Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 12, 2013

R. Stewart
Adara Networks
P. Lei
Cisco Systems, Inc.
M. Tuexen
Univ. of Applied Sciences
Muenster

October 9, 2012

Stream Control Transmission Protocol (SCTP) Packet Drop Reporting
draft-stewart-sctp-pktdrprep-14.txt

Abstract

This document describes a new chunk type for SCTP. This new chunk type can be used by both endhosts and routers to report the loss of SCTP datagrams due to errors in transmission or other drops not due to congestion.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

The modern Internet has a wide variety of link types. A vast majority of these link type present a very low bit error rate. In recent years, however, a large number of higher bit error links are becoming more wide spread for example satellite, 802.11, and 3G cellular to name just a few. Often times one of the segments in the path will realize that it is going to drop a packet due to bit errors. When a drop does occur due to an error other than congestion, the drop will be mistakenly interpreted as congestion in the network by any transport protocol.

This "mis-interpretation" of feed back may cause an SCTP sender to drastically under utilize a link. Depending on how severe the error rate, the sender may stay in a continual state of congestion collapse, thus effecting performance in a very negative way over the entire life of the association.

This draft proposes a new SCTP chunk type that can be used by a sender to discover dropped packets in such a case. This chunk may also be used by an SCTP receiver to report cases of window overrun or received data that may have had bit errors.

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 [RFC2119].

3. Architectural Considerations

The Packet Drop Reports (PKTDROP) can be generated by an SCTP endpoint or a middle box.

The SCTP endpoint can inform its peer that it has received an SCTP packet, but the CRC32c was wrong. The peer can retransmit this packet and does not need to adopt the window for congestion control because this packet-loss is not related to congestion. It is also possible for the endpoint to make clear that the receiver window was overrun.

There are two scenarios where a middle box may send Packet Drop Reports.

For the first scenario consider a middle box in the path between the

communicating SCTP endpoints (see Figure 1), which communicates with a middle box peer. Please note that the middle box peer can be can be located at the same physical device that also runs the SCTP stack or running on separate boxes providing a tunneling service. The crucial point here is, that there is some protocol running between the middle box and the middle box peer.

```
                   +------------+     +-----------------+
           +-------| middle box +-----+ middle box peer +--------+
           |       +------------+     +-----------------+        |
           |                                                     |
      +----+-----+                                          +-----+----+
      |   SCTP   |                                          |   SCTP   |
      | endpoint |                                          | endpoint |
      +----------+                                          +----------+
```

Figure 1

If they run a protocol below SCTP which provides an acknowledgment service in a way that the sending middle box knows that a packet was not received by the middle box peer and the packet was not dropped due to congestion, then the sending middle box can also send a Packet Drop Report back to the sending SCTP endpoint. It can also indicate the current status of the send queue and the bandwidth limit between the middle boxes if applicable.

In the other scenario there is only one middle box involved, which means that there is no middle box specific communication, as shown in Figure 2. In this case the middle box may want to send Packet Drop Reports to report to the SCTP sender the number of queued data and a possible bandwidth limitation between the middle box and the SCTP receiver.

```
                                +------------+
           +--------------------+ middle box +--------------------+
           |                    +------------+                    |
           |                                                      |
      +----+-----+                                           +-----+----+
      |   SCTP   |                                           |   SCTP   |
      | endpoint |                                           | endpoint |
      +----------+                                           +----------+
```

Figure 2

4. New Chunk Types

This section defines the new chunk type that will be used to report dropped packets not due to congestion in the network. Figure 3 illustrates the new chunk types.

Chunk Type	Chunk Name	
0x81	Packet Drop Chunk	(PKTDROP)

Figure 3

It should be noted that the PKTDROP Chunk format requires the receiver to ignore the chunk if it is not understood. This is accomplished as described in RFC2960 [RFC2960] section 3.2. by the use of the upper bit of the chunk type.

4.1. Packet Drop Chunk (PKTDROP)

This chunk is used to communicate to the remote endpoint the purposeful dropping of a packet which is NOT due to congestion or the current state of a network bottleneck.

```
     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 = 0x81   |   Flags=CTBM  |      Chunk Length             |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                 Link Bandwidth or Maximum Rwnd                |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                     Size of data on queue                     |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |   Truncated Length            |          Reserved             |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                       Dropped SCTP Packet                     |
    \                  (No IP header Included - optional)           /
    /                                                               \
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Chunk Type : 8 bits - This value MUST be set to 0x81 for all packet drop chunks.

Flags : 8 bits - The lower 3 bits of this field are used to identify various properties about the packet report:

Bit	Set Value	Meaning
C	00001000	This bit transforms the link bandwidth and queue size fields from a byte count to a packet count. This will be set only by middle boxes that cannot determine byte counts.
T	00000100	This bit informs the receiver the packet was truncated to fit. If this bit is set then the truncated length holds the original packet length (from the IP header).
B	00000010	This bit informs the receiver that a BAD CRC32c was detected by an SCTP endpoint.
M	00000001	This bit informs the receiver that the source of the packet is a middle box, not the endhost. This also tells the receiver to look for the Peers Verification tag in the packet. This is equivalent to the T bit in an ABORT or SHUTDOWN COMPLETE packet.

Chunk Length : 16 bits unsigned int - This value holds the length of the chunk including the chunk header.

Link Bandwidth or Maximum Rwnd : 32 bits unsigned int - If the M bit is set to '1', this value holds the bandwidth capacity in bytes per second of the link the middle box is connected to aka the bottleneck bandwidth being sent towards. If the M bit is set to '0' then this value holds the maximum allowable Rwnd of the peer. This value is normally the same value as that found in the INIT or INIT-ACK's a_rwnd field.

Size of data on queue : 32 bits unsigned int - This value represents the current number of bytes of data onqueue towards the link or reader. In the case of a middle box (M bit set to '1'), this will inform the receiver how much data is currently in queue towards the bottleneck, if the link layer is reliable (e.g. a Reliable Link Protocol) this number will also include any inflight data over the link. In the case of an endhost (M bit set to '0') this will tell

the receiver how much data is still un-read or held for reassembly by the remote SCTP endpoint.

Truncated Length : 16 bits unsigned int - This value is set to the original size of the SCTP packet that was dropped. The size does NOT include the IP header or any other IP option field (i.e. it is the size of the SCTP payload within the IP packet). This value is only valid if the T bit is set to '1'.

Reserved : 16 bits unsigned int - This value SHOULD be set to '0' by the sender and MUST be ignored by the receiver.

Data Field: variable - This field is variable and usually holds the packet that was dropped or a portion of it if the T bit is set. In some instances a middle box may send a packet drop report without this data. In such a case, it is reporting to the SCTP sender the current bandwidth and NOT reporting a dropped packet.

5. Procedures

5.1. Sender of the packet drop

A packet drop chunk MUST NOT be send in response to a packet containing an ABORT chunk or a packet drop chunk.

5.1.1. Middle box

Periodically a middle box may realize that it cannot transmit a chunk due to errors in transmission. In such a case the middle box SHOULD compose a packet drop chunk to send back to the SCTP sender of the dropped packet. The middle box MUST set the M bit to one and copy into the SCTP common header the verification tag found in the packet to be dropped. The IP addresses and SCTP ports MUST be swapped so that the receiver of the packet drop will identify the packet drop report with the correct SCTP association.

After filling out the IP and SCTP headers, the sender MUST copy in all or part of the SCTP packet being dropped not including the IP and SCTP header (i.e. starting at the first SCTP chunk). The sender of the packet drop report MUST assure that the packet fits into a single MTU, truncating the packet and setting the T bit if necessary. If the middle box truncates the packet to fit in a single MTU, the middle box MUST copy the original length of the SCTP packet into the Truncated length field.

The middle box sending the drop packet report SHOULD also total up the data that is inflight (towards the destination, of the dropped

packet) and the data that is inqueue awaiting transmission, placing this size in 'size of data on queue' field. The sender SHOULD also place the link bandwidth, in bytes per second, in the 'bandwidth' field. The receiving SCTP endpoint should use this information to adjust its congestion control parameters.

5.1.2. End host

An SCTP endhost MAY want to send a packet drop for one of two reasons, the SCTP sender has overrun the local receivers rwnd or the inbound packet failed its CRC-32c check.

If the SCTP endhost detects a bad CRC-32c it will still use the SCTP common header to attempt to locate the association. If a valid association is found and the verification tag are correct, chances are good that the common header was not damaged and thus the found TCB can be used to generate a drop report with the rest of the SCTP packet.

In either case the receiver that is sending the drop report MUST copy the packet, with possible truncation as described above. The sender of the drop report MUST set the M bit to 0 and place the verification tag of the peer in the outbound packet. The sender of the drop report should also place the maximum rwnd value in the 'Maximum Rwnd' field, and should place the number of bytes unread in the 'data on queue' field. Note that the unread byte count MUST include data in any local buffer not yet read by the user, data pending reassembly and data awaiting stream re-ordering.

5.2. Receiver side

When receiving a Packet Drop report the SCTP endpoint will want to examine the drop report and based on the information possibly retransmit lost information to the peer. The receiver SHOULD verify that the sender actually had a packet by comparing some of the data that was dropped to the data that was sent. This is done to assure the sender that a malicious receiver is not attempting to induce a retransmission of a congestion related dropped packet. The following list illustrates the handling procedure by chunk type for dropped packets.

1. DATA - For a data chunk drop, the receiver SHOULD locate the identified DATA chunk and mark it for retransmission. The DATA chunk should be treated just as if it had been marked for fast retransmit with the exception that no adjustment should be made to the value of cwnd (providing that the receiver can validate a portion of the packet as being what was sent).

2. SACK - For a lost SACK chunk, a receiver MAY wish to send out a new SACK illustrating the current receiver conditions.

3. INIT - For a INIT chunk, the receiver SHOULD resend the INIT restarting its local T-1 timer.

4. HEARTBEAT REQUEST - For a heartbeat request, the receiver SHOULD resend a heartbeat to the source address of the packet.

5. SHUTDOWN - For a shutdown request, the SCTP receiver SHOULD resend the shutdown request.

6. SHUTDOWN ACKNOWLEDGEMENT - For a shutdown acknowledgement the receiver SHOULD resend the SHUTDOWN-ACK.

7. COOKIE ECHO - For a Cookie Echo the receiver SHOULD retransmit the lost COOKIE ECHO, restarting any cookie timer.

8. COOKIE ACKNOWLEDGMENT - For a lost cookie-ack a receiver should retransmit a cookie-ack to the peer.

9. ASCONF - For a lost ASCONF, the receiver SHOULD retransmit the ASCONF restarting any timer associated with the ASCONF.

10. FORWARD TSN - For a lost forward TSN the endpoint SHOULD resend a new forward TSN reporting the current value that the TSN should be advanced to. Note this may not be the same information as that contained in the dropped chunk.

After queuing for retransmission any lost chunks, the sender MUST also examine the bandwidth and queue fields taking into consideration the source. If the M bit is set to '0' then the source of the drop report was the SCTP peer. In such a case the receiver MUST immediately adjust its peer rwnd by taking the value in the 'Maximum Rwnd' field, subtracting the value of the 'data on queue' field and any data in-flight.

If the sender is a middle box, M bit set to '1', the receiver MAY adjust the cwnd for the source address of the drop packet by applying the following algorithm if the current RTT of the link is larger than the variable RTO.Large. A receiver of a packet drop report MUST NOT adjust its cwnd if the RTT is or has ever been measured to be less than or equal to RTO.Large.

Establish the True RTT using the values normally
used in calculating the RTO, set this value in
milliseconds into the variable 'rtt'.

```
   rtt = (lastsa >> 2) + lastsv) >> 1;

   Validate that an adjustment can be made.

   if ((pd.chunk_flags AND M_BIT) != M_BIT)
       return

   if ( rtt < RTO.Large)
       return



   Set 'bottle_bw' to the value found in the Link Bandwidth
   field.

   bottle_bw = ntohl(pd.bottle_bw);

   Set 'on_queue' to the value found in the size on data queue
   field.

   on_queue =  ntohl(pd.current_onq);

   Adjust the on_queue for any in-flight data that may yet
   not have arrived at the bottle neck.
   if(on_queue < flight_size) {
       on_queue = flight_size;
   }

   Calculate the bandwidth available by multiplying the bottle_bw
   variable times the rtt and dividing the result by a thousand.
   Call this value 'bw_avail'.

   bw_avail = (bottle_bw*rtt)/1000;


   If more is 'on_queue' than the current value of 'bw_avail' a
   negative congestion window adjustment is needed.

   if (on_queue > bw_avail) {
       Clear the partial bytes acked field.
       partial_bytes_acked = 0;

       Subtract the bw_avail from the current on_queue call this
       value the 'decrease'.

       overrun = on_queue - bw_avail;

       Undo any congestion adjustment if a SACK has been processed.
```

```
       if (seen_a_sack_this_packet)
            cwnd = prev_cwnd


       Calculate the portion of the onqueue data that is
       caused by this endpoints in-flight data.

       seg_inflight = flight_size / mtu
       seg_onqueue = on_queue / mtu
       my_portion = (overrun * seg_inflight)/seg_onqueue;

       If we have already adjusted the cwnd, indicated by the
       fact that the cwnd is larger than the flight size, we
       adjust our portion down by a smaller amount i.e the
       amount we have already adjusted it previously.

       if( cwnd > flight_size )
            adjust =  cwnd - flight_size;
            if( adjust > my_portion)
                    my_portion = 0;
            else
                    my_portion -= adjust
       }


       Adjust the cwnd downward by our calculated amount.

       cwnd -= my_portion

       If the current flight size is larger than this new congestion
       window, set the congestion window to the current flight size.
       if (flight_size > cwnd) {
          cwnd = flight_size
       }

       If the current congestion window is smaller than a
       single MTU set the current congestion window to 1 MTU.
       if (cwnd <= mtu) {
             cwnd = mtu;
       }

       Set ssthresh to the current congestion window minus 1 byte.
       ssthresh = cwnd - 1;

   } else {
       Otherwise an increase is needed. Calculate the increase value
       'incr' by taking the minimum of one fourth the bw_avail minus
       the size on queue OR the MTU size times max burst (whichever
```

```
       is smaller).

       incr = min(((bw_avail - on_queue) >> 2),
                  ((int)asoc.max_burst * (int)mtu));

       Add this value to the current congestion window
       cwnd += incr;

       After making an increase to the congestion window verify that
       the value of cwnd is smaller than or equal to the bw_avail if
       not, set the cwnd to the value of bw_avail.

       if (cwnd > bw_avail) {
           cwnd = bw_avail;
       }

   }
```

6. Security Considerations

TBD

7. Recommended Variables

The following are the recommended values for variables defined within this document:

RTO.Large - 500 Milliseconds.

8. Normative References

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

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

[RFC2960] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L., and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, October 2000.

Authors' Addresses

Randall R. Stewart
Adara Networks
2150 First Street
San Jose, CA 29036
USA

Email: randall@lakerest.net

Peter Lei
Cisco Systems, Inc.
8735 West Higgins Road
Suite 300
Chicago, IL 60631
USA

Phone:
Email: peterlei@cisco.com

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

Email: tuexen@fh-muenster.de