Network Working Group
Internet-Draft
Expires: November 14, 2003

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

May 16, 2003

SCTP Partial Reliability Extension
draft-stewart-tsvwg-prsctp-04.txt

Status of this Memo

Copyright Notice

Abstract

This memo describes an extension to the Stream Control Transmission Protocol (SCTP) RFC2960 [5] 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

(1) the protocol extensions, which consist of a new parameter for INIT and INIT ACK, and a new FORWARD TSN chunk type (2) one example 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) RFC2960 [5] 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 is 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 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 "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 [1].

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 versus 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 RFC2119 [3].

Comparisons and arithmetic on TSNs are governed by the rules in Section 1.6 of RFC2960 [5].

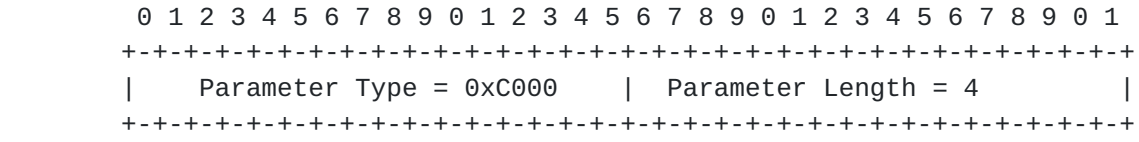
3. Protocol Changes to support PR-SCTP

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	0xC000

At the initialization of the association, the sender of the INIT or INIT ACK chunk shall include this OPTIONAL parameter to inform its peer that it is able to support the Forward TSN chunk. The format of this parameter is defined as follows:



Type: 16 bit u_int

0xC000, indicating Forward-TSN-Supported parameter

Length: 16 bit u_int

Indicate the size of the parameter i.e. 4.

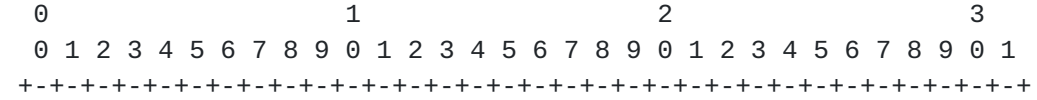
3.2 Forward Cumulative TSN Chunk Definition (FORWARD TSN)

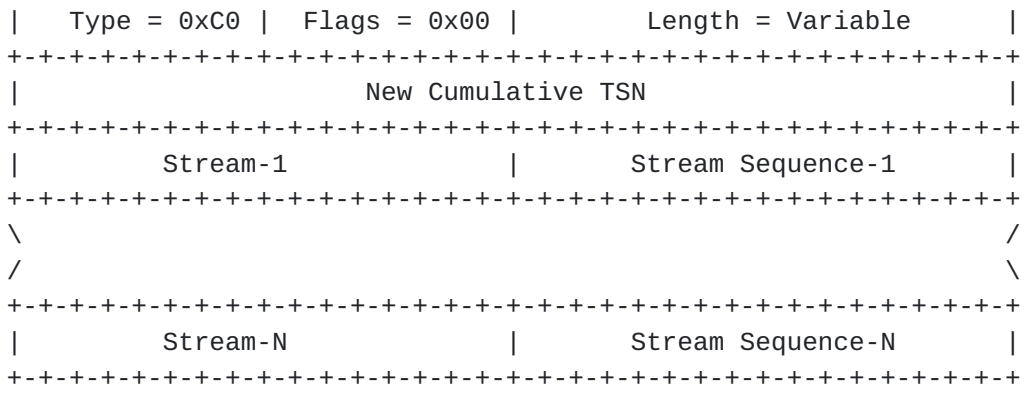
The following new chunk type is defined:

Chunk Type	Chunk Name
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:





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 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.

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 SHOULD include the Forward-TSN-supported parameter in the INIT chunk to indicate this fact to its peer.

3.3.2 Receipt of Forward-TSN-Supported param 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 SHOULD treat this parameter as an invalid or unrecognized parameter and respond to the data sender with an unrecognized parameter in the INIT-ACK, following the rules defined in Section 3.3.3 of RFC2960 [5].

When a receiver of an INIT-ACK detects a Forward-TSN-Supported parameter, and does not support the Forward-TSN chunk type, the receiver SHOULD treat this parameter as an invalid or unrecognized parameter and respond to the data sender with an unrecognized parameter error, following the rules defined in Section 3.3.3 of RFC2960 [4]. This error SHOULD be reported in an ERROR chunk bundled with the COOKIE-ECHO.

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

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

- MAY include the Forward-TSN-Supported parameter in the INIT-ACK,
- SHOULD record the fact that the peer does not support the FORWARD TSN chunk,
- MUST NOT send a FORWARD TSN chunk at any time during the associations life,
- 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 a 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:

- 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)
- special actions that a PR-SCTP implementation must take upon receipt of SACK
- rules governing 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 RFC2960 [5]) and 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 RFC2960 [5], and MUST NOT advance the cwnd based on this "abandoned" data chunk.

A3) 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 RFC2960 [5].

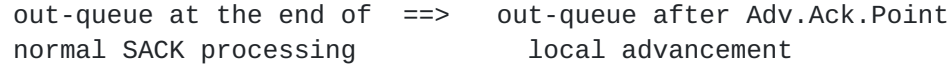
The data sender MUST then perform the following additional steps :

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:



```
                ...                             ...
      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".

C4) For each "abandoned" TSN the sender of the FORWARD TSN SHOULD list each stream and sequence number in the forwarded 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.

A4) 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 C1 - 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 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.

F4) The detection criterion for out-of-order SACKs MUST remain the same as stated in RFC2960, 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 a decision to "abandon" a chunk is made based on a retransmission decision (e.g T3-Timeout or Fast Retransmit) the appropriate congestion adjustment as specified in RFC2960 MUST be made before any FORWARD TSN chunk is sent.

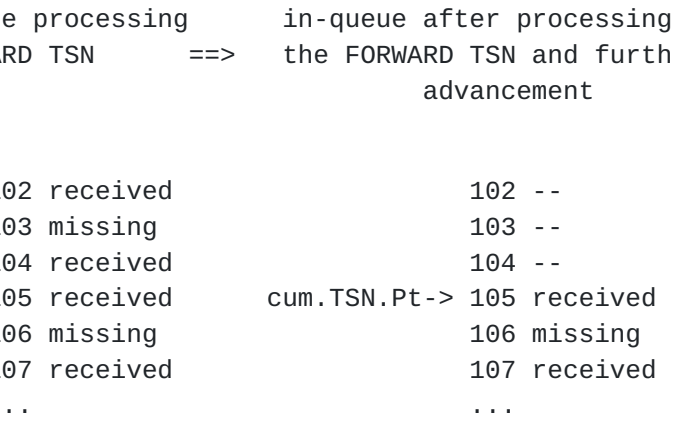
3.6 Receiver Side Implementation of PR-SCTP

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 RFC2960 [5]---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 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 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 RFC2960 [5] 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.

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.

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 an 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 standard.

Section 4 is organized as follows:

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

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.

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

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 RFC2960 [5]; 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, 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) before the lifetime expires. 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) 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, 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.

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 as way 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 RFC2960 [5] 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.

RFC2960 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 RFC2960, 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 supported by the peer.

(1) true indicates that FORWARD TSN is supported by the peer.

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.

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 RFC2960 [5], 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. Fiber 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. Acknowledgments

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

6. Security Considerations

This document does not introduce any new security concerns to SCTP other than the ones already documented in RFC2960 [5]. In particular this document shares the same security issues as unordered data within RFC2960 [5]. An application using the PR-SCTP extension should not use transport layer security. Further details can be found in RFC3436 [4].

7. IANA Considerations

One new chunk type is added to SCTP ('0xC0') by this document.

One new parameter type code is defined by this document to be added to SCTP ('0xC000').

References

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

[2] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

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

[4] Jungmaier, A., Rescorla, E. and M. Tuexen, "TLS over SCTP", RFC 3436, December 2002.

[5] 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.

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
Temple University
CIS Department
Room 303, Computer Building (038-24)
1805 N. Broad St.
Philadelphia, PA 19122
US

Phone: +1 215 204 7910
EMail: conrad@acm.org
URI: http://www.cis.temple.edu/~conrad

Intellectual Property Statement

Full Copyright Statement

Acknowledgement

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