TRANSPORT LAYER MULTIHOMING FOR FAULT TOLERANCE IN FCS NETWORKS[*]

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

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

ABSTRACT

We document a potential inefficiency in the current SCTP retransmission policy. The current scheme intends to improve the chance of success by exploiting the redundant paths between multihomed endpoints, but we have found that the current SCTP retransmission policy often degrades performance. We comparatively evaluate an alternative retransmission policy and show that the current SCTP retransmission policy unexpectedly performs worse under certain conditions. Our analysis exposes the problem and we discuss four possible solutions.

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 [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, radio channel interference, or moving out of range, 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 in military battlefield communications where responsiveness is crucial.

To address TCP's shortcoming, the Stream Control Transmission Protocol (SCTP) has been designed with fault tolerance in mind. SCTP is an IETF (Internet Engineering Task Force) 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 fault tolerance.

Transport layer multihoming provides the network level fault tolerance which is crucial for survivability and persistent on-the-move sessions in FCS (Future Combat Systems) networks. 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 are unaware 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 data during normal transmission. Retransmitted data use alternate peer IP address(es). RFC2960 [11] states in Section 6.4 "when its peer is multi-homed, an endpoint SHOULD try to retransmit [data] to an active destination transport address that is different from the last destination

[*] 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.

address to which the [data] was sent."

SCTP's current retransmission policy attempts to improve the chance of success by sending all retransmissions to an alternate destination address [10]. The underlying assumption is that loss indicates either that the destination address used is unreachable, or its network path is congested. However, in wireless networks, such as in FCS networks, noisy channels significantly contribute to loss. In this case, retransmitting to an alternate destination may not increase the chance of success.

Battlefield applications are likely to experience high loss rates and require many retransmissions. Hence, while of less importance to the Internet in general, the performance of retransmissions is an important issue for persistent on-the-move sessions in FCS networks.

Regardless of the reason for loss, we have found that SCTP's current retransmission policy may actually degrade performance – even in the case of congestion induced loss. This paper documents the potential inefficiency in the current SCTP retransmission policy and evaluates an alternative policy. We simulated data transfers between multihomed hosts under varying loss rates. We compare transfers using the current SCTP that retransmits to an alternate destination versus a modified SCTP that retransmits to the same destination. Initial results show the modified SCTP generally provides improved performance. Under certain conditions, however, the current retransmission policy performs better. Further research is needed to improve the retransmission mechanism in general, and for FCS networks in particular.

We begin in Section 2 by describing the simulation environment used to gather data. Section 3 presents the results and analysis. We discuss some possible solutions in Section 4 and conclude the paper in Section 5.

2 METHODOLOGY

With support from the CTA (Collaborative Technology Alliance) Program, the Protocol Engineering Lab (PEL) at the University of Delaware (UD) implemented an SCTP module [5] for the ns-2 network simulator [2]. This software is being used by over 50 researchers for simulating SCTP behavior. At UD, we investigated the performance of data transfers between multihomed hosts under varying loss rates.

Figure 1 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 endpoint, 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.

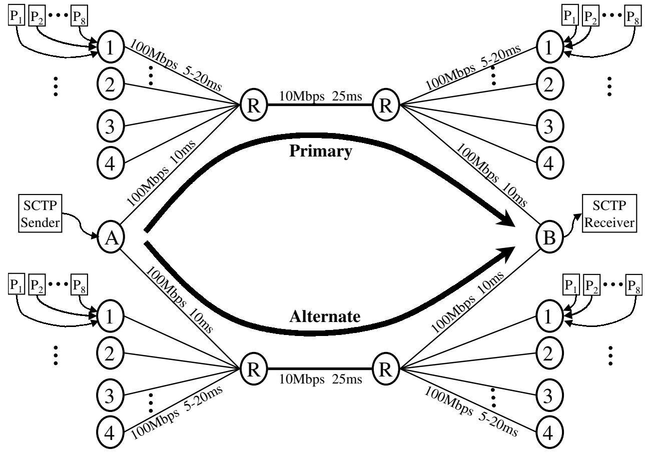

Figure 1: Simulation network topology

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, 6]. 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 [7].

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 SCTP (retransmit to alternate destination) versus modified SCTP (retransmit to same destination)

3 RESULTS AND ANALYSIS

Our results compare the transfer times using two different retransmission policies under various loss rates. The first retransmission policy, labeled "current", is SCTP's current scheme of sending all retransmissions on a different path than used previously. The other policy, labeled "modified", simply sends all retransmissions to the same destination used for the original transmission. The loss rate is calculated as the number of SCTP packets dropped divided by the number of SCTP packets transmitted.

Figure 2 presents the results for runs with a 3% loss rate on the primary path. The graph compares the file transfer time using the "current" versus "modified" SCTP at various loss rates on the alternate path. Transfers using the "modified" SCTP never use the alternate path and therefore are unaffected by the alternate path's loss rate. 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. That is, we are 90% confident that the average transfer time lies between 34.3 and 35.1 seconds.

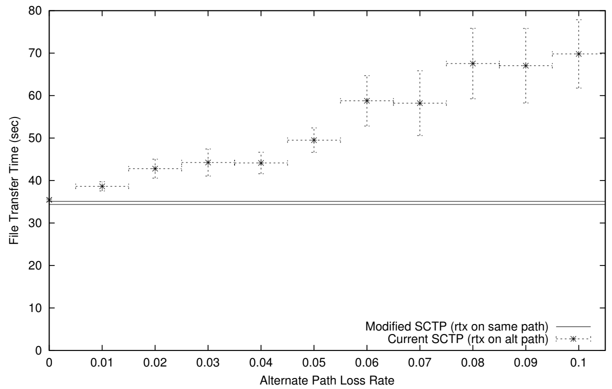

Figure 2: 4MB file transfer with 3% loss rate on primary path

The completion times for transfers using the "current" SCTP which retransmit on the alternate path are grouped by ranges of alternate path loss rates. The graph depicts the mean and the 90% confidence interval for each of these groups. The 90% confidence interval was 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, the value 0.02 on the x-axis indicates that when the alternate path has between 1.5 and 2.5% loss, the time to transfer a 4MB file is on average about 42.8 seconds with a 90% confidence interval between 41.1 and 44.5 seconds. As the graph shows, the "modified" SCTP performs better for all alternate path loss rates except 0%. When the alternate path's loss rate is 0%, both retransmission policies perform similarly.

Figure 3 shows results for transfers when the loss rate on the primary path is 8%. When the loss rate on the alternate path is less than about 4-5%, the "current" SCTP retransmission policy performs better. At higher loss rates on the alternate path, the "modified" SCTP yields superior performance.

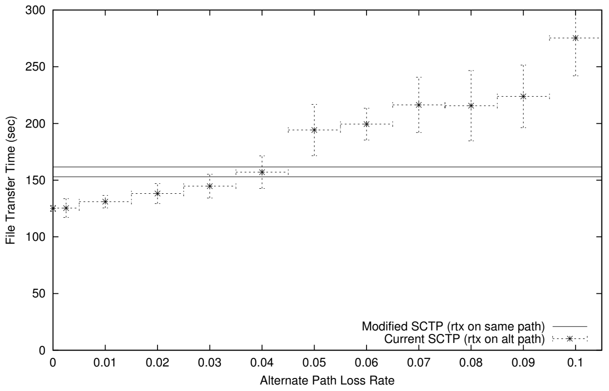

Figure 3: 4MB file transfer with 8% loss rate on primary path

We collected results for loss rates for 0-10%. Due to space constraints, we could not include all graphs, but the trend remains the same. For every primary path loss rate, the "modified" SCTP begins performing better at some threshold. We expected the threshold to be when the primary path's loss rate becomes greater than the loss rate on the alternate path. It is interesting that even when the loss rate on alternate path is less than on the primary path, the file transfer often takes more time when retransmissions are sent on an alternate path. This unexpected behavior is seen in Figures 2 and 3. For example, in Figure 3 at 8% loss on the primary path and 5% loss on the alternate path, it is faster to use the "modified" scheme of only using the primary path. This behavior is not what the SCTP authors expected when specifying the current retransmission policy.

Intuition tells us that when the loss conditions are worse on the alternate path than on the primary path, the "current" retransmission policy will not perform well. We also expect that when the conditions are better on the alternate path, performance will improve if the alternate path is used for retransmissions. However, our results show that often the latter is not the case.

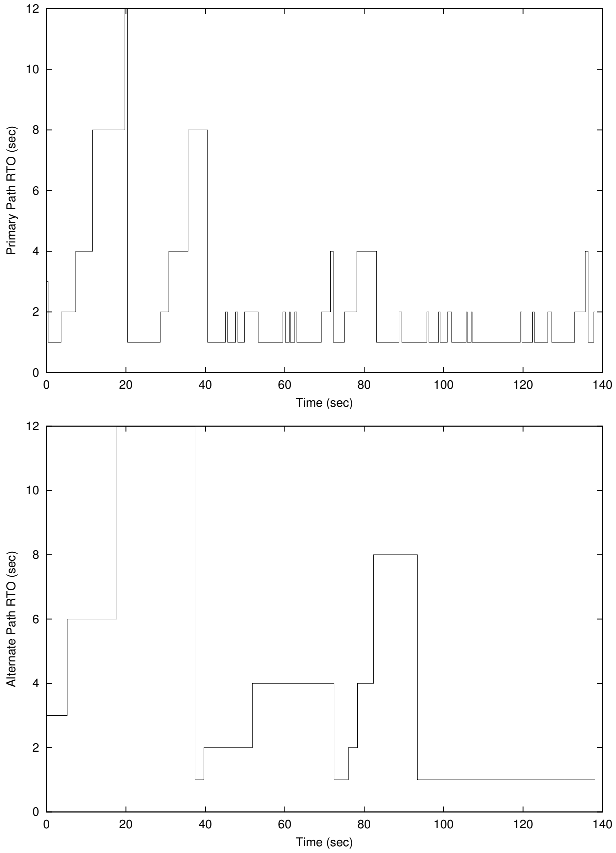

Figure 4: Example RTO dynamics of a 4MB file transfer with 8% primary path loss rate and 5% alternate path loss rate

There are two features of SCTP which contribute to our 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. Any data which has been fast retransmitted, may not be fast retransmitted again [9]. Subsequent retransmissions of the same data may only be triggered by timeouts. Hence, all data traffic on the alternate path are retransmissions, and if lost, must wait for a timeout to be retransmitted again. In and of itself, this requirement is not the problem; the same would be true if the retransmissions used the same path as the original transmissions. Due to Karn's algorithm, successful retransmissions on the alternate path cannot be used to update the round-trip time (RTT) estimation for the alternate path. Timeouts on retransmissions, however, are used to exponentially increase the retransmission timeout (RTO). The only traffic on the alternate path which can update the RTT estimate are the heartbeat probes used to determine destination reachability, but these heartbeats are transmitted fairly infrequently (RFC2960 recommends every 30 seconds with a random jitter of +/- 0 to 15 seconds). In many cases the 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 path for the majority of the association.

Figure 4 illustrates the dynamics of the RTOs for the primary path (8% loss rate) and the alternate path (5% loss rate) during a 4MB file transfer using the "current" SCTP. This specific transfer sent a total of 2,889 original transmissions on the primary path, of which 229 had to be retransmitted on the alternate path. Of those retransmissions, 14 were retransmitted again on the primary path.[1] The RTO for the primary path stays low during most of the transfer, because any successful original transmission on the primary path updates the RTT estimation and reduces the RTO (most likely back to 1 second). The average RTO for the primary path is 2.3 seconds, while the alternate path has an average RTO of 5.9 seconds. In only three occasions does the RTO for the alternate path get reduced, which means that during the entire transfer only three heartbeats were successfully acked. On the other hand, the graph shows seven timeouts exponentially increasing the RTO for the alternate path.

4 POTENTIAL SOLUTIONS

We consider four potential solutions to improve SCTP's performance during loss scenarios. Solution 1 uses the "modified" SCTP's retransmission scheme presented in Section 3, which sends retransmissions to the same destination as their original transmissions. As discussed in Section 3, this solution 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.

Solutions 2-4 use the "current" SCTP's retransmission scheme, but improve performance with enhancements to this retransmission scheme. After a timeout, Solution 2 sends a heartbeat probe immediately to the destination on which a timeout occurred. These extra heartbeat(s) provide a mechanism for the sender to update the alternate destination(s)' RTT estimate more frequently.

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

[1] The network only lost 222 SCTP packets on the primary path and 13 SCTP packets on the alternate path. The sender spuriously retransmitted the others.

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 at the expense of additional overhead in each packet.

Solution 4, named Multiple Fast Retransmit, attempts to minimize the number of timeouts which occur. Currently, SCTP may only Fast Retransmit a TSN once [9]. 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.

5 CONCLUSION AND FUTURE WORK

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. In FCS networks, however, wireless links introduce an additional loss factor: noisy channels. It is important to understand the effects of the current SCTP retransmission policy under such conditions.

Before the retransmission policy can be optimized for wireless networks, we need to ensure that the protocol performs well on wired networks. The results presented in this paper show that there exists a inefficiency in the current SCTP retransmission policy. Our analysis explains that the retransmission ambiguity problem causes the current SCTP retransmission policy to perform surprisingly worse than expected.

We propose four potential solutions which should make SCTP's current retransmission scheme perform better. Since the acceptance of this paper, we have investigated these solutions, and the results are available in [4].

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.

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. In *ICON 2003*, Sydney, Australia, September 2003.

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

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

[7] 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.

[8] R. Ludwig and R. Katz. The Eifel Algorithm: Making TCP Robust Against Spurious Retransmissions. In *ACM Computer Communications Review*, January 2000.

[9] 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.

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

[11] 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.