Retransmission Policies With Transport Layer Multihoming

Armando L. Caro Jr., Paul D. Amer, Janardhan R. Iyengar
Protocol Engineering Lab
Computer and Information Sciences Department
University of Delaware
{acaro, amer, iyengar}@cis.udel.edu

Randall R. Stewart
Cisco Systems Inc.
rrs@cisco.com

***Abstract*— We evaluate several retransmission policies for transport protocols that support multihoming, such as SCTP. We find that schemes that attempt to improve the chance of success by retransmitting to an alternate peer IP address often degrade performance. Our results show that for better performance, new data transmissions and retransmissions should be sent to the same peer IP address. We also find that our Multiple Fast Retransmit algorithm further improves performance by reducing the number of timeouts. Since our results assume reachability of all peer IP addresses, we conclude with suggestions for scenarios where failures are possible. We suggest compromising some of the performance improvements to avoid performance degradation during failures.**

I. 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 [3]. 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 between two endpoints. Any time either endpoint's IP address becomes inaccessible, perhaps due to interface failure or path outage, TCP's connection will timeout and abort, thus forcing the application to recover. This recovery overhead and associated delay can be unacceptable for mission critical applications.

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.

To address TCP's shortcoming, the Stream Control Transmission Protocol (SCTP) has been designed with fault tolerance in mind. SCTP is an IETF standards track transport layer protocol. Telephony signaling applications originally motivated SCTP's development, but its design makes it suitable as a general purpose transport protocol and an alternative to TCP. SCTP is a reliable, message-oriented data transport protocol that provides resistance to SYN flooding attacks, supports multiple streams to prevent head-of-line blocking, and supports multihoming for end-to-end network fault tolerance [11].

Transport layer multihoming provides end-to-end fault tolerance which is crucial for mission critical applications. SCTP multihoming allow connections, 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. Higher layer applications need not be aware of the destination IP address change, as should be expected in a truly fault tolerant system.

Currently, SCTP uses multihoming for redundancy purposes only and not for load balancing. Each endpoint chooses a single destination address as the primary destination address, which is used for all new data during normal transmission. Retransmitted data use alternate peer IP address(es). RFC2960 [12] states in Section 6.4 "when its peer is multihomed, an endpoint SHOULD try to retransmit [data] to an active destination transport address that is different from the last destination address to which the [data] was sent."

SCTP's current retransmission policy attempts to im-

prove the chance of success by sending all retransmissions to an alternate destination address [11]. The underlying assumption is that loss indicates either that the destination address used is unreachable, or the destination's network path is congested. Hence, SCTP retransmits to an alternate destination address in attempt to avoid another loss of the same data. However, SCTP's current retransmission policy has been shown to actually degrade performance in many circumstances [5].

Given the analysis in [5], we evaluate five potential solutions to the problem. Our results show that all five solutions significantly improve performance. For better performance, new data transmissions and retransmissions should be sent to the same peer IP address. We also find that our Multiple Fast Retransmit algorithm further improves performance by reducing the number of timeouts. Since our results assume reachability of all peer IP addresses, we conclude with suggestions for scenarios where failures are possible. We suggest compromising some of the performance improvements to avoid performance degradation during failures.

We begin in Section II by describing the problem with SCTP's current retransmission policy in more detail. Section III describes five alternative retransmission schemes as potential solutions. We comparatively evaluate these schemes using a simulation methodology described in Section IV. The results and analysis are presented in Section V. Section VI concludes the paper with additional discussion on a couple of our solutions and suggestions for scenarios where failures are possible.

II. The Problem

To explain the problem with SCTP's current retransmission policy, we use the example multihoming topology shown in Figure 1. Hosts A and B are both multihomed. Suppose there exists an SCTP association between them: $(\{A_1, A_2\}, \{B_1, B_2\})$. Also suppose that Host A is the sender and B_1 is the primary destination. According to RFC2960 [12], Host A sends all new data to B_1 (assuming that B_1 is reachable). If any of these data packets are lost, their retransmissions are sent to B_2. Similarly, if any of these retransmissions to B_2 are lost, the data packets are retransmitted again to B_1. Subsequent retransmissions of the same data continue changing the destination address until the data successfully reaches the peer endpoint – Host B.

Intuition tells us that when the loss conditions are worse on an alternate destination's path than on the primary destination's path, SCTP's current retransmission policy will not perform well. Similarly, we expect that

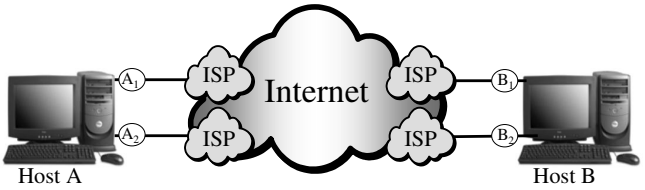

Fig. 1. Example multihoming topology

when the conditions are better on an alternate destination's path, performance will improve if the alternate destination is used for retransmissions. However, the results in [5] show that often the latter is not the case.

Analysis in [5] reveals that two features of SCTP contribute to these counter-intuitive results: (1) one time only fast retransmission, and (2) Karn's algorithm. As in TCP, fast retransmissions and timeouts are the two mechanisms used in SCTP to recover from loss. According to RFC2960 [12] and the SCTP Implementer's Guide [10], data that has been fast retransmitted may not be fast retransmitted again. Hence, retransmissions of retransmissions may only be triggered by timeouts.

In current SCTP, all data traffic to the alternate destination are retransmissions, and if lost, must wait for a timeout to be retransmitted again. In and of itself, this requirement is not a problem; the same would be true if retransmissions used the same destination as the new data transmissions. Unfortunately due to Karn's algorithm, successful retransmissions to the alternate destination cannot be used to update the round-trip time (RTT) estimation of the alternate destination's path. Timeouts on retransmissions, however, exponentially increase the retransmission timeout (RTO) of the alternate destination's path. The only traffic on the alternate destination's path which can update the RTT estimate are heartbeat probes used to determine destination reachability, but these heartbeats are transmitted infrequently (RFC2960 recommends a jittered heartbeat interval of once per RTO of the destination address plus a random value between 15-45 seconds.). In many cases, the alternate path's RTO is exponentially increased more frequently than can be reduced by an RTT estimate. The result is an overly conservative (i.e., too large) RTO on the alternate destination's path for the majority of the association. Thus, losses of retransmissions are expected to significantly increase the total transfer time.

III. Potential Solutions

A. Solution 1: Retransmit to Same Destination

In this solution, all retransmissions are sent to the same destination as their original transmissions. Alternate destinations are not used until a detection of failure results in failover. Solution 1 ensures that if a timeout occurs, the timeout will be for the destination with a more accurate RTO, thus avoiding unnecessary long delays in retransmission.

Using the same destination for retransmissions has the added advantage that the primary destination's cwnd benefits from successful retransmissions. While a successful retransmission sent to an alternate destination benefits the alternate destination's cwnd, increasing the alternate destination's cwnd is less advantageous since little data is sent to the alternate destination. Furthermore, timeouts on retransmissions sent to an alternate destination delay possible cwnd increases for the primary destination, because the cwnd cannot be increased for any destination until a new cumulative ack arrives at the sender.

The disadvantage of Solution 1 is that fewer packets are successfully transmitted in cases where the primary destination is unreachable. In such cases, lost packets would be continually retransmitted to the primary destination and lost until a failover occurs.

B. Solution 2: Heartbeat After RTO

Solution 2, named Heartbeat After RTO, extends the current retransmission policy. In addition to SCTP's current policy of retransmitting to an alternate destination on a timeout, a heartbeat is sent immediately to the destination on which a timeout occurred. Extra heartbeats provide a mechanism for a sender to update an alternate destination's RTT estimate more frequently, thus resulting in a better RTT estimate on which to base the RTO value.

For example, suppose a packet is lost in transit to the primary destination, and later gets retransmitted to an alternate destination. Also suppose that the retransmission times out. The lost packet is retransmitted again to yet another alternate destination (if one exists; otherwise, the primary). More importantly, a heartbeat is also sent to the alternate destination which timed out. If the heartbeat is successfully acked, that destination acquires an additional RTT measurement to help reduce its recently doubled RTO.

The advantage of Solution 2 over Solution 1 is that an alternate destination is still used for retransmissions. Hence, Solution 2 does not lose the chance to successfully transmit some packets when the primary destination is unreachable. The drawback of Solution 2 is that the sender still has few samples to estimate an alternate destination's RTT. So while the RTT estimate is better, it remains a poor estimate.

C. Solution 3: Timestamps

Solution 3 introduces timestamps into each packet, thus allowing a sender to disambiguate original transmissions from retransmissions. By removing retransmission ambiguity, Karn's algorithm can be eliminated, and successful retransmissions on the alternate path can be used to update the RTT estimate and keep the RTO value more accurate. Solution 3 provides more samples for alternate destination(s) to update their RTT estimate.

We designed a 12-byte TIMESTAMP chunk (see Figure 2) that could be included in each SCTP packet. Our timestamp option is modeled after the TCP Timestamp option [8]. The *timestamp* field is populated by the sender in packets containing DATA chunks. The receiver echoes this timestamp back in the *timestamp echo* field. Since traffic may be bi-directional, any combination of these two fields may be in use for a given packet. The low order two bits of the *flags* field specify which TIMESTAMP chunk fields have been populated:

00 Neither field is used.
01 *timestamp* field is used.
10 *timestamp echo* field is used.
11 Both *timestamp* and *timestamp echo* fields are used.

Due to SCTP's use of SACKs, a sender can always identify which newly acked TSNs (Transmission Sequence Number) correspond to the timestamp echoed in the packet.

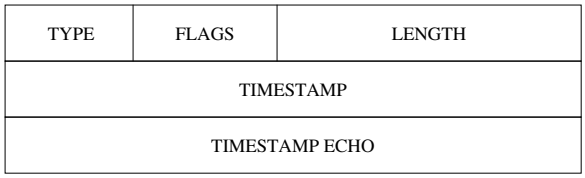

Fig. 2. TIMESTAMP chunk

Solution 3 has the advantage over Solution 2 that the sender has more samples to estimate an alternate destination's RTT. The disadvantage is that each packet has an additional overhead of 12 bytes.

D. Solution 4: Multiple Fast Retransmit

Solution 4, named Multiple Fast Retransmit, attempts to minimize the number of timeouts which occur. Currently, SCTP may only Fast Retransmit a TSN once [10]. If a Fast Retransmitted TSN is lost, a timeout is necessary to retransmit the TSN again. The Multiple Fast Retransmit algorithm allows the same TSN to be Fast Retransmitted several times if needed. Without the Multiple Fast Retransmit algorithm, a large window of outstanding data may generate enough SACKs to incorrectly trigger more than one Fast Retransmit of the same TSN in a single RTT. To avoid these spurious Fast Retransmits, the Multiple Fast Retransmit algorithm introduces a *fastRtxRecover* state variable for each TSN Fast Retransmitted. This variable stores the highest outstanding TSN at the time a TSN is Fast Retransmitted. Then, only SACKs which newly ack TSNs beyond *fastRtxRecover* can increment the missing report for the Fast Retransmitted TSN. If the missing report threshold for the Fast Retransmitted TSN is reached again, the sender has enough evidence that this TSN was lost and can be Fast Retransmitted again.

Solution 4 has the advantage over Solution 3 that no additional chunk (i.e., packet overhead) is needed. The disadvantage is that the sender still suffers from few RTT estimates of alternate destinations. Also, opportunity to use the Multiple Fast Retransmit algorithm may not arise, as in the case of application-limited periods where the sender sends less than a full window of data.

E. Solution 5: Retransmit to Same Destination with Multiple Fast Retransmit

Solution 5 combines Solutions 1 and 4: the Retransmit to Same Destination policy with the Multiple Fast Retransmit algorithm. The Multiple Fast Retransmit algorithm attempts to eliminate timeouts from occurring, but if timeouts do occur, the Retransmit to Same Destination policy ensures that the timeouts occur on the destination with a more accurate RTO.

IV. Methodology

We evaluate our five potential solutions using the ns-2 network simulator [2] with an SCTP module [6] available as a patch from the Protocol Engineering Lab at the University of Delaware. Figure 3 illustrates the network topology used: a dual-dumbbell topology whose core links have a bandwidth of 10Mbps and a one-way propagation delay of 25ms. Each router, R, is attached to five edge nodes. One of these five nodes is dual-homed node for an SCTP agent, while the remaining four nodes are single-homed and introduce cross-traffic that creates loss for the SCTP traffic.

The links to the dual-homed nodes have a bandwidth of 100Mbps and a one-way propagation delay of 10ms. The single-homed nodes also have 100Mbps links, but their propagation delays are randomly chosen from a uniform distribution between 5-20ms. The end-to-end one-way propagation delays range between 35-65ms. These delays roughly approximate reasonable Internet delays for distances such as coast-to-coast of the continental US, and eastern US to/from western Europe. Also, each link (both edge and core) has a buffer size twice the link's bandwidth-delay product.

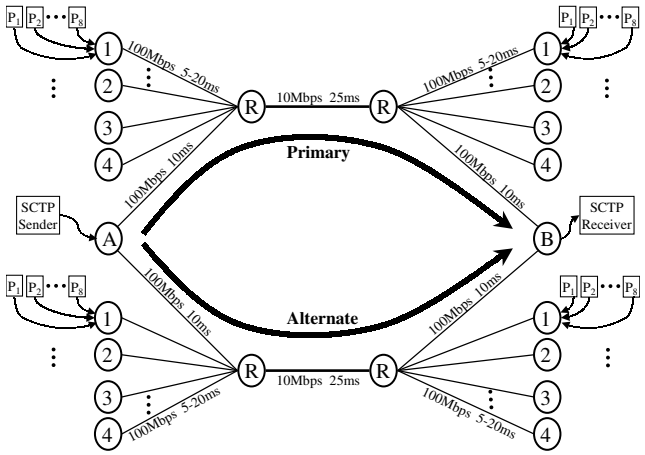

Fig. 3. Simulation network topology

Our configuration has two SCTP endpoints (sender A, receiver B) on either side of the network, which are attached to the dual-homed edge nodes. A has two paths, labeled primary and alternate, to B. Each single-homed edge node has eight traffic generators, each introducing cross-traffic based on a Pareto distribution. The cross-traffic packet sizes are chosen to resemble the distribution found on the Internet: 50% are 44B, 25% are 576B, and 25% are 1500B [1], [7]. The result is an SCTP data transfer over a network with self-similar cross-traffic, which resembles the observed nature of traffic on data networks [9].

We simulate a 4MB file transfer with different network conditions, controlled by varying the load introduced by cross-traffic. All loss experienced is due to congestion only. The aggregate levels of cross-traffic on each path range from 5Mbps to 11Mbps. Although we independently control the levels of cross-traffic on each

of the core links, the controls for the cross-traffic on each forward-return path pair are set the same. Each simulation has three parameters:

1) level of cross-traffic (in Mbps) on the primary path
2) level of cross-traffic (in Mbps) on the alternate path
3) retransmission policy (current, or one of the five potential solutions)

V. Results and Analysis

Figure 4 illustrates a set of our results. This figure summarizes results when the primary path experiences a 3% loss rate. The x-axis represents the possible loss rates on the alternate path, ranging from 0% to 10%. The graph compares the time to transfer a 4MB using SCTP's current retransmission policy versus Solution 4 and 5. We omitted Solutions 1, 2, and 3 from Figure 4 for clarity. The interested reader may refer to the color graphs in the appendix of [4] for the complete set of results.

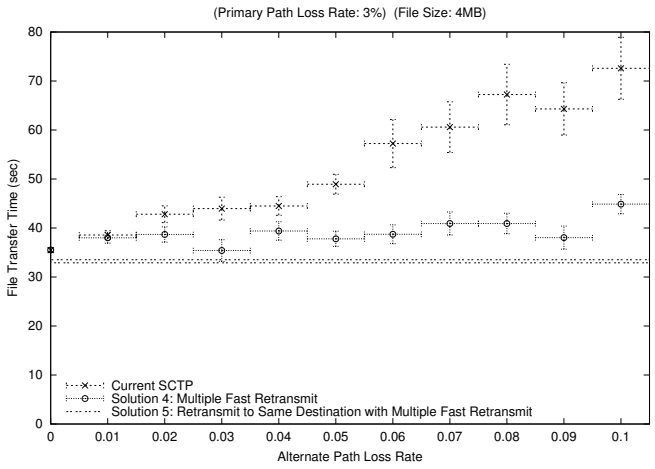

Fig. 4. 4MB file transfer with 3% loss rate on primary path

Transfers which retransmit to the alternate destination (i.e., Solutions 2, 3, and 4) are grouped by ranges of alternate path loss rates $\{0\%, 0-0.5\%, 0.5-1.5\%, 1.5-2.5\%, ..., 9.5-10.5\%\}$. The graph depicts the mean and 90% confidence interval for each of these groups. These statistics are calculated using an acceptable error of 10% of the mean. That is, we ran enough simulations to estimate the mean and 90% confidence interval with an acceptable error of at most 10% of the mean. For example, consider the results of the current SCTP retransmission policy in Figure 4. The value 0.02 on the x-axis indicates that when the alternate path has loss between 1.5 and 2.5%, the time to transfer a 4MB file using current SCTP is about 42.8 seconds on average, with the 90% confidence interval being (41.1 - 44.5) seconds.

Transfers which retransmit to the same destination (Solutions 1 and 5) never use the alternate path, and therefore are unaffected by (i.e., independent of) the alternate path's loss rate (i.e., the x-axis). These transfer times are represented as a band parallel to the x-axis. This band outlines the upper and lower bounds of the 90% confidence interval. For example, the two horizontal dotted lines which form a band in Figure 4 indicate that using Solution 5 requires an average time of 33.2 seconds with the 90% confidence interval being (32.9 - 33.5) seconds.

We collected results for primary and alternate path loss rates 0-10%. Due to space constraints in this paper, we do not include all eleven graphs for the different primary path loss rates, but the trend in Figure 4 is similar to the other graphs (refer to [4] for the complete set of graphs). As Figure 4 shows, Solution 4 improves the performance of SCTP's current policy of retransmitting to an alternate destination. Solutions 2 and 3 (not shown) provide similar improvements. However, retransmitting to the same destination (Solution 1 and 5) yields better performance than Solutions 2-4. The best performance is provided by Solution 5 (see Figure 4), which combines Solution 1 and 4.

In fact, we observe from analyzing all of our results (including graphs not shown) that retransmitting to the alternate path does not perform better than Solution 5 unless the primary path has at least a 7% loss rate (see Figure 5), and even then, the benefits are only for relatively low loss rates (3% or less) on the alternate path. These results indicate that the three potential solutions that maintain the current retransmission policy (Solutions 2-4) succeed at improving performance significantly, but do not provide the best performance. The best performance is provided by Solution 5: the combination of the Retransmit to Same Destination policy with our Multiple Fast Retransmit algorithm. As explained in Section III-E, the Multiple Fast Retransmit algorithm attempts to eliminate timeouts from occurring, but if timeouts do occur, the Retransmit to Same Destination policy ensures that the timeouts occur on the destination with a more accurate RTO.

VI. Discussion and Conclusion

Although our results show that Solution 4 alone improves performance, Solution 4 should only be used together with Solution 1 (i.e., Solution 5). Using Solution

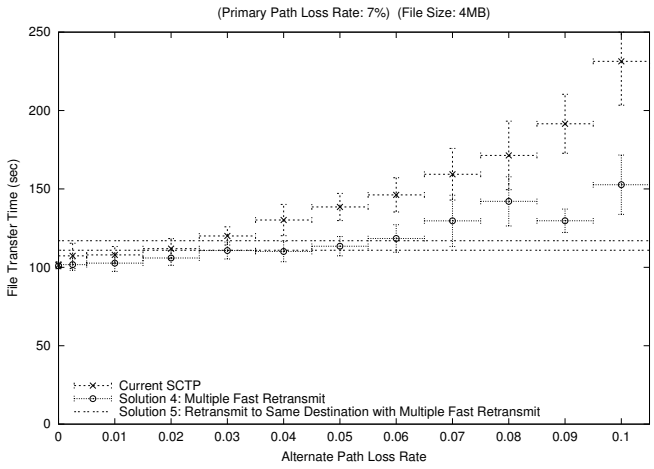

Fig. 5. 4MB file transfer with 7% loss rate on primary path

4 without Solution 1 may cause spurious Fast Retransmits when the paths have different end-to-end delays. To understand why, suppose that the alternate destination's path has a longer delay than the primary. Then, new data transmissions sent to the primary destination after retransmissions sent to the alternate destination may arrive sooner at the receiver. Hence, our Multiple Fast Retransmit algorithm will incorrectly trigger subsequent retransmissions. On the other hand, these spurious retransmissions will not occur if the Retransmit to Same Destination policy of Solution 1 is used.

As explained in Section III-A, retransmitting data to the same destination as the original transmissions has the disadvantage that fewer packets are successfully transmitted if the primary destination becomes unreachable. The SCTP authors intentionally included a retransmission policy which fully utilizes the network redundancy available on multihomed hosts. The intended benefits of the retransmission scheme assume that loss indicates either that the destination address used is unreachable, or its network path is congested. By retransmitting to an alternate peer IP address, SCTP attempts to avoid another loss of the same data, and the sender has the opportunity to successfully transmit some data until a failover occurs.

We suggest Solution 5 with a modification to avoid performance degradation during failures. Instead of all retransmissions following the Retransmit to Same Destination policy of Solution 1, only Fast Retransmissions should follow this policy. Retransmissions triggered by timeouts should follow SCTP's current policy; that is, they should be sent to an alternate destination. This modification compromises some of the performance benefits of Solution 5, but Solutions 2 and/or 3 can be incorporated to further improve the performance when timeout-induced retransmissions are sent to an alternate destination.

ACKNOWLEDGEMENTS

We credit Iván Arias Rodríguez with the original observation that the Retransmit to Same Destination policy has the added benefit of crediting the cwnd of the destination with more traffic.

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.

REFERENCES

[1] CAIDA: Packet Sizes and Sequencing, Mar 1998. http://traffic.caida.org.

[2] UC Berkeley, LBL, USC/ISI, and Xerox Parc. ns-2 documentation and software, Version 2.1b8, 2001. http://www.isi.edu/nsnam/ns.

[3] R. Braden. Requirements for Internet hosts–communication layers. RFC1122, Internet Engineering Task Force (IETF), October 1989.

[4] A. Caro, P. Amer, J. Iyengar, and R. Stewart. Retransmission Policies with Transport Layer Multihoming. Tech Report TR2003-05, CIS Dept, University of Delaware, March 2003.

[5] A. Caro, P. Amer, and R. Stewart. Transport Layer Multihoming for Fault Tolerance in FCS Networks. In *MILCOM 2003*, Boston, MA, October 2003.

[6] A. Caro and J. Iyengar. ns-2 SCTP module, Version 3.2, December 2002. http://pel.cis.udel.edu.

[7] K. Claffy, G. Miller, and K. Thompson. The Nature of the Beast: Recent Traffic Measurements from an Internet Backbone. *INET 1998*, April 1998.

[8] V. Jacobson, R. Braden, and D. Borman. TCP Extensions for High Performance. RFC1323, Internet Engineering Task Force (IETF), May 1992.

[9] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the Self-similar Nature of Ethernet Traffic. In *ACM SIGCOMM 1993*, San Francisco, CA, September 1993.

[10] R. Stewart, L. Ong, I. Arias-Rodriguez, K. Poon, P. Conrad, A. Caro, and M. Tuexen. Stream Control Transmission Protocol (SCTP) Implementer's Guide. draft-ietf-tsvwg-sctpimpguide-08.txt, Internet Draft (work in progress), Internet Engineering Task Force (IETF), March 2003.

[11] R. Stewart and Q. Xie. *Stream Control Transmission Protocol (SCTP): A Reference Guide*. Addison Wesley, New York, NY, 2001.

[12] 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. RFC2960, Internet Engineering Task Force (IETF), October 2000.