Network Working Group
INTERNET-DRAFT
Expires in six months
April 2001

Q. Xie
Motorola
R. R. Stewart
C. Sharp
Cisco
I. Rytina
Ericsson

SCTP Unreliable Data Mode Extension
<draft-ietf-tsvwg-usctp-00.txt>

Abstract

This memo describes an extension to the Stream Control Transmission Protocol (SCTP) [RFC2960] to provide unreliable data transfer services. The benefits of this extension includes unified congestion control over reliable and unreliable data traffics, single association for multi-type content data services, link level fault tolerance for unreliable data applications, unreliable data stream multiplexing, etc.

TABLE OF CONTENTS

1. Introduction

This memo adds unreliable data transfer services to SCTP. The design presented in this memo allows the co-existence of unreliable data streams and reliable streams in a single SCTP association.

The following are some of the advantages for integrating unreliable data service into SCTP:

1) Some applications services may benefit from U-SCTP by being able to use a single SCTP association to carry both reliable contents, such as text pages, billing and accounting information, setup signaling, and unreliable contents, such as certain type of media data that does not need a reliable transport.

2) Unreliable data traffic carried within U-SCTP streams will enjoy the same communication failure detection and protection capabilities as the normal reliable SCTP data traffic does, including the ability of quickly detecting a failed destination address and failing-over to an alternate destination address and the ability of being notified if the data receiver becomes unreachable. This enables one to build high system robustness into unreliable data transfer applications.

3) With U-SCTP streams an application can control its lost data retransmission policies so as to only perform a certain times of retransmission to a lost datagram.

4) In addition to providing unordered unreliable data transfer as UDP does, U-SCTP can provides _ordered_ unreliable data transfer service.

5) U-SCTP employs the same congestion control and congestion avoidance over unreliable data traffic as it does to the normal reliable traffic - this is very desirable since it is much friendlier towards the network than UDP is.

6) Taking advantage of SCTP data chunk bundling function, sending multiple unreliable data streams across a single SCTP association creates a very efficient and effective way of data multiplexing.

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 RFC 2119 [RFC2119].

3. Unreliable Data Design

With the unreliable data extension, an SCTP data sender will be allowed to designate a sub-set of its outbound streams to be unreliable streams. The user data chunks sent to an unreliable stream will share the same TSN space, the same congestion control/avoidance treatment, and the same transmission priority as those sent to a reliable stream, but they will not be retransmitted (or only be retransmitted for a limited times) if they are found missing at the data receiver.

3.1 Unreliable Streams 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
Unreliable Streams	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 unreliable streams and to designate its unreliable outbound streams.

The format of the Unreliable Streams 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 = 0xC000    |  Parameter Length = variable  |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |      u-stream start #1 = US1  |      u-stream end #1 = UE1    |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  /                                                               /
  \                            . . . .                            \
  /                                                               /
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |      u-stream start #k = USk  |       u-stream end #k = UEk   |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

  Type: 16 bit u_int

     0xC000, indicating Unreliable Streams parameter

  Length: 16 bit u_int

     Indicate the size of the parameter in octets, including the

```
      Type, Length, u-stream start, and u-stream end fields.

   u-stream start: 16 bit u_int, and
   u-stream end: 16 bit u_int
```

Each pair of u-stream start and u-stream end fields defines one or more unreliable outbound streams, starting from stream number US and ending with stream number UE. The union of all the pairs together defines the complete sub-set of all unreliable outbound streams.

The following are some examples of unreliable stream designation (assuming OS = 10):

Example 1: (assuming OS = 10)

```
  +------------+-----------+
  |type=0xC000 | length=8  |         Streams    Mode
  +------------+-----------+    ==>  ---------------------
  | u-start= 3 | u-end= 5  |         0 - 2      reliable
  +------------+-----------+         3 - 5      unreliable
                                     6 - 9      reliable
```

Example 2: (assuming OS = 10)

```
  +------------+-----------+
  |type=0xC000 | length=12 |         Streams    Mode
  +------------+-----------+    ==>  ---------------------
  | u-start= 3 |  u-end= 5 |         0 - 2      reliable
  +------------+-----------+         3 - 9     unreliable
  | u-start= 6 |  u-end= 9 |
  +------------+-----------+
```

Example 3: (assuming OS = 10)

```
  +------------+-----------+
  |type=0xC000 | length=12 |         Streams    Mode
  +------------+-----------+    ==>  ---------------------
  | u-start= 9 |  u-end= 9 |         0          unreliable
  +------------+-----------+         1 - 8      reliable
  | u-start= 0 |  u-end= 0 |         9          unreliable
  +------------+-----------+
```

Example 4: (assuming OS = 10)

```
  +------------+-----------+
  |type=0xC000 | length=8  |         Streams    Mode
  +------------+-----------+    ==>  ---------------------
  | u-start= 0 |  u-end= 9 |         0 - 9      unreliable
  +------------+-----------+
```

Example 5: (assuming OS = 10)

```
  +------------+-----------+
  |type=0xC000 | length=4  |         Streams    Mode
```

```
       +------------+-----------+    ==>  ---------------------
                                          0 - 9      reliable
```

If no streams are marked as unreliable but the sender does support the unreliable streams option, the sender still SHOULD include a parameter with no u-stream ranges and a fixed Parameter Length of 4.

3.2 Forward Cumulative TSN Chunk Definition (FORWARD TSN)

The following chunk type is defined in order to support the SCTP unreliable stream operation:

Chunk Type	Chunk Name
11000000	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 unreliable data chunks and will no longer be 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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |1 1 0 0 0 0 0 0|  Chunk Flags  |0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      New Cumulative TSN                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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 shall consider any missing TSNs earlier than or equal to this value as received and stop reporting them as gaps in any subsequent SACKs.

4. Unreliable Data Operation

In this section, we first define the procedures for opening unreliable streams in an SCTP association. Then, we will discuss procedures for sending and receiving unreliable SCTP data chunks.

4.1 Initialization of Unreliable Streams

If the SCTP data sender plans to send unreliable data, at the
initialization of the association it MUST include the Unreliable
Streams parameter in its INIT or INIT ACK chunk to indicate to its
peer which of its outbound streams are going to be used as unreliable

streams.

Upon the reception of the Unreliable Streams parameter, the data receiver SHALL determine and record the mode (reliable or unreliable) of each inbound stream, as it allocates resource for its inbound streams.

Note, if the data receiver does not support unreliable inbound streams, it SHOULD treat the Unreliable Streams parameter as an invalid or unrecognized parameter and respond to the data sender with an operational error, following the rules defined in Section 5.1 of [RFC2960].

Upon reception of the operational error indicating that its peer does not support unreliable streams, the data sender may choose to either:

1) end the initiation process, in consideration of the peer's inability of meeting the requested features for the new association, or
2) continue the initiation process, but with the understanding that ALL its outbound streams will be reliable.

In either case, the data sender SHOULD inform its upper layer its peer's inability of supporting unreliable data transfer.

Initiation of streams as reliable and/or unreliable may be under the control of the SCTP user. Hence, the ULP primitive "ASSOCIATE" (see Section 10.1 of [RFC2960]) should be expanded to contain the optional U-stream-start and U-stream-end values.

4.2 Send Unreliable Data

During the lifetime of the association, any user data sent to an unreliable stream will be treated as unreliable user data and will automatically be transmitted in unreliable mode.

The data sender shall fragment an unreliable user message if its size is larger than the current PMTU. The sender shall follow the fragmentation rules and procedures as defined in [RFC2960].

The SCTP data sender shall handle user data sent to an unreliable stream the same way as it handles user data sent to a reliable stream (i.e., the same timer rules, congestion control rules, failure detection rules, RTO control rules, etc.), with the following exceptions:

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 classic SCTP Cumulative TSN Ack Point as defined
    in [RFC2960]).

A2) Before retransmitting a DATA chunk (due to either a T3-rtx timer expiration as defined in 6.3.3 of [RFC2960] or a 4th missing indication as defined in 7.2.4 of [RFC2960]), the SCTP data sender MUST check whether the DATA chunk is being transmitted on an unreliable stream. If so, it will perform the following:

B1) Check the value of the unreliable retransmission counter "Unrel.Trans.Count" value for the DATA chunk. This value may be set by the SCTP user to 0 (no retransmission) for complete unreliability, or N (where N >0) for limited reliability at the time when the user message is passed to SCTP.

B2) If the "Unrel.Trans.Count" of the chunk is currently greater than 0, the sender MUST retransmit the data chunk and then decrease the "Unrel.Trans.Count" by 1. The same rules for retransmission as defined in [RFC2960] SHALL be used for RTO calculation, destination selection, error reporting, etc.

B3) If the "Unrel.Trans.Count" is currently 0, the sender MUST NOT retransmit the data chunk. Instead, the sender MUST mark the data chunk as being finally acked.

A3) whenever the data sender receives a SACK from the data receiver, it SHALL first process the SACK using the normal procedures as defined in Section 6.2.1 of [RFC2960].

The data sender MUST then perform the following additional steps:

C1) Update the "Advanced.Peer.Ack.Point" to the Cumulative TSN ACK carried in the SACK __if__ the former is behind.

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 is marked as acknowledged. For 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 acked                      103 acked
             104 acked          Adv.Ack.P-> 104 acked
             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"
        becomes more advanced 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, an endpoint MUST NOT use the FORWARD TSN for any purposes other than the above circumstance.

Note, if a TSN is indicated as missing by a SACK carrying gap reports AND the TSN is earlier than the current "Advanced.Peer.Ack.Point", the data sender MUST NOT take any action on this TSN, i.e., it MUST ignore this missing report to this TSN. When this happens, it is normally an indication that a previous FORWARD TSN from the data sender may have been lost in the network.

Note, the detection criterion for out-of-order SACKs MUST remains 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").

The ULP primitive "DATA" (defined in Section 10.1 of [RFC2960]) should be expanded to contain an optional unreliable retransmission parameter to assign a "Unrel.Trans.Count" value to each user message to be sent to an unreliable stream.

4.3 Receive Unreliable Data

Regardless whether a DATA chunk arrives from a reliable stream or an unreliable stream, the receiver MUST perform the same TSN handling (e.g, duplicate detection, gap detection, SACK generation, cumulative TSN advancement, etc.) as defined in [RFC2960].

However, whenever a FORWARD TSN chunk arrives the data receiver MUST update its cumulative TSN to the value carried in the FORWARD TSN chunk, and MUST stop reporting any missing TSNs earlier than or equal to the new cumulative TSN.

Whenever an unreliable 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. However since it is possible that the DATA chunk(s) being waited upon is one that will not be retransmitted by the sender, when a FORWARD TSN chunk arrives, the receiver MUST examine all of its unreliable stream reordering queues, and immediately make available for delivery any messages that carry a TSN (or a starting TSN in the case of reassembled messages) earlier than the new cumulative TSN updated by the FORWARD TSN.

When receiving a FORWARD TSN, cautions MUST also be taken in updating

the re-assembly queue of the receiver, including the removal of any
partially reassembled message which is still missing one or more TSNs
earlier than or equal to the new cumulative TSN updated by the FORWARD
TSN.

4.4. Other Issues on Unreliable Data

4.4.1 Unreliable Data Stream Multiplexing

Sometimes, it is desirable to aggregate different media streams and send them over a single communication connection, and normally unreliable transport is preferred for these types of media streams.

With U-SCTP this is easily achieved by assigning each different media stream to a different unreliable SCTP stream and letting the SCTP's built-in data bundling mechanism to perform the multiplexing at the sender and demultiplexing at the receiver.

4.4.2 Fault Tolerant Unreliable Data Transfer

When the data receiver is multi-homed, unreliable data transfer using U-SCTP will obtain the same fault tolerance benefit as that of the reliable data services across an SCTP association.

This is because the data sender still follows the same failure detection rules and still counts the omitted retransmission against the association and the destination transport address to which the unreliable DATA chunk was originally sent. Thus, when failure occurs, the data sender will detect the failure and shift the unreliable data services to an alternate destination address, following the same procedures as defined in Section 8 of [RFC2960] for reliable data transfer.

4.4.3 Detection of Missing Unreliable Data

Detecting missing data in an unreliable stream is useful for some applications (e.g. Fiber channel or SCSI over IP). With U-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 in that stream are sent in order, i.e. their "U" bit MUST NOT be set.

5. Acknowledgments

The authors would like to thank Scott Bradner, Jon Berger, John Loughney, Ivan Arias Rodriguez, and others for their comments.

6. Authors' Addresses

Qiaobing Xie   Tel: +1-847-632-3028
Motorola, Inc.   EMail: qxie1@email.mot.com
1501 W. Shure Drive, #2309

Arlington Heights, IL 60004
USA

Randall R. Stewart   Tel: +1-815-477-2127
Cisco Systems, Inc.   EMail: rrs@cisco.com
8725 West Higgins Road
Suite 300
Chicago, Ill 60631

Chip Sharp   Tel: +1-919-392-3121
Cisco Systems Inc.   EMail:chsharp@cisco.com
7025 Kit Creek Road
Research Triangle Park, NC 27709
USA

Ian Rytina   Tel: +61-3-9301-6164
Ericsson Australia   EMail:ian.rytina@ericsson.com
37/360 Elizabeth Street
Melbourne, Victoria 3000
Australia

7. References

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", RFC 2026, October 1996.

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

[RFC2960] R. R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. J. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and, V. Paxson, "Stream Control Transmission Protocol," RFC2960, October 2000.

This Internet Draft expires in 6 months from April 2001.