

A Two-level Threshold Recovery Mechanism for SCTP*

Armando L. Caro Jr., Janardhan R. Iyengar, Paul D. Amer, Gerard J. Heinz

Computer and Information Sciences
University of Delaware
{acaro, iyengar, amer, heinz}@cis.udel.edu

Randall R. Stewart

Cisco Systems Inc.
rrs@cisco.com

Abstract

One important feature of the Stream Control Transmission Protocol (SCTP) is its network failure detection and recovery mechanism provided by direct support of multihoming. This mechanism is conservative (and rightfully so), but unfortunately does not fully exploit SCTP's multihoming capability to maintain seamless communication between the endpoints. We decouple the failure detection from the recovery process, and propose a recovery mechanism that increases throughput. Our proposed solution does not wait until failure is detected before initiating a recovery process. Instead, the recovery process begins during early signs of a possible failure, while the same conservative failure detection is maintained. We argue that our proposed recovery scheme improves overall throughput.

Keywords: SCTP, multihoming, failover, persistent connections, mission critical

1 Introduction

Mission critical systems rely on redundancy at multiple levels to provide uninterrupted service during resource failures. Such systems when connected to IP networks often deliver network redundancy by *multihoming* their hosts. A host is multihomed if it can be addressed by multiple IP addresses [1]. Redundancy at the network layer allows a host

to be accessible even if one of its IP addresses becomes unreachable; packets can be rerouted to one of its alternate IP addresses. TCP does not support multihoming. Any time either endpoint's IP address becomes inaccessible, say due to an interface failure, TCP's connection will timeout and abort, thus forcing the upper layer to recover. The recovery delay can be unacceptable for mission critical applications such as IP telephony, IP storage, and military battlefield communications.

To address TCP's shortcoming, the Stream Control Transmission Protocol (SCTP) has been designed with fault tolerance in mind. SCTP supports multihoming at the transport layer to allow sessions, or *associations* in SCTP terminology, to remain alive even when an endpoint's IP address becomes unreachable. SCTP has a built-in failure detection and recovery system, known as *failover*, which allows associations to dynamically send traffic to an alternate peer IP address when needed.

In the current specification, RFC2960, SCTP's failover mechanism acts conservatively to ensure that associations do not prematurely and incorrectly assume failures when in fact either no failure has occurred, or the failure is temporary and repaired quickly. A drawback to being conservative is that performance unnecessarily suffers during the detection period when a failure has indeed occurred. We argue that SCTP is too conservative, and that SCTP's multihoming capability can be better exploited to reduce communication degradation during failure detection without additional overhead.

We propose a new recovery scheme to increase the throughput of an association. Our proposal decouples failure detection from the recovery process. The failure detection remains conservative, while the recovery process becomes more aggressive. Our solution begins the recovery process during early signs of a possible failure. Beginning the recovery process before the failure is detected has the advantage of providing improved throughput.

*Prepared through collaborative participation in the Communications and Networks Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011. The U. S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

2 Example architecture

SCTP’s RFC2960 states that when its peer is multihomed, “an endpoint SHOULD always transmit to the primary path.” However, the endpoint “SHOULD try to retransmit a chunk to an active destination transport address that is different from the last destination address to which the DATA chunk was sent,” that is, to a secondary or alternate destination. Sometimes an association may continue having difficulty transmitting to the primary destination, and as a result, may consecutively timeout multiple times. If a significant number of consecutive timeouts occur, SCTP will failover to an alternate destination and mark the primary destination as inactive [4].

To better understand SCTP’s proposed handling of failover conditions, we present examples which illustrate three cases of a timeout. *No failure* is when a timeout occurs simply due to congestion, and the primary destination is still reachable. *Short term failure* characterizes the condition where multiple consecutive timeouts occur, but not enough, according to RFC2960, to mark the primary destination as unreachable. *Long term failure* includes all cases where the primary destination fails to respond during multiple timeout periods, and is marked as inactive.

Figure F1 illustrates an example architecture which we assume in our example scenarios. An SCTP association exists between two hosts *A* and *B*, which are multihomed peers connected by an IP network. *A* is addressable by interfaces A_1 and A_2 , while *B* is addressable by B_1 and B_2 . We assume due to one of several possible reasons, such as load balancing, policy based routing, or path diversity, that data traffic from *A* to B_1 is routed through A_1 , and from *A* to B_2 is routed through A_2 .

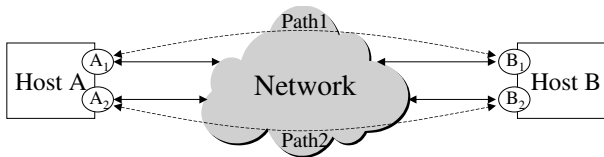


Figure F1: Example multihome architecture

3 Specified behavior

Certain conventions and assumptions in Figures F2-F5 and F7-F9 are used to illustrate our examples. The two pair of vertical lines represent source-destination pairs. The two inner vertical lines correspond to host *A*’s interfaces A_1 and A_2 . Likewise, the two outer vertical lines are host *B*’s interfaces B_1 and B_2 . Arrows leaving A_1 and A_2 are packets with one DATA chunk each, which are destined for B_1 and B_2 , respectively. The packets are labelled with their

corresponding Transmission Sequence Number (TSN). Arrows from B_1 and B_2 to A_1 and A_2 , respectively, are packets with one SACK chunk each. These SACKS are labelled with an ‘S’ followed by the cumulative ack contained in the SACK chunk. C_1 and C_2 denote *A*’s congestion windows – cwnds – for destinations B_1 and B_2 , respectively. C_1 and C_2 are in increments of the MTU and not in bytes.

Figure F2 presents RFC2960’s specified behavior when a timeout at *A* occurs due to network congestion; in other words, the *no failure* case. The scenario presented is simply an excerpt from an association. Hence, the initial TSN = 1 is an arbitrary assignment and not meant to imply the beginning of an association.

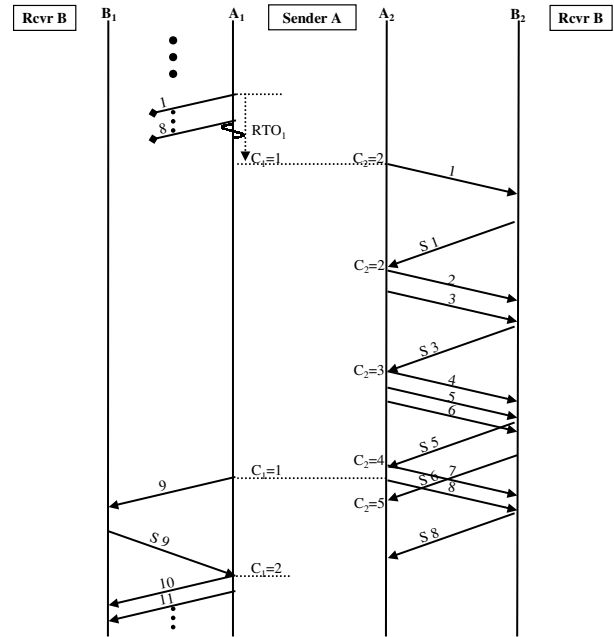


Figure F2: Specified behavior: no failure

At the beginning of this example, we arbitrarily have $C_1 = 8$, and host *A* transmits TSNs 1-8 to destination B_1 . The scenario assumes all 8 TSNs are lost due to congestion and never get acked. After one RTO, a timeout occurs at *A*. TSNs 1-8 are then marked for retransmission, and C_1 is reduced to $1 * MTU$. In accordance with RFC2960, TSNs 1-8 will be retransmitted to alternate destination B_2 , whose C_2 we assume to be $2 * MTU$.¹ Consequently, TSN 1 is retransmitted immediately. TSN 1’s ack allows TSNs 2 and 3 to be retransmitted. Later, the SACK for TSN 3 triggers a growth of C_2 to 3, and the retransmission of TSNs 4-6. Likewise, the arrival of a SACK for TSN 5 causes the retransmission of TSNs 7 and 8. At this point, all marked TSNs have been retransmitted, and the sender can begin sending *new* TSNs again. As shown and in accordance with RFC2960, new TSN 9 is transmitted on the original path to B_1 , and the association proceeds as usual.

¹We assume that destination B_2 has been idle long enough for its C_2 to have degraded to the specified minimum value: $2 * MTU$.

Now consider the case where an association experiences an event more severe than loss due to congestion; a timeout occurs because a failure makes the path to the primary destination unusable. Figure F3 begins to show the behavior that would occur in case of a failure, while Figures F4 and F5 complete the example for the short term and long term failure cases, respectively.

3.1 Short term failure

As shown in Figure F3, assume an earlier failure prevents B_1 from receiving and hence acking any of TSNs 1-8. In other words, B_1 becomes unreachable before TSN 1 arrives. A behaves the same as presented in Figure F2. In this failure scenario, however, sending TSN 9 to B_1 results in another timeout because the destination B_1 is unreachable. Since this timeout is the second one for B_1 , the RTO becomes twice what it was before. Notice that both destinations B_1 and B_2 remain idle for this entire second timeout period! Later, TSN 9 is retransmitted via the alternate path to B_2 , and TSN 10 is transmitted via the original path to B_1 .

Let us consider the short term failure case (Figure F3 continues into Figure F4). Assuming a short term failure, B_1 is restored and then TSN 10 later times out. In this case, B_1 is restored in time to receive and acknowledge TSN 11. No failover occurs in this example.

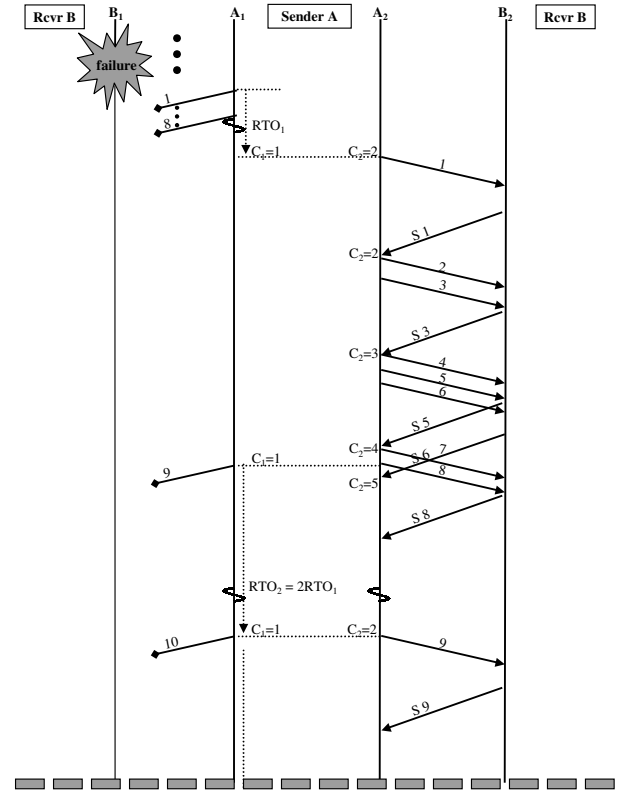


Figure F3: Specified behavior: failure template

3.2 Long term failure

Figure F5 continues Figure F3 in the case of a long term failure; in other words, B_1 is not restored. In this case, failover will occur. TSN 11 is retransmitted by A via the path from A_2 to B_2 , and as done previously, the next new TSN 12 is transmitted to B_1 . The sender repeats this behavior of transmitting one new TSN to B_1 , timing out, and then retransmitting it to B_2 . The cycle continues until the SCTP association determines that sufficient consecutive timeouts indicate the interface B_1 is in fact unreachable, at which point host A fails over to the alternate destination B_2 .

3.3 Discussion

In the case of long term failure, how long will it take for SCTP to give up on the primary destination and failover? RFC2960 states

“Each time the [retransmission] timer expires on any address, or when a HEARTBEAT sent to

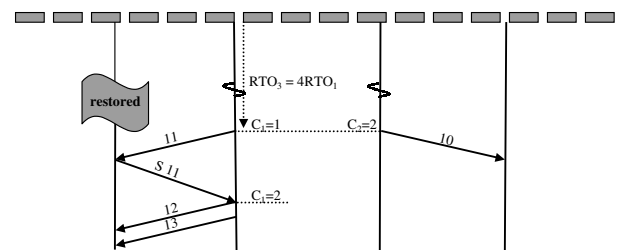


Figure F4: Specified behavior: short term failure

an idle address is not acknowledged within a RTO, the error counter of that destination address will be incremented. When the value in the error counter exceeds the protocol parameter 'Path.Max.Retrans' of that destination address, the endpoint should mark the destination transport address as inactive".

Since the RFC recommended value for Path.Max.Retrans is 5, failover occurs only after six consecutive timeouts. The best case is when the initial RTO is the minimum allowable value, RTO.Min. Assuming RTO.Min is the RFC recommended value of 1 second, the failover will take at *least* 63 seconds! In other words, at *most* four packets will be successfully transmitted within 60 seconds after the second timeout. For the majority of the time, the alternate path to B_2 is unused.

Although Path.Max.Retrans is a tunable parameter, the "average" implementation will use the RFC recommended value as the default. Even if the parameter is tuned, what should it be tuned to? Setting Path.Max.Retrans to 1 minimizes the amount of time an SCTP endpoint waits to mark a path as inactive. However, timeouts do not always indicate an unreachable destination; so, failing over due to only one timeout incorrectly interprets congestion as failure. A Path.Max.Retrans value of 2 *may* be more sensible, but still premature to assume that a destination is unreachable. As the Path.Max.Retrans value increases, it becomes more likely that SCTP will accurately mark a destination as inactive. But as pointed out earlier, being too conservative incurs a large penalty. Multihoming should be able to provide immediate seamless recovery in the case of failure of the primary address.

In the next section, we present a two-level threshold mechanism which we argue better utilizes the path to B_2 , and results in better overall SCTP end-to-end performance.

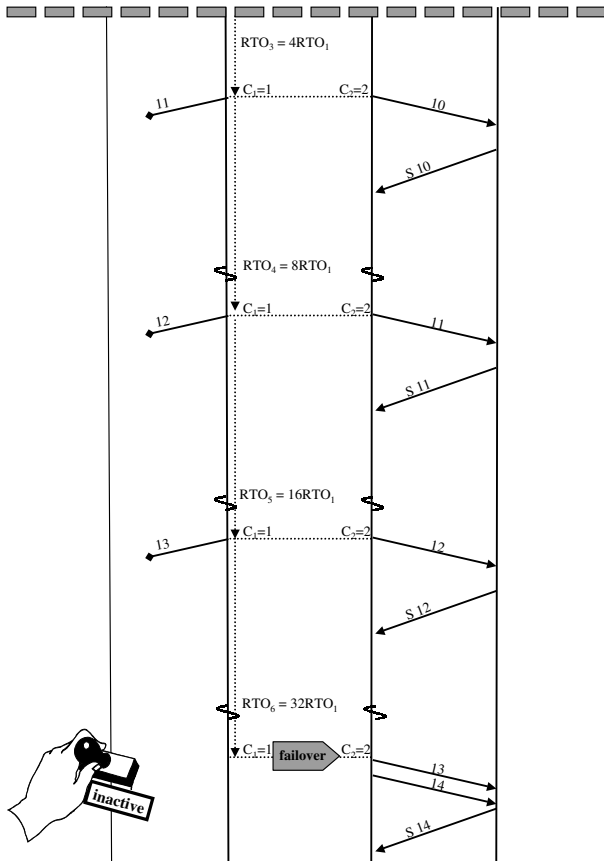


Figure F5: Specified behavior: long term failure

4 Two-level threshold

We investigate a two-level threshold recovery mechanism² to alleviate SCTP's current significant drop in throughput when failure occurs. Our mechanism's finite state machine for an arbitrary number of destinations, D_i , is shown in Figure F6. The states show the primary destination and their status (active/inactive). The destinations used for new transmissions and retransmissions are also shown.

The first threshold, α , determines when new transmissions are moved to an alternate destination path, initially on a temporary basis. Even while new transmissions are moved

²We credit Jacob Heitz for the initial concept of using a two-level threshold for failover (see the IETF tsvwg mailing list [2]).

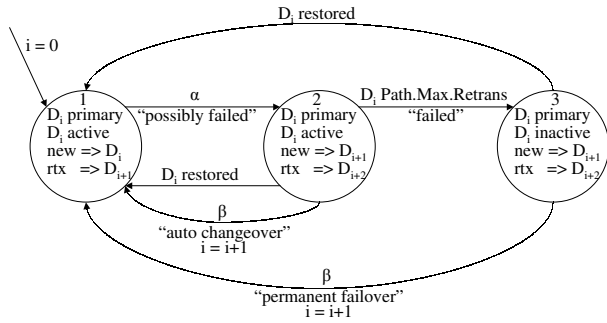


Figure F6: Two-level threshold recovery FSM

to an alternate path, the sender remains optimistic about the primary destination's reachability. The sender monitors the reachability of the primary destination using the normal SCTP HEARTBEAT mechanism. If a HEARTBEAT is successfully acked, the sender returns to using the primary destination for new transmissions. If, however, the primary destination does not become reachable in a timely manner, the sender's optimism eventually runs out.

It is the second threshold, β , that defines when the sender decides not to wait for the primary destination's restoration any longer. If the Path.Max.Retrans has not been reached, β triggers an *auto changeover* to the alternate destination the sender has been using since the α threshold. On the other hand, if Path.Max.Retrans has already been reached, β triggers a *permanent failover*. In either case, giving up on the primary destination causes the alternate destination to automatically become the primary destination.

We emphasize that automatically switching the primary destination is a new concept and is not currently supported by RFC2960. Also note that the parameter Path.Max.Retrans is independent of both thresholds. Path.Max.Retrans is still used to mark a destination as inactive and report a failure to the upper layer, while the α and β thresholds control the destination of new transmissions.

Let us reexamine SCTP's behavior using our proposed two-level threshold mechanism under the same three conditions considered in Figures F2-F5: no failure, short term failure, and long term failure. The thresholds chosen in our examples are $\alpha = 2$ timeouts, and $\beta = 5$ timeouts. Using these thresholds, the behavior of a congestion induced timeout does not change from RFC2960's behavior. Therefore, the case of no failure has the same behavior as presented in Figure F2.

For both short term and long term failures, Figure F7 shows the initial behavior. Figure F8 completes Figure F7 for short term, while Figure F9 completes Figure F7 for long term.

4.1 Short term failure

Figures F7/F8 show that the two-level threshold contrasts RFC2960's behavior presented previously in Figures F3/F4. When TSN 9 times out, the α threshold is reached. Thus, all new transmissions are now destined for B_2 , and only HEARTBEATS are sent to B_1 . Once B_1 is restored by acking a HEARTBEAT (shown in Figure F8 occurring after the third timeout but before the fourth timeout), SCTP returns to sending new transmissions to B_1 and only retransmissions to B_2 .

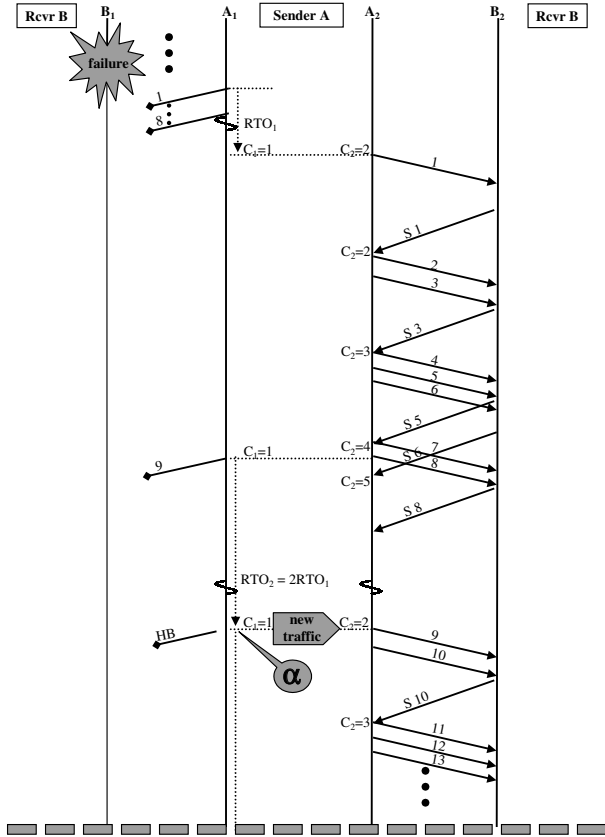


Figure F7: Two-level threshold ($\alpha = 2 \cdot RTO$, $\beta = 5 \cdot RTO$): failure template

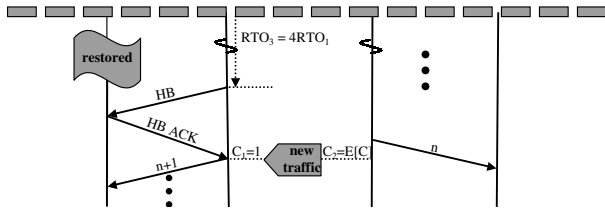


Figure F8: Two-level threshold ($\alpha = 2 \cdot RTO$, $\beta = 5 \cdot RTO$): short term failure

4.2 Long term failure

Figures F7/F9 show the two-level threshold behavior for long term failures. Here the association never returns to using B_1 for new transmissions. Because all HEARTBEAT attempts to reach B_1 fail, eventually the β threshold is reached and host A does an auto changeover to B_2 . Notice that in this case, a failover has been preempted since B_1 remains marked active until Path.Max.Retrans is reached.

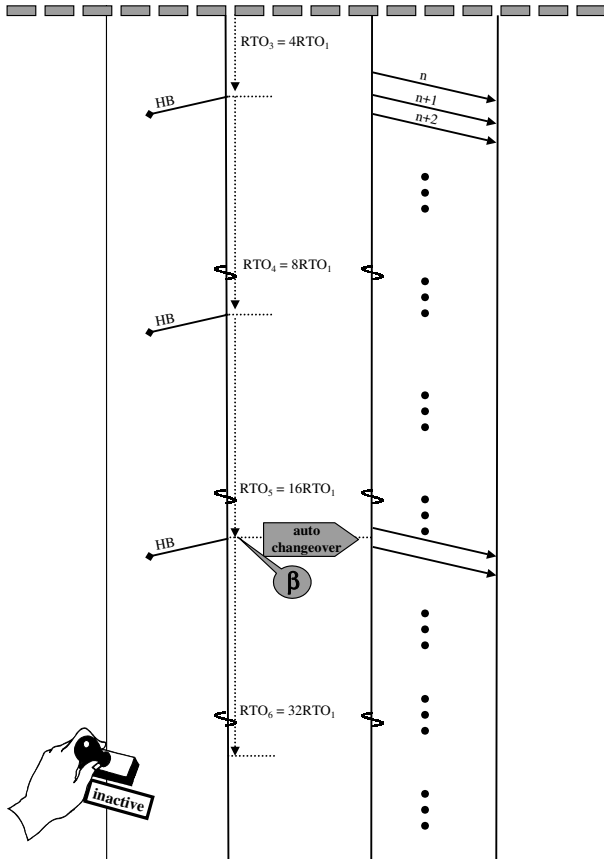


Figure F9: Two-level threshold ($\alpha = 2 * RTO$, $\beta = 5 * RTO$): long term failure

4.3 Discussion

The α threshold reduces the unnecessary idle time during the failure detection process by redirecting new traffic to an alternate destination before the primary is declared inactive. Hence, throughput is increased.

Also, once α is reached, the alternate destination's cwnd will continue to grow. Eventually, it may not make sense to move new traffic back to the primary destination (whose cwnd is $1 * MTU$) even if the primary does become active. Doing so will throttle the sending rate significantly. Our mechanism uses the β threshold to automatically change the primary and avoid throttling the sending rate. Again, throughput is increased.

5 Conclusion and future work

RFC2960 tightly couples failure detection and recovery. As a result, the recovery process cannot begin until failure detection is complete. We have decoupled the two and replaced the existing recovery system with a two-level threshold mechanism. Consequently, SCTP associations experience higher throughput for both short term and longer term failures.

The key to maximizing the performance gain is in accurately selecting the α and β threshold values. We are currently investigating via ns-2 simulations the tradeoffs of various thresholds settings.

6 Disclaimer

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U. S. Government.

7 References

- [1] R. Braden. Requirements for Internet hosts—communication layers. RFC1122, Internet Engineering Task Force (IETF), October 1989.
- [2] J. Heitz. SCTP multihoming issue. IETF Transport Area Working Group mailing list, August 2001. <ftp://ftp.ietf.org/ietf-mail-archive/tsvwg>.
- [3] R. Stewart and Q. Xie. *Stream Control Transmission Protocol (SCTP): A Reference Guide*. Addison Wesley, New York, NY, 2001.
- [4] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. Stream Control Transmission Protocol. Proposed standard, RFC2960, Internet Engineering Task Force (IETF), October 2000.