Network Working Group
Request for Comments: 3758
Category: Standards Track

R. Stewart
M. Ramalho
Cisco Systems, Inc.
Q. Xie
Motorola, Inc.
M. Tuexen
Univ. of Applied Sciences Muenster
P. Conrad
University of Delaware

May 2004

Stream Control Transmission Protocol (SCTP) Partial Reliability Extension

Status of this Memo

Copyright Notice

Abstract

This memo describes an extension to the Stream Control Transmission Protocol (SCTP) that allows an SCTP endpoint to signal to its peer that it should move the cumulative ack point forward. When both sides of an SCTP association support this extension, it can be used by an SCTP implementation to provide partially reliable data transmission service to an upper layer protocol. This memo describes the protocol extensions, which consist of a new parameter for INIT and INIT ACK, and a new FORWARD TSN chunk type, and provides one example of a partially reliable service that can be provided to the upper layer via this mechanism.

Table of Contents

1. Introduction

This memo describes an extension to the Stream Control Transmission Protocol (SCTP) RFC 2960 [2] that allows an SCTP sender to signal to its peer that it should no longer expect to receive one or more DATA chunks.

1.1. Overview of Protocol Extensions

The protocol extension described in this document consists of two new elements:

1. a single new parameter in the INIT/INIT-ACK exchange that indicates whether the endpoint supports the extension

2. a single new chunk type, FORWARD TSN, that indicates that the receiver should move its cumulative ack point forward (possibly skipping past one or more DATA chunks that may not yet have been received and/or acknowledged.)

1.2. Overview of New Services Provided to the Upper Layer

When this extension is supported by both sides of an SCTP association, it can be used to provide partially reliable transport service over an SCTP association. We define partially reliable transport service as a service that allows the user to specify, on a per message basis, the rules governing how persistent the transport service should be in attempting to send the message to the receiver.

One example of partially reliable service is specified in this document, namely a "timed reliability" service. This service allows the service user to indicate a limit on the duration of time that the sender should try to transmit/retransmit the message (this is a natural extension of the "lifetime" parameter already in the base protocol).

In addition to this example, we will also show that defining the semantics of a particular partially reliable service involves two elements, namely:

1. how the service user indicates the level of reliability required for a particular message, and

2. how the sender side implementation uses that reliability level to determine when to give up on further retransmissions of that message.

Note that other than the fact that the FORWARD-TSN chunk is required, neither of these two elements impacts the "on-the-wire" protocol; only the API and the sender side implementation are affected by the way in which the service is defined to the upper layer. Therefore, in principle, it is feasible to implement many varieties of partially reliable services in a particular SCTP implementation without changing the on-the-wire protocol. Also, the SCTP receiver does not necessarily need to know which semantics of partially reliable service are being used by the sender, since the receiver's only role is to correctly interpret FORWARD TSN chunks, thereby skipping past messages that the sender has decided to no longer transmit (or retransmit).

Nevertheless, it is recommended that a limited number of standard definitions of partially reliable services be standardized by the IETF so that the designers of IETF application layer protocols can match the requirements of their upper layer protocols to standard service definitions provided by a particular SCTP implementation. One such definition, "timed reliability", is included in this document. Given the extensions proposed in this document, other definitions may be standardized as the need arises without further changes to the on-the-wire protocol.

1.3. Benefits of PR-SCTP

Hereafter, we use the notation "Partial Reliable Stream Control Transmission Protocol (PR-SCTP)" to refer to the SCTP protocol, extended as defined in this document.

The following are some of the advantages for integrating partially reliable data service into SCTP, i.e., benefits of PR-SCTP:

1. Some application layer protocols may benefit from being able to use a single SCTP association to carry both reliable content, -- such as text pages, billing and accounting information, setup signaling -- and unreliable content, e.g., state that is highly sensitive to timeliness, where generating a new packet is more advantageous than transmitting an old one [3].

2. Partially reliable data traffic carried by PR-SCTP will enjoy the same communication failure detection and protection capabilities as the normal reliable SCTP data traffic does. This includes the ability to quickly detect a failed destination address, fail-over to an alternate destination address, and be notified if the data receiver becomes unreachable.

3. In addition to providing unordered, unreliable data transfer as UDP does, PR-SCTP can provide ordered, unreliable data transfer service.

4. PR-SCTP employs the same congestion control and congestion avoidance for all data traffic, whether reliable or partially reliable - this is very desirable since SCTP enforces TCP-friendliness (unlike UDP.)

5. Because of the chunk bundling function of SCTP, reliable and unreliable messages can be multiplexed over a single PR-SCTP association. Therefore, the number of IP datagrams (and hence the network overhead) can be reduced instead of having to send these different types of data using separate protocols. Additionally, this multiplexing allows for port savings versus using different ports for reliable and unreliable connections.

2. Conventions

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, NOT RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in BCP 14, RFC 2119 [1].

Comparisons and arithmetic on Transport Sequence Numbers (TSNs) are governed by the rules in Section 1.6 of RFC 2960 [2].

3. Protocol Changes to support PR-SCTP

3.1. Forward-TSN-Supported Parameter For INIT and INIT ACK

The following new OPTIONAL parameter is added to the INIT and INIT ACK chunks.

Parameter Name	Status	Type Value
Forward-TSN-Supported	OPTIONAL	49152 (0xC000)

At the initialization of the association, the sender of the INIT or INIT ACK chunk MAY include this OPTIONAL parameter to inform its peer that it is able to support the Forward TSN chunk (see Section 3.3 for further details). The format of this parameter is defined as follows:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Parameter Type = 49152     |  Parameter Length = 4         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type: 16 bit u_int

49152, indicating Forward-TSN-Supported parameter

Length: 16 bit u_int

Indicates the size of the parameter, i.e., 4.

3.2 Forward Cumulative TSN Chunk Definition (FORWARD TSN)

The following new chunk type is defined:

Chunk Type	Chunk Name
192 (0xC0)	Forward Cumulative TSN (FORWARD TSN)

This chunk shall be used by the data sender to inform the data receiver to adjust its cumulative received TSN point forward because some missing TSNs are associated with data chunks that SHOULD NOT be transmitted or retransmitted by the sender.

Forward Cumulative TSN chunk has the following format:

```
 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 = 192  |  Flags = 0x00 |        Length = Variable      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      New Cumulative TSN                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Stream-1              |       Stream Sequence-1       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                                                               /
/                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Stream-N              |       Stream Sequence-N       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Chunk Flags:

Set to all zeros on transmit and ignored on receipt.

New Cumulative TSN: 32 bit u_int

This indicates the new cumulative TSN to the data receiver. Upon the reception of this value, the data receiver MUST consider any missing TSNs earlier than or equal to this value as received, and stop reporting them as gaps in any subsequent SACKs.

Stream-N: 16 bit u_int

This field holds a stream number that was skipped by this FWD-TSN.

Stream Sequence-N: 16 bit u_int

This field holds the sequence number associated with the stream that was skipped. The stream sequence field holds the largest stream sequence number in this stream being skipped. The receiver of the FWD-TSN’s can use the Stream-N and Stream Sequence-N fields to enable delivery of any stranded TSN’s that remain on the stream re-ordering queues. This field MUST NOT report TSN’s corresponding to DATA chunks that are marked as unordered. For ordered DATA chunks this field MUST be filled in.

3.3. Negotiation of Forward-TSN-Supported parameter

3.3.1. Sending Forward-TSN-Supported param in INIT

If an SCTP endpoint supports the FORWARD TSN chunk, then any time it sends an INIT during association establishment, it MAY include the Forward-TSN-supported parameter in the INIT chunk to indicate this fact to its peer.

Note that if the endpoint chooses NOT to include the parameter, then at no time during the life of the association can it send or process a FORWARD TSN. It MUST instead act as if it does NOT support the FORWARD TSN chunk, returning an ERROR to the peer upon receipt of any FORWARD TSN.

3.3.2. Receipt of Forward-TSN-Supported parameter in INIT or INIT-ACK

When a receiver of an INIT detects a Forward-TSN-Supported parameter and does not support the Forward-TSN chunk type, the receiver MUST follow the rules defined in Section 3.3.3 of RFC 2960 [2].

When a receiver of an INIT-ACK detects a Forward-TSN-Supported parameter and it does not support the Forward-TSN chunk type, the receiver MUST follow the rules defined in Section 3.3.3 of RFC 2960 [2].

When a receiver of an INIT detects a Forward-TSN-Supported parameter and it does support the Forward-TSN chunk type, the receiver MAY respond with a Forward-TSN-supported parameter in the INIT-ACK chunk.

Note that if the endpoint chooses NOT to include the parameter, then at no time during the life of the association can it send or process a FORWARD TSN. It MUST instead act as if it does NOT support the FORWARD TSN chunk, returning an ERROR to the peer upon receipt of any FORWARD TSN.

When an endpoint that supports the FORWARD TSN chunk receives an INIT that does not contain the Forward-TSN-Supported Parameter, that endpoint:

- o MAY include the Forward-TSN-Supported parameter in the INIT-ACK,
- o SHOULD record the fact that the peer does not support the FORWARD TSN chunk,
- o MUST NOT send a FORWARD TSN chunk at any time during the associations life,
- o SHOULD inform the upper layer if the upper layer has requested such notification.

3.3.3. Receipt of Op. Error for Forward-TSN-Supported Param

When an SCTP endpoint that desires to use the FORWARD TSN chunk feature for partially reliable data transfer receives an operational error from the remote endpoint (either bundled with the COOKIE or as an unrecognized parameter in the INIT-ACK), indicating that the remote endpoint does not recognize the Forward-TSN-Supported parameter, the local endpoint SHOULD inform its upper layer of the remote endpoint’s inability to support partially reliable data transfer.

The local endpoint may then choose to either:

1) end the initiation process (in cases where the initiation process has already ended, the endpoint may need to send an ABORT) in consideration of the peer’s inability to supply the requested features for the new association, or

2) continue the initiation process (in cases where the initiation process has already completed, the endpoint MUST just mark the association as not supporting partial reliability), but with the understanding that partially reliable data transmission is not supported. In this case, the endpoint receiving the operational error SHOULD note that the FORWARD TSN chunk is not supported, and MUST NOT transmit a FORWARD TSN chunk at any time during the life of the association.

3.4. Definition of "abandoned" in the context of PR-SCTP

At some point, a sending PR-SCTP implementation MAY determine that a particular data chunk SHOULD NOT be transmitted or retransmitted further, in accordance with the rules governing some particular PR-SCTP service definition (such as the definition of "timed reliability" in Section 4.1.) For purposes of this document, we define the term "abandoned" to refer to any data chunk about which the SCTP sender has made this determination.

Each PR-SCTP service defines the rules for determining when a TSN is "abandoned", and accordingly, the rules that govern how, whether, and when to "abandon" a TSN may vary from one service definition to another. However, the rules governing the actions taken when a TSN is "abandoned" do NOT vary between service definitions; these rules are included in Section 3.5.

3.5. Sender Side Implementation of PR-SCTP

The sender side implementation of PR-SCTP is identical to that of the base SCTP protocol, except for:

- o actions a sending side PR-SCTP implementation must take when a TSN is "abandoned" (as per the rules of whatever PR-SCTP service definition is in effect)
- o special actions that a PR-SCTP implementation must take upon receipt of SACK
- o rules governing the generation of FORWARD TSN chunks.

In detail, these exceptions are as follows:

A1) The sender maintains an "Advanced.Peer.Ack.Point" for each peer to track a theoretical cumulative TSN point of the peer (Note, this is a _new_ protocol variable and its value is NOT necessarily the same as the SCTP "Cumulative TSN Ack Point" as defined in Section 1.4 of RFC 2960 [2], and as discussed throughout that document.)

A2) From time to time, as governed by the rules of a particular PR-SCTP service definition (see Section 4), the SCTP data sender may make a determination that a particular data chunk that has already been assigned a TSN SHOULD be "abandoned".

When a data chunk is "abandoned", the sender MUST treat the data chunk as being finally acked and no longer outstanding.

The sender MUST NOT credit an "abandoned" data chunk to the partial_bytes_acked as defined in Section 7.2.2 of RFC 2960 [2], and MUST NOT advance the cwnd based on this "abandoned" data chunk.

A3) When a TSN is "abandoned", if it is part of a fragmented message, all other TSN's within that fragmented message MUST be abandoned at the same time.

A4) Whenever the data sender receives a SACK from the data receiver, it MUST first process the SACK using the normal procedures as defined in Section 6.2.1 of RFC 2960 [2].

The data sender MUST then perform the following additional steps:

C1) Let SackCumAck be the Cumulative TSN ACK carried in the received SACK.

If (Advanced.Peer.Ack.Point < SackCumAck), then update Advanced.Peer.Ack.Point to be equal to SackCumAck.

C2) Try to further advance the "Advanced.Peer.Ack.Point" locally, that is, to move "Advanced.Peer.Ack.Point" up as long as the chunk next in the out-queue space is marked as "abandoned", as shown in the following example:

Assuming that a SACK arrived with the Cumulative TSN ACK = 102 and the Advanced.Peer.Ack.Point is updated to this value:

```
out-queue at the end of  ==>   out-queue after Adv.Ack.Point
normal SACK processing         local advancement

             ...                          ...
Adv.Ack.Pt-> 102 acked                    102 acked
             103 abandoned                  103 abandoned
             104 abandoned        Adv.Ack.P-> 104 abandoned
             105                          105
             106 acked                    106 acked
             ...                          ...
```

In this example, the data sender successfully advanced the "Advanced.Peer.Ack.Point" from 102 to 104 locally.

C3) If, after step C1 and C2, the "Advanced.Peer.Ack.Point" is greater than the Cumulative TSN ACK carried in the received SACK, the data sender MUST send the data receiver a FORWARD TSN chunk containing the latest value of the "Advanced.Peer.Ack.Point". Note that the sender MAY delay the sending of a FORWARD TSN as defined in rule F2 below. IMPLEMENTATION NOTE: It is an implementation decision as to which destination address it is to be sent to, the only restriction being that the address MUST be one that is CONFIRMED.

C4) For each "abandoned" TSN, the sender of the FORWARD TSN MUST determine if the chunk has a valid stream and sequence number (i.e., it was ordered). If the chunk has a valid stream and sequence number, the sender MUST include the stream and sequence number in the FORWARD TSN. This information will enable the receiver to easily find any stranded TSN’s waiting on stream reorder queues. Each stream SHOULD only be reported once; this means that if multiple abandoned messages occur in the same stream, then only the highest abandoned stream sequence number is reported. If the total size of the FORWARD TSN does NOT fit in a single MTU, then the sender of the FORWARD TSN SHOULD lower the Advanced.Peer.Ack.Point to the last TSN that will fit in a single MTU.

C5) If a FORWARD TSN is sent, the sender MUST assure that at least one T3-rtx timer is running. IMPLEMENTATION NOTE: Any destination's timer may be used for the purposes of rule C5.

A5) Any time the T3-rtx timer expires, on any destination, the sender SHOULD try to advance the "Advanced.Peer.Ack.Point" by following the procedures outlined in C2 - C5.

The following additional rules govern the generation of FORWARD TSN chunks:

F1) An endpoint MUST NOT use the FORWARD TSN for any purposes other than circumstances described in this document.

F2) The data sender SHOULD always attempt to bundle an outgoing FORWARD TSN with outbound DATA chunks for efficiency.

A sender MAY even choose to delay the sending of the FORWARD TSN in the hope of bundling it with an outbound DATA chunk.

IMPLEMENTATION NOTE: An implementation may wish to limit the number of duplicate FORWARD TSN chunks it sends by either only sending a duplicate FORWARD TSN every other SACK or waiting a full RTT before sending a duplicate FORWARD TSN.

IMPLEMENTATION NOTE: An implementation may allow the maximum delay for generating a FORWARD TSN to be configured either statically or dynamically in order to meet the specific timing requirements of the protocol being carried, but see the next rule:

F3) Any delay applied to the sending of FORWARD TSN chunk SHOULD NOT exceed 200ms and MUST NOT exceed 500ms. In other words, an implementation MAY lower this value below 500ms but MUST NOT raise it above 500ms.

NOTE: Delaying the sending of FORWARD TSN chunks may cause delays in the receiver's ability to deliver other data being held at the receiver for re-ordering. The values of 200ms and 500ms match the required values for the delayed acknowledgement in RFC 2960 [2] since delaying a FORWARD TSN has the same consequences but in the reverse direction.

F4) The detection criterion for out-of-order SACKs MUST remain the same as stated in RFC 2960, that is, a SACK is only considered out-of-order if the Cumulative TSN ACK carried in the SACK is earlier than that of the previous received SACK (i.e., the comparison MUST NOT be made against "Advanced.Peer.Ack.Point").

F5) If the decision to "abandon" a chunk is made, no matter how such a decision is made, the appropriate congestion adjustment MUST be made as specified in RFC 2960 if the chunk would have been marked for retransmission later (e.g., either by T3-Timeout or by Fast Retransmit).

3.6. Receiver Side Implementation of PR-SCTP

The receiver side implementation of PR-SCTP at an SCTP endpoint A is capable of supporting any PR-SCTP service definition used by the sender at endpoint B, even if that service definition is not supported by the sending side functionality of host A. All that is necessary is that the receiving side correctly handle the Forward-TSN-Supported parameter as specified in Section 3.3, and correctly handle the receipt of FORWARD TSN chunks as specified below.

DATA chunk arrival at a PR-SCTP receiver proceeds exactly as for DATA chunk arrival at a base protocol SCTP receiver---that is, the receiver MUST perform the same TSN handling, including duplicate detection, gap detection, SACK generation, cumulative TSN advancement, etc. as defined in RFC 2960 [2]---with the following exceptions and additions.

When a FORWARD TSN chunk arrives, the data receiver MUST first update its cumulative TSN point to the value carried in the FORWARD TSN chunk, and then MUST further advance its cumulative TSN point locally if possible, as shown by the following example:

Assuming that the new cumulative TSN carried in the arrived FORWARD TSN is 103:

```
 in-queue before processing      in-queue after processing
      the FORWARD TSN      ==>   the FORWARD TSN and further
                                          advancement
```

```
cum.TSN.Pt-> 102 received                   102 --
             103 missing                    103 --
             104 received                   104 --
             105 received      cum.TSN.Pt-> 105 received
             106 missing                    106 missing
             107 received                   107 received
             ...                            ...
```

In this example, the receiver's cumulative TSN point is first updated to 103 and then further advanced to 105.

After the above processing, the data receiver MUST stop reporting any missing TSNs earlier than or equal to the new cumulative TSN point.

Note, if the "New Cumulative TSN" value carried in the arrived FORWARD TSN chunk is found to be behind or at the current cumulative TSN point, the data receiver MUST treat this FORWARD TSN as out-of-date and MUST NOT update its Cumulative TSN. The receiver SHOULD send a SACK to its peer (the sender of the FORWARD TSN) since such a duplicate may indicate the previous SACK was lost in the network.

Any time a FORWARD TSN chunk arrives, for the purposes of sending a SACK, the receiver MUST follow the same rules as if a DATA chunk had been received (i.e., follow the delayed sack rules specified in RFC 2960 [2] section 6.2).

Whenever a DATA chunk arrives with the 'U' bit set to '0' (indicating ordered delivery) and is out of order, the receiver must hold the chunk for reordering. Since it is possible with PR-SCTP that a DATA chunk being waited upon will not be retransmitted, special actions will need to be taken upon the arrival of a FORWARD TSN.

In particular, during processing of a FORWARD TSN, the receiver MUST use the stream sequence information to examine all of the listed stream reordering queues, and immediately make available for delivery stream sequence numbers earlier than or equal to the stream sequence number listed inside the FORWARD TSN. Any such stranded data SHOULD be made immediately available to the upper layer application.

An application using PR-SCTP receiving data should be aware of possible missing messages. The stream sequence number can be used, in such a case, to determine that an intervening message has been skipped. When intervening messages are missing, it is an application decision to process the messages or to take some other corrective action.

After receiving and processing a FORWARD TSN, the data receiver MUST take cautions in updating its re-assembly queue. The receiver MUST remove any partially reassembled message, which is still missing one or more TSNs earlier than or equal to the new cumulative TSN point. In the event that the receiver has invoked the partial delivery API, a notification SHOULD also be generated to inform the upper layer API that the message being partially delivered will NOT be completed.

Note that after receiving a FORWARD TSN and updating the cumulative acknowledgement point, if a TSN that was skipped does arrive (i.e., due to network reordering), then the receiver will follow the normal rules defined in RFC 2960 [2] for handling duplicate data. This implies that the receiver will drop the chunk and report it as a duplicate in the next outbound SACK chunk.

4. Services provided by PR-SCTP to the upper layer

As described in Section 1.2, it is feasible to implement a variety of partially reliable transport services using the new protocol mechanisms introduced in Section 3; introducing these new services requires making changes only at the sending side API, and the sending side protocol implementation. Thus, there may be a temptation to standardize only the protocol, and leave the service definition as "implementation specific" or leave it to be defined in "informational" documents.

However, for those who may wish to write IETF standards for upper layer protocols implemented over PR-SCTP, it is important to be able to refer to a standard definition of services provided. Therefore, this section provides example definitions of one such service, while also providing guidelines for the definition of additional services as required. Each such service may be proposed as a separate new RFC.

Section 4 is organized as follows:

- o Section 4.1 provides the definition of one specific PR-SCTP service: timed reliability.

- o Section 4.2 describes how a particular PR-SCTP service definition is requested by the upper layer during association establishment, and how the upper layer is notified if that request cannot be satisfied.

- o Section 4.3 then provides guidelines for the specification of PR-SCTP services other then the one defined in this memo.

- o Finally, Section 4.4 describes some additional usage notes that upper layer protocol designers and implementors may find helpful.

4.1. PR-SCTP Service Definition for "timed reliability"

The "timed reliability" service is a natural extension of the "lifetime" concept already present in the base SCTP protocol.

When this service is requested for an SCTP association, it changes the meaning of the lifetime parameter specified in the SEND primitive (see Section 10.1, part (E) of RFC 2960 [2]; note that the parameter is spelled "life time" in that document.)

In the base SCTP protocol, the lifetime parameter is used to avoid sending stale data. When a lifetime value is indicated for a particular message and that lifetime expires, SCTP cancels the sending of this message, and notifies the ULP if the first transmission of the data does not take place (because of rwnd or cwnd limitations, or for any other reason). However, in the base protocol, if SCTP has sent the first transmission before the lifetime expires, then the message MUST be sent as a normal reliable message. During episodes of congestion this is particularly unfortunate, as retransmission wastes bandwidth that could have been used for other (non-lifetime expired) messages.

When the "timed reliability" service is invoked, this latter restriction is removed. Specifically, when the "timed reliability" service is in effect, the following rules govern all messages that are sent with a lifetime parameter:

TR1) If the lifetime parameter of a message is SCTP_LIFETIME_RELIABLE (or unspecified see Section 5), that message is treated as a normal reliable SCTP message, just as in the base SCTP protocol.

TR2) If the lifetime parameter is not SCTP_LIFETIME_RELIABLE (see Section 5), then the SCTP sender MUST treat the message just as if it were a normal reliable SCTP message, as long as the lifetime has not yet expired.

TR3) Before assigning a TSN to any message, the SCTP sender MUST evaluate the lifetime of that message. If it is expired, the SCTP sender MUST NOT assign a TSN to that message, but instead, SHOULD issue a notification to the upper layer and abandon the message.

TR4) Before transmitting or retransmitting a message for which a TSN is already assigned, the SCTP sender MUST evaluate the lifetime of the message. If the lifetime of the message is expired, the

SCTP sender MUST "abandon" the message, as per the rules specified in Section 3.5 marking that TSN as eligible for forward TSN. Note that this meets the requirement G1 defined in Section 4.3. IMPLEMENTATION NOTE: An implementation SHOULD delay TSN assignment as mentioned in RFC 2960 [2] Section 10.1. In such a case, the lifetime parameter should be checked BEFORE assigning a TSN, thus allowing a message to be abandoned without the need to send a FORWARD TSN.

TR5) The sending SCTP MAY evaluate the lifetime of messages at anytime. Expired messages that have not been assigned a TSN MAY be handled as per rule TR3. Expired messages that HAVE been assigned a TSN MAY be handled as per rule TR4.

TR6) The sending application MUST NOT change the lifetime parameter once the message is passed to the sending SCTP.

Implementation Note: Rules TR1 through TR4 are designed in such a way as to avoid requiring the implementer to maintain a separate timer for each message; instead, the lifetime need only be evaluated at points in the life of the message where actions are already being taken, such as TSN assignment, transmission, or expiration of a retransmission timeout. Rule TR5 is intended to give the SCTP implementor flexibility to evaluate lifetime at any other convenient opportunity, WITHOUT requiring that lifetime be evaluated immediately at the point in time where it expires.

4.2. PR-SCTP Association Establishment

An upper layer protocol (ULP) that uses PR-SCTP may need to know whether PR-SCTP can be supported on a given association. Therefore, the ULP needs to have some indication of whether the FORWARD-TSN chunk is supported by its peer.

Section 10.1 of RFC 2960 [2] describes abstract primitives for the ULP-to-SCTP interface, while noting that "individual implementations must define their own exact format, and may provide combinations or subsets of the basic functions in single calls."

In this section, we describe one additional return value that may be added to the ASSOCIATE primitive to allow an SCTP service user to indicate whether the FORWARD-TSN chunk is supported by its peer.

RFC 2960 indicates that the ASSOCIATE primitive "allows the upper layer to initiate an association to a specific peer endpoint". It is structured as follows:

```
Format: ASSOCIATE(local SCTP instance name, destination transport
      addr, outbound stream count)
-> association id [,destination transport addr list]
   [,outbound stream count]
```

This extension adds one new OPTIONAL return value, such that the new primitive reads as follows:

```
Format: ASSOCIATE(local SCTP instance name, destination transport
      addr, outbound stream count )
-> association id [,destination transport addr list]
   [,outbound stream count] [,forward tsn supported]
```

NOTE: As per RFC 2960, if the ASSOCIATE primitive is implemented as a non-blocking call, the new OPTIONAL return value shall be passed with the association parameters using the COMMUNICATION UP notification.

The new OPTIONAL parameter "forward tsn supported" is a boolean flag:

(0) false [default] indicates that FORWARD TSN is not enabled by both endpoints.

(1) true indicates that FORWARD TSN is enabled on both endpoints.

We also add a new primitive to allow the user application to enable/disable the PR-SCTP service on its endpoint before an association is established.

Format: ENABLE_PRSCTP(local SCTP instance name, boolean enable)

The boolean parameter enable, if set to true, will enable PR-SCTP upon future endpoint associations. If the boolean parameter is set to false, then the local endpoint will not advertise support of PR-SCTP and thus disable the feature on future associations. It is recommended that this option be disabled by default, i.e., in order to enable PR-SCTP, the user will need to call this API option with the enable flag set to "true".

4.3. Guidelines for defining other PR-SCTP Services

Other PR-SCTP services may be defined and implemented as dictated by the needs of upper layer protocols. If such upper layer protocols are to be standardized and require some particular PR-SCTP service other than the one defined in this document (i.e., "timed reliability"), then those additional PR-SCTP services should also be specified and standardized in a new RFC.

It is suggested that any such additional service definitions be modeled after the contents of Section 4.1. In particular, the service definition should provide:

1. A description of how the service user specifies any parameters that need to be associated with a particular message (and/or any other communication that takes place between the application and the SCTP transport sender) that provides the SCTP transport sender with the information needed to determine when to give up on transmission of a particular message.

   Preferably, this description should reference the primitives in the abstract API provided in Section 10 of RFC 2960 [2], indicating any:

   * changes to the interpretation of the existing parameters of existing primitives,

   * additional parameters to be added to existing primitives (these should be OPTIONAL, and default values should be indicated),

   * additional primitives that may be needed.

2. A description of the rules used by the sender side implementation to determine when to give up on messages that have not yet been assigned a TSN. This description should also indicate what protocol events trigger the evaluation, and what actions to take (e.g., notifications.)

3. A description of the rules used by the sender side implementation to determine when to give up on the transmission or retransmission of messages that have already been assigned a TSN, and may have been transmitted and possibly retransmitted zero or more times.

Items (2) and (3) in the list above should also indicate what protocol events trigger the evaluation, and what actions to take if the determination is made that the sender should give up on transmitting the message (e.g., notifications to the ULP.)

Note that in any PR-SCTP service, the following rule MUST be specified to avoid a protocol deadlock:

(G1) When the sender side implementation gives up on transmitting a message that has been assigned a TSN (i.e., when that message is "abandoned", as defined in Section 3.4), the sender side MUST mark that TSN as eligible for forward TSN, and the rules in Section 3.4 regarding the sending of FORWARD TSN chunks MUST be followed.

Finally, a PR-SCTP service definition should specify a "canonical service name" to uniquely identify the service, and distinguish it from other PR-SCTP services. This name can then be used in upper layer protocol standards to indicate which PR-SCTP service definition is required by that upper layer protocol. It can also be used in the documentation of APIs of PR-SCTP implementations to indicate how an upper layer indicates which definition of PR-SCTP service should apply. The canonical service name for the PR-SCTP service defined in Section 4.1 is "timed reliability".

4.4. Usage Notes

Detecting missing data in a PR-SCTP stream is useful for some applications (e.g., Fibre channel or SCSI over IP). With PR-SCTP, this becomes possible - the upper layer simply needs to examine the stream sequence number of the arrived user messages of that stream to detect any missing data. Note, this detection only works when all the messages on that stream are sent in order, i.e., the "U" bit is not set.

5. Variables

This section defines variables used throughout this document:

SCTP_LIFETIME_RELIABLE - A user interface indication defined by an implementation and used to indicate when a message is to be considered fully reliable.

6. Acknowledgments

The authors would like to thank Brian Bidulock, Scott Bradner, Jon Berger, Armando L. Caro Jr., John Loughney, Jon Peterson, Ivan Arias Rodriguez, Ian Rytina, Chip Sharp, and others for their comments.

7. Security Considerations

This document does not introduce any new security concerns to SCTP other than the ones already documented in RFC 2960 [2]. In particular, this document shares the same security issues as unordered data within RFC 2960 [2] identified by RFC 3436 [4]. An application using the PR-SCTP extension should not use transport layer security; further details can be found in RFC 3436 [4].

Note that the ability to cause a message to be skipped (i.e, the FORWARD TSN chunk) does not provide any new attack for a Man-In-the-Middle (MIM), since the MIM already is capable of changing and/or withholding data, thus effectively skipping messages. However, the FORWARD TSN chunk does provide a mechanism to make it easier for a

MIM to skip selective messages when the application has this feature enabled since the MIM would have less state to maintain.

8. IANA Considerations

IANA has assigned 192 as a new chunk type to SCTP.

IANA has assigned 49152 as a new parameter type code to SCTP.

9. References

9.1. Normative References

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

[2] 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, October 2000.

9.2. Informative References

[3] Clark, D. and D. Tennenhouse, "Architectural Considerations for a New Generation of Protocols", SIGCOMM 1990 pp. 200-208, September 1990.

[4] Jungmaier, A., Rescorla, E. and M. Tuexen, "Transport Layer Security over Stream Control Transmission Protocol", RFC 3436, December 2002.

10. Authors' Addresses

Randall R. Stewart
Cisco Systems, Inc.
8725 West Higgins Road
Suite 300
Chicago, IL 60631
USA

Phone: +1-815-477-2127
EMail: rrs@cisco.com

Michael A. Ramalho
Cisco Systems, Inc.
1802 Rue de la Porte
Wall Township, NJ 07719-3784
USA

Phone: +1.732.449.5762
EMail: mramalho@cisco.com

Qiaobing Xie
Motorola, Inc.
1501 W. Shure Drive, #2309
Arlington Heights, IL 60004
USA

Phone: +1-847-632-3028
EMail: qxie1@email.mot.com

Michael Tuexen
Univ. of Applied Sciences Muenster
Stegerwaldstr. 39
48565 Steinfurt
Germany

EMail: tuexen@fh-muenster.de

Phillip T. Conrad
University of Delaware
Department of Computer and Information Sciences
Newark, DE 19716
USA

Phone: +1 302 831 8622
EMail: conrad@acm.org
URI: http://www.cis.udel.edu/~pconrad

11. Full Copyright Statement

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgement

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