 bits

This field holds the IANA defined chunk type for NR-SACK chunk. The suggested value of this field for IANA is 0x10.

Chunk Flags: 8 bits

Currently not used. It is recommended a sender set all bits to zero on transmit, and a receiver ignore this field.

Chunk Length: 16 bits (unsigned integer) [Same as SACK chunk]

This value represents the size of the chunk in bytes including the Chunk Type, Chunk Flags, Chunk Length, and Chunk Value fields.

Cumulative TSN Ack: 32 bits (unsigned integer) [Same as SACK chunk]

The value of the Cumulative TSN Ack is the last TSN received before a break in the sequence of received TSNs occurs. The next TSN value following the Cumulative TSN Ack has not yet been received at the endpoint sending the NR-SACK.

Advertised Receiver Window Credit (a_rwnd): 32 bits (unsigned integer) [Same as SACK chunk]

Indicates the updated receive buffer space in bytes of the sender of this NR-SACK, see Section 6.2.1 of [RFC4960] for details.

Number of (R)enegable Gap Ack Blocks (N): 16 bits (unsigned integer)

Indicates the number of Renegable Gap Ack Blocks included in this NR-SACK.

Number of (N)on(R)enegable Gap Ack Blocks (M): 16 bits (unsigned integer)

Indicates the number of Non-Renegable Gap Ack Blocks included in this NR-SACK.

Number of Duplicate TSNs (X): 16 bits [Same as SACK chunk]

Contains the number of duplicate TSNs the endpoint has received. Each duplicate TSN is listed following the NR Gap Ack Block list.

Reserved : 16 bits

Currently not used. It is recommended a sender set all bits to zero on transmit, and a receiver ignore this field.

(R)enegable Gap Ack Blocks:

The NR-SACK contains zero or more R Gap Ack Blocks. Each R Gap Ack Block acknowledges a subsequence of renegable out-of-order TSNs. By definition, all TSNs acknowledged by R Gap Ack Blocks are "greater than" the value of the Cumulative TSN Ack.

Because of TSN numbering wraparound, comparisons and all arithmetic operations discussed in this document are based on "Serial Number Arithmetic" as described in Section 1.6 of [RFC4960].

R Gap Ack Blocks are repeated for each R Gap Ack Block up to 'N' defined in the Number of R Gap Ack Blocks field. All DATA chunks with TSNs >= (Cumulative TSN Ack + R Gap Ack Block Start) and <= (Cumulative TSN Ack + R Gap Ack Block End) of each R Gap Ack Block are assumed to have been received correctly, and are renegable.

R Gap Ack Block Start: 16 bits (unsigned integer)

Indicates the Start offset TSN for this R Gap Ack Block. This number is set relative to the cumulative TSN number defined in Cumulative TSN Ack field. To calculate the actual start TSN number, the Cumulative TSN Ack is added to this offset number. The calculated TSN identifies the first TSN in this R Gap Ack Block that has been received.

R Gap Ack Block End: 16 bits (unsigned integer)

Indicates the End offset TSN for this R Gap Ack Block. This number is set relative to the cumulative TSN number defined in the Cumulative TSN Ack field. To calculate the actual TSN number, the Cumulative TSN Ack is added to this offset number. The calculated TSN identifies the TSN of the last DATA chunk received in this R Gap Ack Block.

N(on)R(enegable) Gap Ack Blocks:

The NR-SACK contains zero or more NR Gap Ack Blocks. Each NR Gap Ack Block acknowledges a continuous subsequence of non-renegable out-of-order DATA chunks. If a TSN is nr-gap-acked in any NR-SACK chunk, then all subsequently transmitted NR-SACKs with a smaller cum-ack value than that TSN SHOULD also nr-gap-ack that TSN.

NR Gap Ack Blocks are repeated for each NR Gap Ack Block up to 'M' defined in the Number of NR Gap Ack Blocks field. All DATA chunks with TSNs >= (Cumulative TSN Ack + NR Gap Ack Block Start) and <= (Cumulative TSN Ack + NR Gap Ack Block End) of each NR Gap Ack Block are assumed to be received correctly, and are Non-Renegable.

NR Gap Ack Block Start: 16 bits (unsigned integer)

Indicates the Start offset TSN for this NR Gap Ack Block. This number is set relative to the cumulative TSN number defined in Cumulative TSN Ack field. To calculate the actual TSN number, the Cumulative TSN Ack is added to this offset number. The calculated TSN identifies the first TSN in this NR Gap Ack Block that has been received.

NR Gap Ack Block End: 16 bits (unsigned integer)

Indicates the End offset TSN for this NR Gap Ack Block. This number is set relative to the cumulative TSN number defined in Cumulative TSN Ack field. To calculate the actual TSN number, the Cumulative TSN Ack is added to this offset number. The calculated TSN identifies the TSN of the last DATA chunk received in this NR Gap Ack Block.

Note:

NR Gap Ack Blocks and R Gap Ack Blocks in an NR-SACK chunk SHOULD acknowledge disjoint sets of TSNs. That is, an out-of-order TSN SHOULD be listed in either an R Gap Ack Block or an NR Gap Ack Block, but not the both. R Gap Ack Blocks and NR Gap Ack Blocks together provide the information as do the Gap Ack Block of a SACK chunk, plus additional information about non-renegability.

If all out-of-order data acked by an NR-SACK are renegable, then the Number of NR Gap Ack Blocks MUST be set to 0. If all out-of-order data acked by an NR-SACK are non-renegable, then the Number of R Gap Ack Blocks SHOULD be set to 0. TSNs listed in R Gap Ack Block will be referred as r-gap-acked.

Duplicate TSN: 32 bits (unsigned integer) [Same as SACK chunk]

Indicates a duplicate TSN received since the last NR-SACK was sent. Exactly ’X’ duplicate TSNs SHOULD be reported where ’X’ was defined in Number of Duplicate TSNs field.

Each duplicate TSN is listed in this field as many times as the TSN was received since the previous NR-SACK was sent. For example, if a data receiver were to get the TSN 19 three times, the data receiver would list 19 twice in the outbound NR-SACK. After sending the NR-SACK if the receiver received one more TSN 19, the receiver would list 19 as a duplicate once in the next outgoing NR-SACK.

5. An Illustrative Example

Assume the following DATA chunks have arrived at the receiver.

```
--------------------------------
| TSN=16| SID=2 | SSN=N/A| U=1 |
--------------------------------
| TSN=15| SID=1 | SSN= 4 | U=0 |
--------------------------------
| TSN=14| SID=0 | SSN= 4 | U=0 |
--------------------------------
| TSN=13| SID=2 | SSN=N/A| U=1 |
--------------------------------
|                              |
--------------------------------
| TSN=11| SID=0 | SSN= 3 | U=0 |
-------------------------------
|                              |
--------------------------------
|                              |
--------------------------------
| TSN=8 | SID=2 | SSN=N/A| U=1 |
--------------------------------
| TSN=7 | SID=1 | SSN= 2 | U=0 |
--------------------------------
| TSN=6 | SID=1 | SSN= 1 | U=0 |
--------------------------------
| TSN=5 | SID=0 | SSN= 1 | U=0 |
--------------------------------
|                              |
--------------------------------
| TSN=3 | SID=1 | SSN= 0 | U=0 |
--------------------------------
| TSN=2 | SID=0 | SSN= 0 | U=0 |
--------------------------------
```

The above figure shows the list of DATA chunks at the receiver. TSN denotes the transmission sequence number of the DATA chunk, SID denotes the stream id to which the DATA chunk belongs, SSN denotes the sequence number of the DATA chunk within its stream, and the U bit denotes whether the DATA chunk requires ordered(=0) or unordered(=1) delivery [RFC4960]. Note that TSNs 4,9,10, and 12 have not arrived.

This data can be viewed as three separate streams as follows (assume each stream begins with SSN=0.) Note that in this example, the application uses stream 2 for unordered data transfer. By definition, SSN fields of unordered DATA chunks are ignored.

Stream-0:

SSN:	0	1	2	3	4
TSN:	2	5		11	14
U-Bit:	0	0		0	0

Stream-1:

SSN:	0	1	2	3	4
TSN:	3	6	7		15
U-Bit:	0	0	0		0

Stream-2:

SSN:	N/A	N/A	N/A
TSN:	8	13	16
U-Bit:	1	1	1

The NR-SACK to acknowledge the above data SHOULD be constructed as follows for each of the three cases described below (the a_rwnd is arbitrarily set to 4000):

CASE-1: Minimal Data Receiver Responsibility - no out-of-order deliverable data yet delivered

None of the deliverable out-of-order DATA chunks have been delivered, and the receiver of the above data does not take responsibility for any of the received out-of-order DATA chunks. The receiver reserves the right to renege any or all of the out-of-order DATA chunks.

```
       +-----------------------------+-----------------------------+
       | Type = 0x10  |   00000000   |      Chunk Length = 32      |
       +-----------------------------+-----------------------------+
       |                    Cumulative TSN Ack = 3                 |
       +-----------------------------+-----------------------------+
       |                        a_rwnd = 4000                      |
       +-----------------------------+-----------------------------+
       | Num of R Gap Ack Blocks = 3 |Num of NR Gap Ack Blocks = 0 |
       +-----------------------------+-----------------------------+
       |    Num of Duplicates = 0    |            0x00             |
       +-----------------------------+-----------------------------+
       |R Gap Ack Block #1 Start = 2 | R Gap Ack Block #1 End = 5  |
       +-----------------------------+-----------------------------+
       |R Gap Ack Block #2 Start = 8 | R Gap Ack Block #2 End = 8  |
       +-----------------------------+-----------------------------+
       |R Gap Ack Block #3 Start = 10| R Gap Ack Block #3 End = 13 |
       +-----------------------------+-----------------------------+
```

CASE-2: Minimal Data Receiver Responsibility - all out-of-order deliverable data delivered

In this case, the NR-SACK chunk is being sent after the data receiver has delivered all deliverable out-of-order DATA chunks to its receiving application(i.e., TSNs 5,6,7,8,13, and 16.) The receiver reserves the right to renege on all undelivered out-of-order DATA chunks(i.e., TSNs 11,14, and 15.)

```
      +------------------------------+------------------------------+
      | Type = 0x10  |     0x00      |       Chunk Length = 40      |
      +------------------------------+------------------------------+
      |                     Cumulative TSN Ack = 3                   |
      +------------------------------+------------------------------+
      |                        a_rwnd = 4000                         |
      +------------------------------+------------------------------+
      | Num of R Gap Ack Blocks = 2  | Num of NR Gap Ack Blocks = 3 |
      +------------------------------+------------------------------+
      |    Num of Duplicates = 0     |              0x00            |
      +------------------------------+------------------------------+
      | R Gap Ack Block #1 Start = 8 | R Gap Ack Block #1 End = 8   |
      +------------------------------+------------------------------+
      | R Gap Ack Block #2 Start = 11| R Gap Ack Block #2 End = 12  |
      +------------------------------+------------------------------+
      |NR Gap Ack Block #1 Start = 2 | NR Gap Ack Block #1 End = 5  |
      +------------------------------+------------------------------+
      |NR Gap Ack Block #2 Start = 10| NR Gap Ack Block #2 End = 10 |
      +------------------------------+------------------------------+
      |NR Gap Ack Block #3 Start = 13| NR Gap Ack Block #3 End = 13 |
      +------------------------------+------------------------------+
```

CASE-3: Maximal Data Receiver Responsibility

In this special case, all out-of-order data blocks acknowledged are non-renegable. This case would occur when the data receiver is programmed never to renege, and takes responsibility to deliver all DATA chunks that arrive out-of-order. In this case Num of R Gap Ack Blocks is zero indicating all reported out-of-order TSNs are nr-gap-acked.

```
     +--------------------------------+-------------------------------+
     |  Type = 0x10   |     0x00      |     Chunk Length = 32         |
     +--------------------------------+-------------------------------+
     |                      Cumulative TSN Ack = 3                    |
     +--------------------------------+-------------------------------+
     |                          a_rwnd = 4000                         |
     +--------------------------------+-------------------------------+
     |  Num of R Gap Ack Blocks = 0   |  Num of NR Gap Ack Blocks = 3 |
     +--------------------------------+-------------------------------+
     |     Num of Duplicates = 0      |             0x00              |
     +--------------------------------+-------------------------------+
     | NR Gap Ack Block #1 Start = 2  | NR Gap Ack Block #1 End = 5   |
     +--------------------------------+-------------------------------+
     | NR Gap Ack Block #2 Start = 8  | NR Gap Ack Block #2 End = 8   |
     +--------------------------------+-------------------------------+
     | NR Gap Ack Block #3 Start = 10 | NR Gap Ack Block #3 End = 13  |
     +--------------------------------+-------------------------------+
```

6. Procedures

The procedures regarding "when" to send an NR-SACK chunk are identical to the procedures regarding when to send a SACK chunk, as outlined in Section 6.2 of [RFC4960].

6.1. Sending an NR-SACK chunk

All of the NR-SACK chunk fields identical to the SACK chunk MUST be formed as described in Section 6.2 of [RFC4960].

It is up to the data receiver whether or not to take responsibility for delivery of each out-of-order DATA chunk. An out-of-order DATA chunk that has already been delivered, or that the receiver takes responsibility to deliver (i.e., guarantees not to renege) is Non Renegable(NR), and SHOULD be included in an NR Gap Ack Block field of the outgoing NR-SACK. All other out-of-order data is (R)enegable, and SHOULD be included in R Gap Ack Block field of the outgoing NR-SACK.

Consider three types of data receiver:

CASE-1: Data receiver takes no responsibility for delivery of any out-of-order DATA chunks

CASE-2: Data receiver takes responsibility for all out-of-order DATA chunks that are "deliverable" (i.e., DATA chunks in-sequence within the stream they belong to, or DATA chunks whose (U)nordered bit is 1)

CASE-3: Data receiver takes responsibility for delivery of all out-of-order DATA chunks, whether deliverable or not deliverable

The data receiver SHOULD follow the procedures outlined below for building the NR-SACK.

CASE-1:

1A) Identify the TSNs received out-of-order.

1B) For these out-of-order TSNs, identify the R Gap Ack Blocks. Fill the Number of R Gap Ack Blocks (N) field, R Gap Ack Block #i Start, and R Gap Ack Block #i End where i goes from 1 to N.

1C) Set the Number of NR Gap Ack Blocks (M) field to 0.

CASE-2:

2A) Identify the TSNs received out-of-order.

2B) For the received out-of-order TSNs, check the (U)nordered bit of each TSN. Tag unordered TSNs as NR.

2C) For each stream, also identify the TSNs received out-of-order but are in-sequence within that stream. Tag those in-sequence TSNs as NR.

2D) Tag all out-of-order data that is not NR as (R)enegable.

2E) For those TSNs tagged as (R)enegable, identify the (R)enegable Blocks. Fill the Number of R Gap Ack Blocks(N) field, R Gap Ack Block #i Start, and R Gap Ack Block #i End where i goes from 1 to N.

2F) For those TSNs tagged as NR, identify the NR Blocks. Fill the Number of NR Gap Ack Blocks(M) field, NR Gap Ack Block #i Start, and NR Gap Ack Block #i End where i goes from 1 to M.

CASE-3:

3A) Identify the TSNs received out-of-order. All of these TSNs SHOULD be nr-gap-acked.

3B) Set the Number of R Gap Ack Blocks (N) field to 0.

3C) For these out-of-order TSNs, identify the NR Gap Ack Blocks. Fill the Number of NR Gap Ack Blocks (M) field, NR Gap Ack Block #i Start, and NR Gap Ack Block #i End where i goes from 1 to M.

RFC4960 states that the SCTP endpoint MUST report as many Gap Ack Blocks as can fit in a single SACK chunk limited by the current path MTU. When using NR-SACKs, the SCTP endpoint SHOULD fill as many R Gap Ack Blocks and NR Gap Ack Blocks starting from the Cumulative TSN Ack value as can fit in a single NR-SACK chunk limited by the current path MTU. If space remains, the SCTP endpoint SHOULD fill as many Duplicate TSNs as possible starting from Cumulative TSN Ack value.

6.2. Receiving an NR-SACK Chunk

When an NR-SACK chunk is received, all of the NR-SACK fields identical to a SACK chunk SHOULD be processed and handled as in SACK chunk handling outlined in Section 6.2.1 of [RFC4960].

The NR Gap Ack Block Start(s) and NR Gap Ack Block End(s) are offsets relative to the cum-ack. To calculate the actual range of nr-gap-acked TSNs, the cum-ack MUST be added to the Start and End.

For example, assume an incoming NR-SACK chunk’s cum-ack is 12 and an NR Gap Ack Block defines the NR Gap Ack Block Start=5, and the NR Gap Ack Block End=7. This NR Gap Ack block nr-gap-acks TSNs 17 through 19 inclusive.

Upon reception of an NR-SACK chunk, all TSNs listed in either R Gap Ack Block(s) or NR Gap Ack Block(s) SHOULD be processed as would be TSNs included in Gap Ack Block(s) of a SACK chunk. All TSNs in all

NR Gap Ack Blocks SHOULD be removed from the data sender’s retransmission queue as their delivery to the receiving application has either already occurred, or is guaranteed by the data receiver.

Although R Gap Ack Blocks and NR Gap Ack Blocks SHOULD be disjoint sets, NR-SACK processing SHOULD work if an NR-SACK chunk has a TSN listed in both an R Gap Ack Block and an NR Gap Ack Block. In this case, the TSN SHOULD be treated as Non-Renegable.

Implementation Note:

Most of NR-SACK processing at the data sender can be implemented by using the same routines as in SACK that process the cum ack and the gap ack(s), followed by removal of nr-gap-acked DATA chunks from the retransmission queue. However, with NR-SACKs, as out-of-order DATA is sometimes removed from the retransmission queue, the gap ack processing routine should recognize that the data sender’s retransmission queue has some transmitted data removed. For example, while calculating missing reports, the gap ack processing routine cannot assume that the highest TSN transmitted is always at the tail (right edge) of the retransmission queue.

7. Security Considerations

This document does not add any security considerations to those specified in [RFC4960].

8. IANA considerations

This document defines a new chunk type to transfer the NR-SACK information. Table 2 illustrates the new chunk type.

The new chunk type must come from the range of chunk types where the upper two bits are zero. We recommend 0x10 but any other available code point with the upper two bits set to zero is acceptable.

Chunk Type	Chunk Name	
0x10	Non-Renegable Selective Acknowledgment	(NR-SACK)

Table 2: NR-SACK Chunk

9. Acknowledgments

This Internet Draft is the result of a great deal of constructive discussion with several people, notably Phillip Conrad, Ertugrul Yilmaz, and Jonathan Leighton.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, September 2007.

[RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

10.2. Informative References

[Natarajan] Natarajan, P., Ekiz, N., Yilmaz, E., Amer, P., Iyengar, J., and R. Stewart, "Non-Renegable Selective Acknowledgments (NR-SACKs) for SCTP", International Conference on Network Protocols (ICNP) 2008, Orlando, October 2008 .

[Yilmaz] Yilmaz, E., Ekiz, N., Natarajan, P., Amer, P., Leighton, J., Baker, F., and R. Stewart, "Throughput analysis of Non-Renegable Selective Acknowledgments (NR-SACKs) for SCTP", Comput. Commun. (2010), doi:10.1016/j.comcom.2010.06.028 .

Authors' Addresses

Nasif Ekiz
University of Delaware
Computer and Information Sciences Department
Newark, DE 19716
USA

Phone:
Email: nekiz@udel.edu

Paul D. Amer
University of Delaware
Computer and Information Sciences Department
Newark, DE 19716
USA

Phone: 302-831-1944
Email: amer@cis.udel.edu

Preethi Natarajan
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
USA

Phone:
Email: prenatar@cisco.com

Randall R. Stewart
Researcher
Chapi, SC 29036
USA

Phone: 202-595-8626
Email: randall@lakerest.net

Janardhan Iyengar
Franklin & Marshall College
Math and Computer Science
PO Box 3003
Lancaster, PA 17604-3003
USA

Phone: 717-358-4774
Email: jiyengar@fandm.edu