Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 19, 2016

R. Stewart
Netflix, Inc.
M. Tuexen
Muenster Univ. of Appl. Sciences
K. Nielsen
M. Proshin
Ericsson

March 18, 2016

RFC 4960 Errata and Issues
draft-tuexen-tsvwg-rfc4960-errata-02.txt

Abstract

This document is a compilation of issues found since the publication of RFC4960 in September 2007 based on experience with implementing, testing, and using SCTP along with the suggested fixes. This document provides deltas to RFC4960 and is organized in a time based way. The issues are listed in the order they were brought up. Because some text is changed several times the last delta in the text is the one which should be applied. In addition to the delta a description of the problem and the details of the solution are also provided.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 19, 2016.

Table of Contents

1. Introduction

This document contains a compilation of all defects found up until the publishing of this document for [RFC4960] specifying the Stream Control Transmission Protocol (SCTP). These defects may be of an editorial or technical nature. This document may be thought of as a companion document to be used in the implementation of SCTP to clarify errors in the original SCTP document.

This document provides a history of the changes that will be compiled into a BIS document for [RFC4960]. It is structured similar to [RFC4460].

Each error will be detailed within this document in the form of:

- The problem description,
- The text quoted from [RFC4960],
- The replacement text that should be placed into an upcoming BIS document,
- A description of the solution.

Note that when reading this document one must use care to assure that a field or item is not updated further on within the document. Each section should be applied in sequence to the original [RFC4960] since this document is a historical record of the sequential changes that have been found necessary at various inter-op events and through discussion on the list.

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. Corrections to RFC 4960

3.1. Path Error Counter Threshold Handling

3.1.1. Description of the Problem

The handling of the 'Path.Max.Retrans' parameter is described in Section 8.2 and Section 8.3 of [RFC4960] in an Inconsistent way. Whereas Section 8.2 describes that a path is marked inactive when the path error counter exceeds the threshold, Section 8.3 says the path is marked inactive when the path error counter reaches the threshold.

This issue was reported as an Errata for [RFC4960] with Errata ID 1440.

3.1.2. Text Changes to the Document

Old text: (Section 8.3)

When the value of this counter reaches the protocol parameter 'Path.Max.Retrans', the endpoint should mark the corresponding destination address as inactive if it is not so marked, and may also optionally report to the upper layer the change of reachability of this destination address. After this, the endpoint should continue HEARTBEAT on this destination address but should stop increasing the counter.

New text: (Section 8.3)

When the value of this counter exceeds the protocol parameter 'Path.Max.Retrans', the endpoint should mark the corresponding destination address as inactive if it is not so marked, and may also optionally report to the upper layer the change of reachability of this destination address. After this, the endpoint should continue HEARTBEAT on this destination address but should stop increasing the counter.

3.1.3. Solution Description

The intended state change should happen when the threshold is exceeded.

3.2. Upper Layer Protocol Shutdown Request Handling

3.2.1. Description of the Problem

Section 9.2 of [RFC4960] describes the handling of received SHUTDOWN chunks in the SHUTDOWN-RECEIVED state instead of the handling of shutdown requests from its upper layer in this state.

This issue was reported as an Errata for [RFC4960] with Errata ID 1574.

3.2.2. Text Changes to the Document

Old text: (Section 9.2)

Once an endpoint has reached the SHUTDOWN-RECEIVED state, it MUST NOT send a SHUTDOWN in response to a ULP request, and should discard subsequent SHUTDOWN chunks.

New text: (Section 9.2)

Once an endpoint has reached the SHUTDOWN-RECEIVED state, it MUST NOT send a SHUTDOWN in response to a ULP request, and should discard subsequent ULP shutdown requests.

3.2.3. Solution Description

The text never intended the SCTP endpoint to ignore SHUTDOWN chunks from its peer. If it did the endpoints could never gracefully terminate associations in some cases.

3.3. Registration of New Chunk Types

3.3.1. Description of the Problem

Section 14.1 of [RFC4960] should deal with new chunk types, however, the text refers to parameter types.

This issue was reported as an Errata for [RFC4960] with Errata ID 2592.

3.3.2. Text Changes to the Document

Old text: (Section 14.1)

The assignment of new chunk parameter type codes is done through an IETF Consensus action, as defined in [RFC2434]. Documentation of the chunk parameter MUST contain the following information:

New text: (Section 14.1)

The assignment of new chunk type codes is done through an IETF Consensus action, as defined in [RFC2434]. Documentation of the chunk type MUST contain the following information:

3.3.3. Solution Description

Refer to chunk types as intended.

3.4. Variable Parameters for INIT Chunks

3.4.1. Description of the Problem

Newlines in wrong places break the layout of the table of variable parameters for the INIT chunk in Section 3.3.2 of [RFC4960].

This issue was reported as an Errata for [RFC4960] with Errata ID 3291 and Errata ID 3804.

3.4.2. Text Changes to the Document

Old text: (Section 3.3.2)

Variable Parameters Status Type Value

IPv4 Address (Note 1) Optional 5 IPv6 Address
(Note 1) Optional 6 Cookie Preservative
Optional 9 Reserved for ECN Capable (Note 2) Optional
32768 (0x8000) Host Name Address (Note 3) Optional
11 Supported Address Types (Note 4) Optional 12

New text: (Section 3.3.2)

Variable Parameters Status Type Value

IPv4 Address (Note 1) Optional 5
IPv6 Address (Note 1) Optional 6
Cookie Preservative Optional 9
Reserved for ECN Capable (Note 2) Optional 32768 (0x8000)
Host Name Address (Note 3) Optional 11
Supported Address Types (Note 4) Optional 12

3.4.3. Solution Description

Fix the formatting of the table.

3.5. CRC32c Sample Code on 64-bit Platforms

3.5.1. Description of the Problem

The sample code for computing the CRC32c provided in [RFC4960] assumes that a variable of type unsigned long uses 32 bits. This is not true on some 64-bit platforms (for example the ones using LP64).

This issue was reported as an Errata for [RFC4960] with Errata ID 3423.

3.5.2. Text Changes to the Document

```
---------
Old text: (Appendix C)
---------

unsigned long
generate_crc32c(unsigned char *buffer, unsigned int length)
{
  unsigned int i;
  unsigned long crc32 = ~0L;

---------
New text: (Appendix C)
---------

unsigned long
generate_crc32c(unsigned char *buffer, unsigned int length)
{
  unsigned int i;
  unsigned long crc32 = 0xffffffffL;
```

3.5.3. Solution Description

Use 0xffffffffL instead of ~0L which gives the same value on platforms using 32 bits or 64 bits for variables of type unsigned long.

3.6. Endpoint Failure Detection

3.6.1. Description of the Problem

The handling of the association error counter defined in Section 8.1 of [RFC4960] can result in an association failure even if the path used for data transmission is available, but idle.

This issue was reported as an Errata for [RFC4960] with Errata ID 3788.

3.6.2. Text Changes to the Document

Old text: (Section 8.1)

An endpoint shall keep a counter on the total number of consecutive retransmissions to its peer (this includes retransmissions to all the destination transport addresses of the peer if it is multi-homed), including unacknowledged HEARTBEAT chunks.

New text: (Section 8.1)

An endpoint shall keep a counter on the total number of consecutive retransmissions to its peer (this includes data retransmissions to all the destination transport addresses of the peer if it is multi-homed), including the number of unacknowledged HEARTBEAT chunks observed on the path which currently is used for data transfer. Unacknowledged HEARTBEAT chunks observed on paths different from the path currently used for data transfer shall not increment the association error counter, as this could lead to association closure even if the path which currently is used for data transfer is available (but idle).

3.6.3. Solution Description

A more refined handling for the association error counter is defined.

3.7. Data Transmission Rules

3.7.1. Description of the Problem

When integrating the changes to Section 6.1 A) of [RFC2960] as described in Section 2.15.2 of [RFC4460] some text was duplicated and became the final paragraph of Section 6.1 A) of [RFC4960].

This issue was reported as an Errata for [RFC4960] with Errata ID 4071.

3.7.2. Text Changes to the Document

Old text: (Section 6.1 A))

The sender MUST also have an algorithm for sending new DATA chunks to avoid silly window syndrome (SWS) as described in [RFC0813]. The algorithm can be similar to the one described in Section 4.2.3.4 of [RFC1122].

However, regardless of the value of rwnd (including if it is 0), the data sender can always have one DATA chunk in flight to the receiver if allowed by cwnd (see rule B below). This rule allows the sender to probe for a change in rwnd that the sender missed due to the SACK having been lost in transit from the data receiver to the data sender.

New text: (Section 6.1 A))

The sender MUST also have an algorithm for sending new DATA chunks to avoid silly window syndrome (SWS) as described in [RFC0813]. The algorithm can be similar to the one described in Section 4.2.3.4 of [RFC1122].

3.7.3. Solution Description

Last paragraph of Section 6.1 A) removed as intended in Section 2.15.2 of [RFC4460].

3.8. T1-Cookie Timer

3.8.1. Description of the Problem

Figure 4 of [RFC4960] illustrates the SCTP association setup. However, it incorrectly shows that the T1-init timer is used in the COOKIE-ECHOED state whereas the T1-cookie timer should have been used instead.

This issue was reported as an Errata for [RFC4960] with Errata ID 4400.

3.8.2. Text Changes to the Document

```
---------
Old text: (Section 5.1.6, Figure 4)
---------

COOKIE ECHO [Cookie_Z] ------\
(Start T1-init timer)         \
(Enter COOKIE-ECHOED state)    \---> (build TCB enter ESTABLISHED
                                      state)
                               /---- COOKIE-ACK
                              /
(Cancel T1-init timer, <-----/
 Enter ESTABLISHED state)

---------
New text: (Section 5.1.6, Figure 4)
---------

COOKIE ECHO [Cookie_Z] ------\
(Start T1-cookie timer)       \
(Enter COOKIE-ECHOED state)    \---> (build TCB enter ESTABLISHED
                                      state)
                               /---- COOKIE-ACK
                              /
(Cancel T1-cookie timer, <---/
 Enter ESTABLISHED state)
```

3.8.3. Solution Description

Change the figure such that the T1-cookie timer is used instead of the T1-init timer.

3.9. Miscellaneous Typos

3.9.1. Description of the Problem

While processing [RFC4960] some typos were not catched.

3.9.2. Text Changes to the Document

Old text: (Section 1.6)

Transmission Sequence Numbers wrap around when they reach 2**32 - 1.
That is, the next TSN a DATA chunk MUST use after transmitting TSN =
2*32 - 1 is TSN = 0.

New text: (Section 1.6)

Transmission Sequence Numbers wrap around when they reach 2**32 - 1.
That is, the next TSN a DATA chunk MUST use after transmitting TSN =
2**32 - 1 is TSN = 0.

Old text: (Section 3.3.10.9)

No User Data: This error cause is returned to the originator of a

DATA chunk if a received DATA chunk has no user data.

New text: (Section 3.3.10.9)

No User Data: This error cause is returned to the originator of a
DATA chunk if a received DATA chunk has no user data.

```
---------
Old text: (Section 6.7, Figure 9)
---------

Endpoint A                                      Endpoint Z {App
sends 3 messages; strm 0} DATA [TSN=6,Strm=0,Seq=2] ----------
-----> (ack delayed) (Start T3-rtx timer)

DATA [TSN=7,Strm=0,Seq=3] --------> X (lost)

DATA [TSN=8,Strm=0,Seq=4] ----------------> (gap detected,
                                            immediately send ack)
                               /----- SACK [TSN Ack=6,Block=1,
                              /             Start=2,End=2]
                        <-----/ (remove 6 from out-queue,
 and mark 7 as "1" missing report)

---------
New text: (Section 6.7, Figure 9)
---------

Endpoint A                                      Endpoint Z
{App sends 3 messages; strm 0}
DATA [TSN=6,Strm=0,Seq=2] ----------------> (ack delayed)
(Start T3-rtx timer)

DATA [TSN=7,Strm=0,Seq=3] --------> X (lost)

DATA [TSN=8,Strm=0,Seq=4] ----------------> (gap detected,
                                            immediately send ack)
                               /----- SACK [TSN Ack=6,Block=1,
                              /             Strt=2,End=2]
                        <-----/
(remove 6 from out-queue,
 and mark 7 as "1" missing report)
```

Old text: (Section 6.10)

An endpoint bundles chunks by simply including multiple chunks in one outbound SCTP packet. The total size of the resultant IP datagram,

including the SCTP packet and IP headers, MUST be less that or equal to the current Path MTU.

New text: (Section 6.10)

An endpoint bundles chunks by simply including multiple chunks in one outbound SCTP packet. The total size of the resultant IP datagram, including the SCTP packet and IP headers, MUST be less that or equal to the current Path MTU.

Old text: (Section 10.1)
---
o Receive Unacknowledged Message

Format: RECEIVE_UNACKED(data retrieval id, buffer address, buffer size, [,stream id] [, stream sequence number] [,partial flag] [,payload protocol-id])

New text: (Section 10.1)

O) Receive Unacknowledged Message

Format: RECEIVE_UNACKED(data retrieval id, buffer address, buffer size, [,stream id] [, stream sequence number] [,partial flag] [,payload protocol-id])

```
---------
Old text: (Appendix C)
---------
ICMP2) An implementation MAY ignore all ICMPv6 messages where the
       type field is not "Destination Unreachable", "Parameter
       Problem",, or "Packet Too Big".

---------
New text: (Appendix C)
---------

ICMP2) An implementation MAY ignore all ICMPv6 messages where the
       type field is not "Destination Unreachable", "Parameter
       Problem", or "Packet Too Big".
```

3.9.3. Solution Description

Typos fixed.

3.10. CRC32c Sample Code

3.10.1. Description of the Problem

The CRC32c computation is described in Appendix B of [RFC4960]. However, the corresponding sample code and its explanation appears at the end of Appendix C, which deals with ICMP handling.

3.10.2. Text Changes to the Document

Move the sample code related to CRC32c computation and its explanation from the end of Appendix C to the end of Appendix B.

3.10.3. Solution Description

Text moved to the appropriate location.

3.11. partial_bytes_acked after T3-rtx Expiration

3.11.1. Description of the Problem

Section 7.2.3 of [RFC4960] explicitly states that partial_bytes_acked should be reset to 0 after packet loss detecting from SACK but the same is missed for T3-rtx timer expiration.

3.11.2. Text Changes to the Document

```
---------
Old text: (Section 7.2.3)
---------

When the T3-rtx timer expires on an address, SCTP should perform slow
start by:

ssthresh = max(cwnd/2, 4*MTU)
cwnd = 1*MTU

---------
New text: (Section 7.2.3)
---------

When the T3-rtx timer expires on an address, SCTP should perform slow
start by:

ssthresh = max(cwnd/2, 4*MTU)
cwnd = 1*MTU
partial_bytes_acked = 0
```

3.11.3. Solution Description

Specify that partial_bytes_acked should be reset to 0 after T3-rtx timer expiration.

3.12. Order of Adjustments of partial_bytes_acked and cwnd

3.12.1. Description of the Problem

Section 7.2.2 of [RFC4960] is unclear about the order of adjustments applied to partial_bytes_acked and cwnd in the congestion avoidance phase.

3.12.2. Text Changes to the Document

Old text: (Section 7.2.2)
---

o When partial_bytes_acked is equal to or greater than cwnd and before the arrival of the SACK the sender had cwnd or more bytes of data outstanding (i.e., before arrival of the SACK, flightsize was greater than or equal to cwnd), increase cwnd by MTU, and reset partial_bytes_acked to (partial_bytes_acked - cwnd).

New text: (Section 7.2.2)
---

o When partial_bytes_acked is equal to or greater than cwnd and before the arrival of the SACK the sender had cwnd or more bytes of data outstanding (i.e., before arrival of the SACK, flightsize was greater than or equal to cwnd), partial_bytes_acked is reset to (partial_bytes_acked - cwnd). Next, cwnd is increased by MTU.

3.12.3. Solution Description

The new text defines the exact order of adjustments of partial_bytes_acked and cwnd in the congestion avoidance phase.

3.13. HEARTBEAT ACK and the association error counter

3.13.1. Description of the Problem

Section 8.1 and Section 8.3 of [RFC4960] prescribe that the receiver of a HEARTBEAT ACK must reset the association overall error counter. In some circumstances, e.g. when a router discards DATA chunks but not HEARTBEAT chunks due to the larger size of the DATA chunk, it might be better to not clear the association error counter on reception of the HEARTBEAT ACK and reset it only on reception of the SACK to avoid stalling the association.

3.13.2. Text Changes to the Document

Old text: (Section 8.1)

The counter shall be reset each time a DATA chunk sent to that peer endpoint is acknowledged (by the reception of a SACK) or a HEARTBEAT ACK is received from the peer endpoint.

New text: (Section 8.1)

The counter shall be reset each time a DATA chunk sent to that peer endpoint is acknowledged (by the reception of a SACK). When a HEARTBEAT ACK is received from the peer endpoint, the counter should also be reset. The receiver of the HEARTBEAT ACK may choose not to clear the counter if there is outstanding data on the association. This allows for handling the possible difference in reachability based on DATA chunks and HEARTBEAT chunks.

Old text: (Section 8.3)

Upon the receipt of the HEARTBEAT ACK, the sender of the HEARTBEAT should clear the error counter of the destination transport address to which the HEARTBEAT was sent, and mark the destination transport address as active if it is not so marked. The endpoint may optionally report to the upper layer when an inactive destination address is marked as active due to the reception of the latest HEARTBEAT ACK. The receiver of the HEARTBEAT ACK must also clear the association overall error count as well (as defined in Section 8.1).

New text: (Section 8.3)

Upon the receipt of the HEARTBEAT ACK, the sender of the HEARTBEAT should clear the error counter of the destination transport address to which the HEARTBEAT was sent, and mark the destination transport address as active if it is not so marked. The endpoint may optionally report to the upper layer when an inactive destination address is marked as active due to the reception of the latest HEARTBEAT ACK. The receiver of the HEARTBEAT ACK should also clear the association overall error counter (as defined in Section 8.1).

3.13.3. Solution Description

The new text provides a possibility to not reset the association overall error counter when a HEARTBEAT ACK is received if there are valid reasons for it.

3.14. Path for Fast Retransmission

3.14.1. Description of the Problem

[RFC4960] clearly describes where to retransmit data that is timed out when the peer is multi-homed but the same is not stated for fast retransmissions.

3.14.2. Text Changes to the Document

Old text: (Section 6.4)

Furthermore, when its peer is multi-homed, an endpoint SHOULD try to retransmit a chunk that timed out to an active destination transport address that is different from the last destination address to which the DATA chunk was sent.

New text: (Section 6.4)

Furthermore, when its peer is multi-homed, an endpoint SHOULD try to retransmit a chunk that timed out to an active destination transport address that is different from the last destination address to which the DATA chunk was sent.

When its peer is multi-homed, an endpoint SHOULD send fast retransmissions to the same destination transport address where original data was sent to. If the primary path has been changed and original data was sent there before the fast retransmit, the implementation MAY send it to the new primary path.

3.14.3. Solution Description

The new text clarifies where to send fast retransmissions.

3.15. Transmittal in Fast Recovery

3.15.1. Description of the Problem

The Fast Retransmit on Gap Reports algorithm intends that only the very first packet may be sent regardless of cwnd in the Fast Recovery phase but rule 3) of [RFC4960], Section 7.2.4, misses this clarification.

3.15.2. Text Changes to the Document

Old text: (Section 7.2.4)
---

3) Determine how many of the earliest (i.e., lowest TSN) DATA chunks marked for retransmission will fit into a single packet, subject to constraint of the path MTU of the destination transport address to which the packet is being sent. Call this value K. Retransmit those K DATA chunks in a single packet. When a Fast Retransmit is being performed, the sender SHOULD ignore the value of cwnd and SHOULD NOT delay retransmission for this single packet.

New text: (Section 7.2.4)
---

3) If not in Fast Recovery, determine how many of the earliest (i.e., lowest TSN) DATA chunks marked for retransmission will fit into a single packet, subject to constraint of the path MTU of the destination transport address to which the packet is being sent. Call this value K. Retransmit those K DATA chunks in a single packet. When a Fast Retransmit is being performed, the sender SHOULD ignore the value of cwnd and SHOULD NOT delay retransmission for this single packet.

3.15.3. Solution Description

The new text explicitly specifies to send only the first packet in the Fast Recovery phase disregarding cwnd limitations.

3.16. Initial Value of ssthresh

3.16.1. Description of the Problem

The initial value of ssthresh should be set arbitrarily high. Using the advertised receiver window of the peer is inappropriate if the peer increases its window after the handshake. Furthermore, use a higher requirements level, since not following the advice may result in performance problems.

3.16.2. Text Changes to the Document

Old text: (Section 7.2.1)
---

o The initial value of ssthresh MAY be arbitrarily high (for example, implementations MAY use the size of the receiver advertised window).

New text: (Section 7.2.1)
---

o The initial value of ssthresh SHOULD be arbitrarily high (e.g., to the size of the largest possible advertised window).

3.16.3. Solution Description

Use the same value as suggested in [RFC5681], Section 3.1, as an appropriate initial value. Furthermore use the same requirements level.

3.17. Automatically Confirmed Addresses

3.17.1. Description of the Problem

The Path Verification procedure of [RFC4960] prescribes that any address passed to the sender of the INIT by its upper layer is automatically CONFIRMED. This however is unclear if only addresses in the request to initiate association establishment are considered or any addresses provided by the upper layer in any requests (e.g. in 'Set Primary').

3.17.2. Text Changes to the Document

Old text: (Section 5.4)
---

1) Any address passed to the sender of the INIT by its upper layer is automatically considered to be CONFIRMED.

New text: (Section 5.4)
---

1) Any addresses passed to the sender of the INIT by its upper layer in the request to initialize an association is automatically considered to be CONFIRMED.

3.17.3. Solution Description

The new text clarifies that only addresses provided by the upper layer in the request to initialize an association are automatically confirmed.

3.18. Only One Packet after Retransmission Timeout

3.18.1. Description of the Problem

[RFC4960] is not completely clear when it describes data transmission after T3-rtx timer expiration. Section 7.2.1 does not specify how many packets are allowed to be sent after T3-rtx timer expiration if more than one packet fit into cwnd. At the same time, Section 7.2.3 has the text without normative language saying that SCTP should ensure that no more than one packet will be in flight after T3-rtx timer expiration until successful acknowledgment. It makes the text inconsistent.

3.18.2. Text Changes to the Document

Old text: (Section 7.2.1)
---

o The initial cwnd after a retransmission timeout MUST be no more than 1*MTU.

New text: (Section 7.2.1)
---

o The initial cwnd after a retransmission timeout MUST be no more than 1*MTU and only one packet is allowed to be in flight until successful acknowledgement.

3.18.3. Solution Description

The new text clearly specifies that only one packet is allowed to be sent after T3-rtx timer expiration until successful acknowledgement.

3.19. INIT ACK Path for INIT in COOKIE-WAIT State

3.19.1. Description of the Problem

In case of an INIT received in the COOKIE-WAIT state [RFC4960] prescribes to send an INIT ACK to the same destination address to which the original INIT has been sent. This text does not address the possibility of the upper layer to provide multiple remote IP addresses while requesting the association establishment. If the upper layer has provided multiple IP addresses and only a subset of these addresses are supported by the peer then the destination address of the original INIT may be absent in the incoming INIT and sending INIT ACK to that address is useless.

3.19.2. Text Changes to the Document

Old text: (Section 5.2.1)

Upon receipt of an INIT in the COOKIE-WAIT state, an endpoint MUST respond with an INIT ACK using the same parameters it sent in its original INIT chunk (including its Initiate Tag, unchanged). When responding, the endpoint MUST send the INIT ACK back to the same address that the original INIT (sent by this endpoint) was sent.

New text: (Section 5.2.1)

Upon receipt of an INIT in the COOKIE-WAIT state, an endpoint MUST respond with an INIT ACK using the same parameters it sent in its original INIT chunk (including its Initiate Tag, unchanged). When responding, the following rules MUST be applied:

1) The INIT ACK MUST only be sent to an address passed by the upper layer in the request to initialize the association.

2) The INIT ACK MUST only be sent to an address reported in the incoming INIT.

3) The INIT ACK SHOULD be sent to the source address of the received INIT.

3.19.3. Solution Description

The new text requires sending INIT ACK to the destination address that is passed by the upper layer and reported in the incoming INIT. If the source address of the INIT fulfills it then sending the INIT ACK to the source address of the INIT is the preferred behavior.

3.20. Zero Window Probing and Unreachable Primary Path

3.20.1. Description of the Problem

Section 6.1 of [RFC4960] states that when sending zero window probes, SCTP should neither increment the association counter nor increment the destination address error counter if it continues to receive new packets from the peer. But receiving new packets from the peer does not guarantee peer's accessibility and, if the destination address becomes unreachable during zero window probing, SCTP cannot get a changed rwnd until it switches the destination address for probes.

3.20.2. Text Changes to the Document

Old text: (Section 6.1)

If the sender continues to receive new packets from the receiver while doing zero window probing, the unacknowledged window probes should not increment the error counter for the association or any destination transport address. This is because the receiver MAY keep its window closed for an indefinite time. Refer to Section 6.2 on the receiver behavior when it advertises a zero window.

New text: (Section 6.1)

If the sender continues to receive SACKs from the peer while doing zero window probing, the unacknowledged window probes should not increment the error counter for the association or any destination transport address. This is because the receiver MAY keep its window closed for an indefinite time. Refer to Section 6.2 on the receiver behavior when it advertises a zero window.

3.20.3. Solution Description

The new text clarifies that if the receiver continues to send SACKs, the sender of probes should not increment the error counter of the association and the destination address even if the SACKs do not acknowledge the probes.

3.21. Normative Language in Section 10

3.21.1. Description of the Problem

Section 10 of [RFC4960] is informative and normative language such as MUST and MAY cannot be used there. However, there are several places in Section 10 where MUST and MAY are used.

3.21.2. Text Changes to the Document

Old text: (Section 10.1)

E) Send

Format: SEND(association id, buffer address, byte count [,context]

```
          [,stream id] [,life time] [,destination transport address]
          [,unordered flag] [,no-bundle flag] [,payload protocol-id] )
 -> result

...

o  no-bundle flag - instructs SCTP not to bundle this user data with
   other outbound DATA chunks.  SCTP MAY still bundle even when this
   flag is present, when faced with network congestion.

---------
New text: (Section 10.1)
---------

E) Send

 Format: SEND(association id, buffer address, byte count [,context]
          [,stream id] [,life time] [,destination transport address]
          [,unordered flag] [,no-bundle flag] [,payload protocol-id] )
 -> result

...

o  no-bundle flag - instructs SCTP not to bundle this user data with
   other outbound DATA chunks.  SCTP may still bundle even when this
   flag is present, when faced with network congestion.

---------
Old text: (Section 10.1)
---------

G) Receive

 Format: RECEIVE(association id, buffer address, buffer size
          [,stream id])
 -> byte count [,transport address] [,stream id] [,stream sequence
    number] [,partial flag] [,delivery number] [,payload protocol-id]

...

o  partial flag - if this returned flag is set to 1, then this
   Receive contains a partial delivery of the whole message.  When
   this flag is set, the stream id and Stream Sequence Number MUST
   accompany this receive.  When this flag is set to 0, it indicates
   that no more deliveries will be received for this Stream Sequence
   Number.

---------
```

New text: (Section 10.1)

G) Receive

Format: RECEIVE(association id, buffer address, buffer size
[,stream id])
-> byte count [,transport address] [,stream id] [,stream sequence
number] [,partial flag] [,delivery number] [,payload protocol-id]

...

o partial flag - if this returned flag is set to 1, then this
Receive contains a partial delivery of the whole message. When
this flag is set, the stream id and Stream Sequence Number must
accompany this receive. When this flag is set to 0, it indicates
that no more deliveries will be received for this Stream Sequence
Number.

Old text: (Section 10.1)

N) Receive Unsent Message

Format: RECEIVE_UNSENT(data retrieval id, buffer address, buffer
size [,stream id] [, stream sequence number] [,partial
flag] [,payload protocol-id])

...

o partial flag - if this returned flag is set to 1, then this
message is a partial delivery of the whole message. When this
flag is set, the stream id and Stream Sequence Number MUST
accompany this receive. When this flag is set to 0, it indicates
that no more deliveries will be received for this Stream Sequence
Number.

New text: (Section 10.1)

N) Receive Unsent Message

Format: RECEIVE_UNSENT(data retrieval id, buffer address, buffer
size [,stream id] [, stream sequence number] [,partial
flag] [,payload protocol-id])

...

o partial flag - if this returned flag is set to 1, then this message is a partial delivery of the whole message. When this flag is set, the stream id and Stream Sequence Number must accompany this receive. When this flag is set to 0, it indicates that no more deliveries will be received for this Stream Sequence Number.

Old text: (Section 10.1)

O) Receive Unacknowledged Message

Format: RECEIVE_UNACKED(data retrieval id, buffer address, buffer size, [,stream id] [, stream sequence number] [,partial flag] [,payload protocol-id])

...

o partial flag - if this returned flag is set to 1, then this message is a partial delivery of the whole message. When this flag is set, the stream id and Stream Sequence Number MUST accompany this receive. When this flag is set to 0, it indicates that no more deliveries will be received for this Stream Sequence Number.

New text: (Section 10.1)

O) Receive Unacknowledged Message

Format: RECEIVE_UNACKED(data retrieval id, buffer address, buffer size, [,stream id] [, stream sequence number] [,partial flag] [,payload protocol-id])

...

o partial flag - if this returned flag is set to 1, then this message is a partial delivery of the whole message. When this flag is set, the stream id and Stream Sequence Number must accompany this receive. When this flag is set to 0, it indicates that no more deliveries will be received for this Stream Sequence Number.

=======
Old text: (Section 7.2.2)

---

o Whenever cwnd is greater than ssthresh, upon each SACK arrival that advances the Cumulative TSN Ack Point, increase partial_bytes_acked by the total number of bytes of all new chunks acknowledged in that SACK including chunks acknowledged by the new Cumulative TSN Ack and by Gap Ack Blocks.

New text: (Section 7.2.2)
---

o Whenever cwnd is greater than ssthresh, upon each SACK arrival, increase partial_bytes_acked by the total number of bytes of all new chunks acknowledged in that SACK including chunks acknowledged by the new Cumulative TSN Ack, by Gap Ack Blocks and by the number of bytes of duplicated chunks reported in Duplicate TSNs.

3.21.3. Solution Description

The normative language is removed from Section 10.

3.22. Increase of partial_bytes_acked in Congestion Avoidance

3.22.1. Description of the Problem

Two issues have been discovered with the partial_bytes_acked handling described in Section 7.2.2 of [RFC4960]:

o If the Cumulative TSN Ack Point is not advanced but the SACK chunk acknowledges new TSNs in the Gap Ack Blocks, these newly acknowledged TSNs are not considered for partial_bytes_acked although these TSNs were successfully received by the peer.
o Duplicate TSNs are not considered in partial_bytes_acked although they confirm that the DATA chunks were successfully received by the peer.

3.22.2. Text Changes to the Document

Old text: (Section 7.2.2)
---

o Whenever cwnd is greater than ssthresh, upon each SACK arrival that advances the Cumulative TSN Ack Point, increase partial_bytes_acked by the total number of bytes of all new chunks acknowledged in that SACK including chunks acknowledged by the new Cumulative TSN Ack and by Gap Ack Blocks.

New text: (Section 7.2.2)
---

o Whenever cwnd is greater than ssthresh, upon each SACK arrival, increase partial_bytes_acked by the total number of bytes of all new chunks acknowledged in that SACK including chunks acknowledged by the new Cumulative TSN Ack, by Gap Ack Blocks and by the number of bytes of duplicated chunks reported in Duplicate TSNs.

3.22.3. Solution Description

Now partial_bytes_acked is increased by TSNs reported as duplicated as well as TSNs newly acknowledged in Gap Ack Blocks even if the Cumulative TSN Ack Point is not advanced.

3.23. Inconsistency in Notifications Handling

3.23.1. Description of the Problem

[RFC4960] uses inconsistent normative and non-normative language when describing rules for sending notifications to the upper layer. E.g. Section 8.2 of [RFC4960] says that when a destination address becomes inactive due to an unacknowledged DATA chunk or HEARTBEAT chunk, SCTP SHOULD send a notification to the upper layer while Section 8.3 of [RFC4960] says that when a destination address becomes inactive due to an unacknowledged HEARTBEAT chunk, SCTP may send a notification to the upper layer.

This makes the text inconsistent.

3.23.2. Text Changes to the Document

The following cahnge is based on the change described in Section 3.6.

Old text: (Section 8.1)

An endpoint shall keep a counter on the total number of consecutive retransmissions to its peer (this includes data retransmissions to all the destination transport addresses of the peer if it is multi-homed), including the number of unacknowledged HEARTBEAT chunks observed on the path which currently is used for data transfer. Unacknowledged HEARTBEAT chunks observed on paths different from the path currently used for data transfer shall not increment the association error counter, as this could lead to association closure even if the path which currently is used for data transfer is available (but idle). If the value of this counter exceeds the limit indicated in the protocol parameter 'Association.Max.Retrans', the endpoint shall consider the peer endpoint unreachable and shall stop transmitting any more data to it (and thus the association enters the CLOSED state). In addition, the endpoint MAY report the failure to the upper layer and optionally report back all outstanding user data remaining in its outbound queue. The association is automatically closed when the peer endpoint becomes unreachable.

New text: (Section 8.1)

An endpoint shall keep a counter on the total number of consecutive retransmissions to its peer (this includes data retransmissions to all the destination transport addresses of the peer if it is multi-homed), including the number of unacknowledged HEARTBEAT chunks observed on the path which currently is used for data transfer. Unacknowledged HEARTBEAT chunks observed on paths different from the path currently used for data transfer shall not increment the association error counter, as this could lead to association closure even if the path which currently is used for data transfer is available (but idle). If the value of this counter exceeds the limit indicated in the protocol parameter 'Association.Max.Retrans', the endpoint shall consider the peer endpoint unreachable and shall stop transmitting any more data to it (and thus the association enters the CLOSED state). In addition, the endpoint SHOULD report the failure to the upper layer and optionally report back all outstanding user data remaining in its outbound queue. The association is automatically closed when the peer endpoint becomes unreachable.

The following changes are based on [RFC4960].

Old text: (Section 8.2)

When an outstanding TSN is acknowledged or a HEARTBEAT sent to that address is acknowledged with a HEARTBEAT ACK, the endpoint shall clear the error counter of the destination transport address to which the DATA chunk was last sent (or HEARTBEAT was sent). When the peer endpoint is multi-homed and the last chunk sent to it was a retransmission to an alternate address, there exists an ambiguity as to whether or not the acknowledgement should be credited to the address of the last chunk sent. However, this ambiguity does not seem to bear any significant consequence to SCTP behavior. If this ambiguity is undesirable, the transmitter may choose not to clear the error counter if the last chunk sent was a retransmission.

New text: (Section 8.2)

When an outstanding TSN is acknowledged or a HEARTBEAT sent to that address is acknowledged with a HEARTBEAT ACK, the endpoint shall clear the error counter of the destination transport address to which the DATA chunk was last sent (or HEARTBEAT was sent), and SHOULD also report to the upper layer when an inactive destination address is marked as active. When the peer endpoint is multi-homed and the last chunk sent to it was a retransmission to an alternate address, there exists an ambiguity as to whether or not the acknowledgement should be credited to the address of the last chunk sent. However, this ambiguity does not seem to bear any significant consequence to SCTP behavior. If this ambiguity is undesirable, the transmitter may choose not to clear the error counter if the last chunk sent was a retransmission.

Old text: (Section 8.3)

When the value of this counter reaches the protocol parameter 'Path.Max.Retrans', the endpoint should mark the corresponding destination address as inactive if it is not so marked, and may also optionally report to the upper layer the change of reachability of this destination address. After this, the endpoint should continue HEARTBEAT on this destination address but should stop increasing the counter.

New text: (Section 8.3)

When the value of this counter exceeds the protocol parameter 'Path.Max.Retrans', the endpoint should mark the corresponding destination address as inactive if it is not so marked, and SHOULD also report to the upper layer the change of reachability of this destination address. After this, the endpoint should continue HEARTBEAT on this destination address but should stop increasing the counter.

Old text: (Section 8.3)

Upon the receipt of the HEARTBEAT ACK, the sender of the HEARTBEAT should clear the error counter of the destination transport address to which the HEARTBEAT was sent, and mark the destination transport address as active if it is not so marked. The endpoint may optionally report to the upper layer when an inactive destination address is marked as active due to the reception of the latest HEARTBEAT ACK. The receiver of the HEARTBEAT ACK must also clear the association overall error count as well (as defined in Section 8.1).

New text: (Section 8.3)

Upon the receipt of the HEARTBEAT ACK, the sender of the HEARTBEAT should clear the error counter of the destination transport address to which the HEARTBEAT was sent, and mark the destination transport address as active if it is not so marked. The endpoint SHOULD report to the upper layer when an inactive destination address is marked as active due to the reception of the latest HEARTBEAT ACK. The receiver of the HEARTBEAT ACK should also clear the association overall error counter (as defined in Section 8.1).

Old text: (Section 9.2)

An endpoint should limit the number of retransmissions of the SHUTDOWN chunk to the protocol parameter 'Association.Max.Retrans'. If this threshold is exceeded, the endpoint should destroy the TCB and MUST report the peer endpoint unreachable to the upper layer (and thus the association enters the CLOSED state).

New text: (Section 9.2)

An endpoint should limit the number of retransmissions of the SHUTDOWN chunk to the protocol parameter 'Association.Max.Retrans'. If this threshold is exceeded, the endpoint should destroy the TCB and SHOULD report the peer endpoint unreachable to the upper layer (and thus the association enters the CLOSED state).

Old text: (Section 9.2)

The sender of the SHUTDOWN ACK should limit the number of retransmissions of the SHUTDOWN ACK chunk to the protocol parameter 'Association.Max.Retrans'. If this threshold is exceeded, the endpoint should destroy the TCB and may report the peer endpoint unreachable to the upper layer (and thus the association enters the CLOSED state).

New text: (Section 9.2)

The sender of the SHUTDOWN ACK should limit the number of retransmissions of the SHUTDOWN ACK chunk to the protocol parameter 'Association.Max.Retrans'. If this threshold is exceeded, the endpoint should destroy the TCB and SHOULD report the peer endpoint unreachable to the upper layer (and thus the association enters the CLOSED state).

3.23.3. Solution Description

The inconsistencies are removed by using consistently SHOULD.

4. IANA Considerations

This documents does not require any actions from IANA.

5. Security Considerations

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

6. Acknowledgments

The authors wish to thank Pontus Andersson, Eric W. Biederman, Jeff Morriss, Tom Petch and Julien Pourtet for their invaluable comments.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <http://www.rfc-editor.org/info/rfc4960>.

7.2. Informative References

[RFC2960] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L., and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, DOI 10.17487/RFC2960, October 2000, <http://www.rfc-editor.org/info/rfc2960>.

[RFC4460] Stewart, R., Arias-Rodriguez, I., Poon, K., Caro, A., and M. Tuexen, "Stream Control Transmission Protocol (SCTP) Specification Errata and Issues", RFC 4460, DOI 10.17487/RFC4460, April 2006, <http://www.rfc-editor.org/info/rfc4460>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <http://www.rfc-editor.org/info/rfc5681>.

Authors' Addresses

Randall R. Stewart
Netflix, Inc.
Chapin, SC 29036
United States

Email: randall@lakerest.net

Michael Tuexen
Muenster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt
Germany

Email: tuexen@fh-muenster.de

Karen E. E. Nielsen
Ericsson
Kistavaegen 25
Stockholm 164 80
Sweden

Email: karen.nielsen@tieto.com

Maksim Proshin
Ericsson
Kistavaegen 25
Stockholm 164 80
Sweden

Email: mproshin@tieto.mera.ru