END-TO-END FAILOVER THRESHOLDS FOR TRANSPORT LAYER MULTIHOMING[*][†]

Armando L. Caro Jr.
Protocol Engineering Lab
Computer and Information Sciences
University of Delaware
acaro@cis.udel.edu

Paul D. Amer
Protocol Engineering Lab
Computer and Information Sciences
University of Delaware
amer@cis.udel.edu

Randall R. Stewart
Transport Technologies
Internet Technologies Division
Cisco Systems, Inc.
rrs@cisco.com

ABSTRACT

SCTP's multihoming failure detection time depends on three tunable parameters: RTO.min (minimum retransmission timeout), RTO.max (maximum retransmission timeout), and Path.Max.Retrans (threshold number of consecutive timeouts that must be exceeded to detect failure). RFC2960 recommends Path.Max.Retrans $= 5$*, which translates to a failure detection time of at least 63 seconds – unacceptable to many applications. This research investigates the tradeoff between a more aggressive (i.e., lower) threshold, and spurious failovers for the application of bulk file transfer. We surprisingly find that spurious failovers do not degrade overall performance, and sometimes actually improve goodput performance.*

1 INTRODUCTION

Multihoming among networked machines is a technologically feasible and increasingly economical proposition. A host is multihomed if it can be addressed by multiple IP addresses, as is the case when a host has multiple network interfaces. Though feasibility alone does not determine adoption of an idea, multihoming can be expected to be the rule rather than the exception in the near future. Cheaper network interfaces and cheaper Internet access will motivate content providers to have simultaneous connectivity through multiple ISPs. For added flexibility and fault-tolerance, more home users will have both wired and wireless connections. Furthermore, many wireless devices, especially in FCS (Future Combat Systems) networks, will be connected through multiple access technologies. At an increasingly economical cost, multihoming improves a host's fault tolerance, which is crucial for survivability and persistent on-the-move sessions.

The current transport protocol workhorses, UDP and TCP, are ignorant of multihoming; UDP has no endpoint concept, and TCP allows end host applications to bind to only one network address at each end of a connection. When TCP was designed, network interfaces were expensive, and multihoming was beyond the ken of research. Lower interface costs and a desire for networked applications to be fault tolerant on an end-to-end level have brought multihoming within the purview of the transport layer.

Two recent transport layer protocols, the Stream Control Transmission Protocol (SCTP) [6, 10] and the Datagram Congestion Control Protocol (DCCP) [8] support multihoming at the transport layer. The motivation for multihoming in DCCP is mobility, while SCTP is driven by a broader and more generic application base, which includes fault tolerance and mobility. We use SCTP primarily because our focus is on fault tolerance, but the results and conclusions presented in this paper are applicable to reliable transport protocols that support multihoming.

SCTP, an IETF standards track transport layer protocol, allows binding of one transport layer *association* (SCTP's term for a connection) to multiple IP addresses at each end of the association. SCTP's n to m binding allows a multihomed sender to send data out any of n interfaces to any of a multihomed receiver's m destination addresses. For example, an SCTP multihomed association between hosts A and B in Figure 1 could be bound to both IP addresses at each host: $(\{A_1, A_2\}, \{B_1, B_2\})$. Such an association allows data transmission from host A to host B to be sent to either B_1 or B_2.

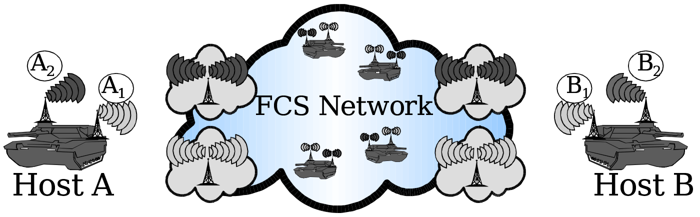

Figure 1: Example multihoming topology

Presently, SCTP uses multihoming for fault tolerance purposes only and not for concurrent multipath transfer [7]. Each endpoint chooses a single peer IP address as the primary destination address, which is used for transmission of new data during normal transmission. If the primary destination address becomes unreachable, the SCTP sender detects the failure, and fails over to an alternate destination address to complete the transfer.

Failure detection time depends on three SCTP tunable parameters: RTO.min (minimum retransmission timeout), RTO.max (maximum retransmission timeout), and Path.Max.Retrans (threshold number of consecutive timeouts that must be exceeded to detect failure). RFC2960 recommends Path.Max.Retrans (PMR) $= 5$, which translates to a failure detection time of at least 63 seconds – unacceptable to many applications. This research

[*] 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.

[†] Supported in part by the University Research Program of Cisco Systems, Inc.

investigates the tradeoff between a more aggressive (i.e., lower) threshold and spurious failovers for the application of bulk file transfer. We surprisingly find that spurious failovers do not degrade overall performance, and sometimes actually improve goodput performance.

Section 2 describes SCTP's failover algorithm. We investigate different PMR settings using ns-2 simulation as described in Section 3. Section 4 presents the tradeoffs between PMR settings and spurious failovers. Section 5 shows how different PMR settings affect goodput. We conclude the paper and discuss future work in Section 6.

2 FAILOVER ALGORITHM

Each SCTP endpoint uses both implicit and explicit probes to dynamically determine the reachability of its peer's IP addresses. Transmitted data serve as implicit probes to the destination receiving new transmissions (generally, the primary destination), while explicit probes, called *heartbeats*, periodically probe idle destinations. Each timeout (for data or heartbeats) on a particular destination increments an error count for that destination. The error count per destination is cleared whenever data or a heartbeat sent to that destination is acked. A destination is marked as failed when its error count *exceeds* PMR. If the primary destination fails, the sender fails over to an alternate destination address. This alternate destination, however, does not become the new primary destination. The primary destination remains unchanged to allow a sender to resume sending new data to the primary destination if and when a future probe to the primary destination is successfully acked.

If a sender fails over to an alternate destination that in turn fails, the sender will failover to yet another alternate destination. If needed, the sender continues to failover to other alternate destinations until all alternate destinations are exhausted. RFC2960 provides implementations the freedom to choose what action to take when the last alternate destination fails (called the *dormant state*) [10]. Our implementation continues changing destinations in a round-robin fashion until a destination responds or the association aborts. SCTP's Association.Max.Retrans parameter sets the threshold of how many consecutive timeouts across all destinations may occur before the association notifies the application and aborts [10].

RFC2960 recommends default settings of: minimum RTO $= 1$ second, maximum RTO $= 60$ seconds, and PMR $= 5$. Thus, in the best case, the first timeout towards failure detection takes 1 second. Then, the exponential back-off procedure doubles the RTO on each timeout towards failure detection. With PMR $= 5$, six consecutive timeouts are needed to detect failure, taking at least $1 + 2 + 4 + 8 + 16 + 32 = 63$ seconds. In the worst case, the first timeout takes the maximum of 60 seconds, and the failure detection time takes $6 * 60 = 360$ seconds. Reducing PMR decreases the failure detection time significantly, but increases the possibility of *spurious failovers*, where a sender mistakenly concludes a failure has occurred.

3 METHODOLOGY

We evaluate different PMR settings using the University of Delaware's SCTP module [5] for the ns-2 network simulator [1]. Figure 2 illustrates the network topology used. The multihomed sender, A, has two paths (labeled *Primary* and *Alternate*) to the multihomed receiver, B. The core links have a 10Mbps bandwidth and a 25ms one-way delay. Each router, R, is attached to a dual-homed node (A or B) via an edge link with 100Mbps bandwidth and 10ms one-way delay. The end-to-end one-way delay is 45ms, which approximates reasonable Internet delays for distances such as coast-to-coast of the continental US, and eastern US to/from western Europe. We believe the results and conclusions in this paper are independent of the actual bandwidth and delay configurations, as long as these configurations are the same on both paths.

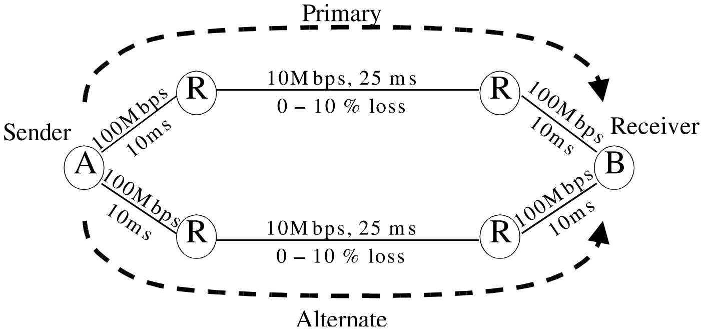

Figure 2: Simulation network topology

We simulate 80MB file transfers with PMR $= \{0, 1, 2, 3, 4, 5\}$. We introduce uniform loss on these paths (0-10% each way) at the core links. Uniform loss is a simple, yet sufficient model to provide insight about the effectiveness of different PMR settings accurately detecting failure. In this study, no link or interface failures are introduced; hence, all failovers are spurious.

The sender uses the following retransmission scheme: (1) send fast retransmissions to the same peer IP address as new data transmissions, (2) send timeout retransmissions to a non-failed alternate peer IP address (if one exists), and (3) employ our Multiple Fast Retransmit algorithm [3]. This scheme differs from the current scheme in RFC2960, but results in [4] show this scheme to perform better. This scheme has been proposed to the IETF for inclusion in the SCTP Implementer's Guide [9].

Three input parameters for each simulation are: the primary path's loss rate, the alternate path's loss rate, and the PMR setting. Each parameter set is simulated with 60 different seeds.

4 SPURIOUS FAILOVERS

Figure 3 plots, for each PMR setting, the fraction of transfers that experience at least a single spurious failover at primary path loss rates 0-10%. Note that the graph aggregates all alternate path loss rates for each particular primary path loss rate. As expected, we found the alternate path loss rate to have little influence on the failure detection process.

Since PMR $= 0$ triggers a failover on a single timeout, this setting provides little robustness against spurious failovers at loss rates greater than 1%. At the other extreme, PMR = 5 experiences nearly no spurious failovers at loss rates less than 9%. As the PMR increases from 0-5, their corresponding curves shift to the right by a loss rate of about 2%. This trend implies a simple linear relationship between the PMR setting and the robustness against spurious failovers. However, the slopes of the curves slowly flatten as the PMR increases, which argues that the robustness increases by more than a constant for each PMR setting.

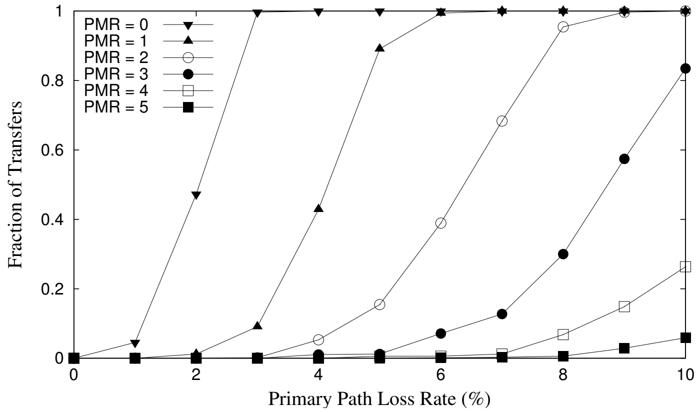

Figure 3: Fraction of transfers with spurious failovers

The frequency of spurious failovers is also important when considering the robustness of various PMR settings. Figure 4 plots the cumulative distribution function (CDF) of the number of spurious failovers for primary path loss rates 2-10%. The CDFs for 1% primary path loss rate are omitted, because PMR = $\{1, 2, 3, 4, 5\}$ experience no spurious failovers, and PMR = 0 experience spurious failovers in only 5% of the transfers. Again, each graph in Figure 4 aggregates all alternate path loss rates for each primary path loss rate.

At only a 2% loss rate, 47% of transfers with PMR = 0 spuriously failover at least once, and 16% of transfers spuriously failover at least twice. More than 99% of transfers with PMR = 0 experience at least one spurious failover at 3% loss and at least ten spurious failovers at 4%.

As expected, PMR = 1 is more robust against spurious failovers than PMR = 0. At 3% loss, only 9% of the transfers spuriously failover. Furthermore, at 4% loss, 43% of the transfers spuriously failover and no transfers experience more than four failovers. When the loss rate is 8%, more than 99% of transfers observe at least ten spurious failovers.

This trend continues for PMR = $\{2, 3, 4, 5\}$ in that they begin to experience a non-negligible number of spurious failovers at about 4%, 6%, 8%, and 9% loss, respectively. In addition, more than 50% of the transfers experience spurious failovers at 7% and 9% loss for PMR = $\{2, 3\}$, respectively. 26% of transfers with PMR = 4 and 6% of transfers with PMR = 5 observe spurious failovers at 10% loss.

5 GOODPUT PERFORMANCE AND ANALYSIS

Figure 5 plots the average 80MB file transfer time for primary path loss rates 2-10%. Each graph has a fixed primary path loss rate, and varies the alternate path loss rate on the x-axis from 1-10%. The PMR setting has little affect on the goodput for primary path loss rates less than 7%. Above 7%, the results show that lower PMR settings begin to improve performance, with PMR = 0 providing the most improvement. Counter to our intuition, lower PMR settings often provide improved performance even when the alternate path loss rate is higher than that of the primary path. For example, lowering the PMR from 5 to 0 improves the performance by 4% when the primary and alternate path loss rates are 8% and 10%, respectively.

To help understand why lower PMR settings (in particular, PMR = 0) surprisingly improve performance, we present in Figure 6 four timeout scenarios for PMR = $\{0, 1\}$. They all begin with TSN 1 being lost in transit to the primary destination and subsequently timing out. For PMR = 0, the sender immediately fails over, retransmits TSN 1 to the alternate destination, and sends a heartbeat to the primary destination. For PMR = 1, the sender retransmits TSN 1 to the alternate destination and sends TSN 2 to the primary destination. We compare the behavior of these two PMR settings by following the details of four (of many) possible scenarios beyond this point.

5.1 Scenario 1

The first packet sent to the primary destination and the first packet sent to the alternate destination following TSN 1's timeout are both delivered successfully.

- **PMR = 0** The failover is cancelled when the heartbeat is acked. Although both TSN 1 and the heartbeat get acked at about the same time, it is a race condition. If the heartbeat gets acked first (as shown in Figure 6), then TSN 2 is sent on the primary and normal data transfer continues from this point. If TSN 1 gets acked first (not shown), then TSNs 2-3 are sent to the alternate destination, TSN 4 is sent to the primary destination when the heartbeat is acked, and normal data transfer continues to the primary destination.
- **PMR = 1** As both TSN 1 and 2 are sent at about the same time, again a race condition occurs. If TSN 1 arrives at the receiver first, the receiver's delayed ack algorithm causes a single cumulative ack (denoted SACK 2) to be generated for both TSN 1 and 2 (as shown in Figure 6). When this ack arrives, TSNs 3-4 are sent to the primary destination and normal data transfer continues to the primary destination. If TSN 2 arrives at the receiver first, the receiver generates two acks (not shown). The first selectively ack TSN 2 with a missing report for TSN 1, and the second cumulatively acks TSN 2. Upon receiving the first, the sender sends TSN 3 to the primary destination and normal data transfer continues to the primary destination.

This scenario presents a marginal difference between the two PMR settings. Similarly, PMR settings greater than PMR = 1 do not achieve more than a marginal improvement either.

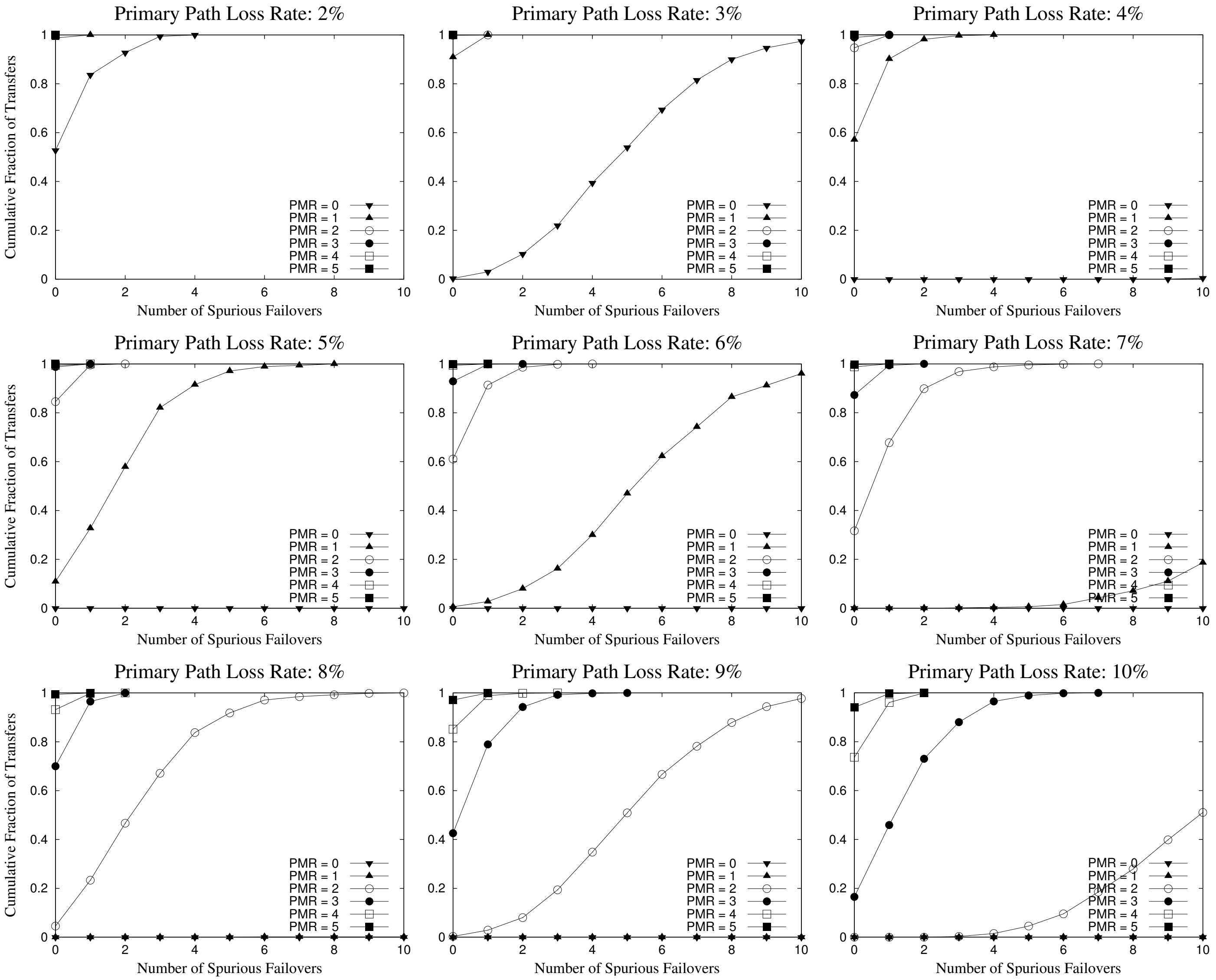

Figure 4: CDF of the number of spurious failovers for primary path loss rates 2-10%

5.2 Scenario 2

The first packet sent to the primary destination following TSN 1's timeout is successfully delivered, and the first packet sent to the alternate destination is lost.

- **PMR = 0** The failover is cancelled when the heartbeat is acked. TSN 2 is sent to the primary destination. When TSN 2 is selectively acked, TSN 3 is then sent to the primary destination. The sender continues sending one packet at a time to the primary destination until TSN 1's retransmission times out. TSN 1 is then re-retransmitted to the primary destination and normal data transfer continues to the primary destination.
- **PMR = 1** When TSN 2 is selectively acked, TSN 3 is sent to the primary destination, and when it is selectively acked, TSN 4 is sent to the primary destination. The sender continues sending one packet at a time to the primary destination until TSN 1's retransmission times out. TSN 1 is then re-retransmitted to the primary destination and normal data transfer continues to the primary destination.

Again, this scenario presents a marginal difference between the two PMR settings, which shows that the alternate path loss rate alone does not affect the performance gap between PMR settings.

5.3 Scenario 3

The first packet sent to the primary destination following TSN 1's timeout is lost, and the first packet sent to the alternate destination is delivered successfully.

- **PMR = 0** When TSN 1 is acked, TSNs 2-3 are sent to the alternate destination, and normal data transfer continues temporarily to the alternate destination. Eventually, the heartbeat times out, and another heartbeat is then sent to the primary destination. Since this timeout is the second consecutive timeout on the primary destination, it will take at least

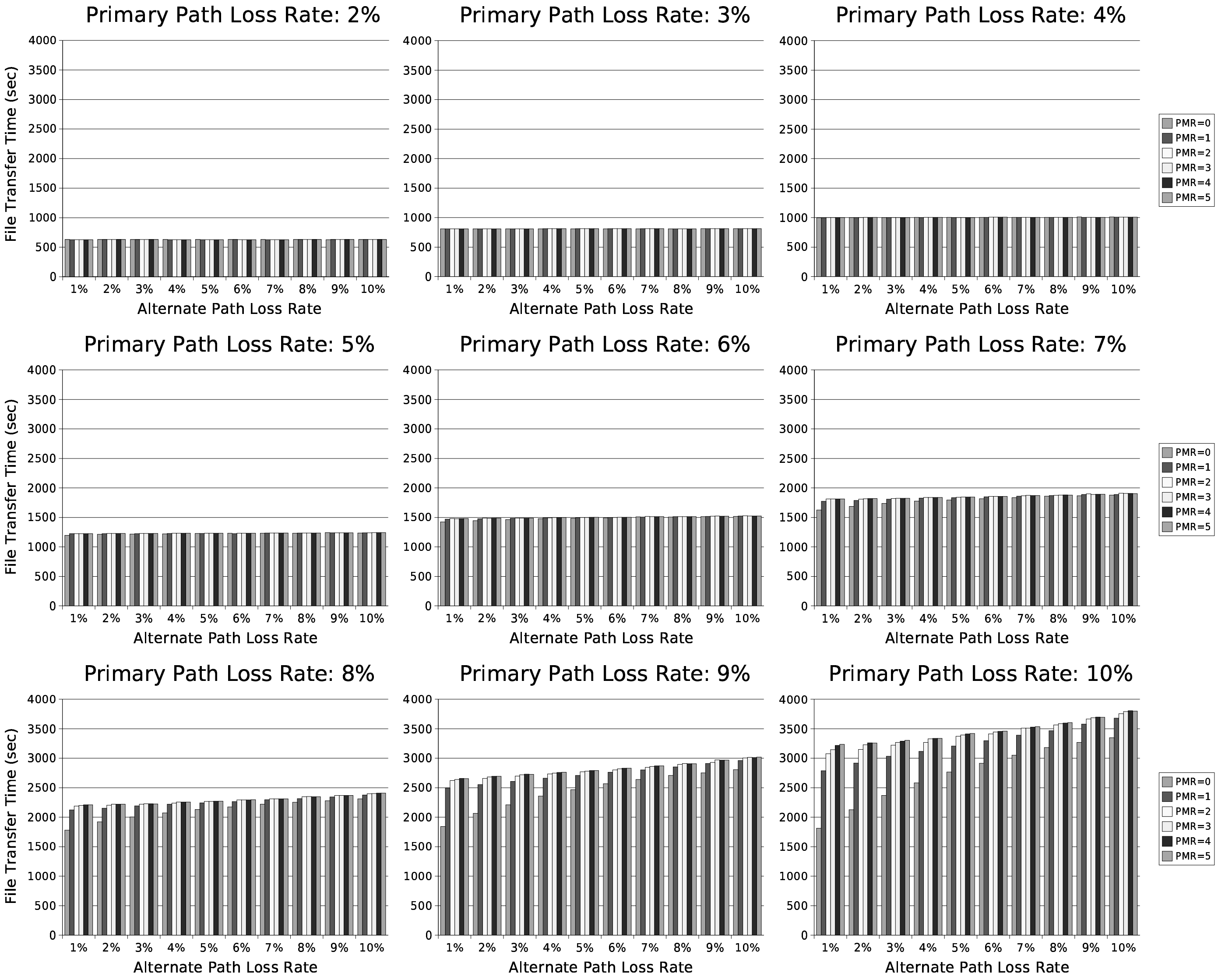

Figure 5: File transfer time for primary path loss rates 2-10%

2 seconds to expire (assuming RTO.Min is 1 second). Once the second heartbeat is successfully acked, the sender cancels the failover, and resumes normal data transmission to the primary destination.

- **PMR = 1** When TSN 1 is acked, the sender is temporarily blocked and does not send any new data. When TSN 2 times out (again, at least 2 seconds to expire), the sender fails over to the alternate destination, retransmits TSN 2 to the alternate destination, and sends a heartbeat to the primary destination. From this point, normal data transfer continues to the alternate destination until the heartbeat is acked and the failover is cancelled. Then the sender resumes normal data transfer to the primary destination.

This scenario presents a more significant difference in performance between the two PMR settings. PMR = 0 smoothly transitions from the primary path to the alternate path and back to the primary path as needed, with little disruption. PMR = 1, however, experiences a disruption in the data transfer. PMR settings greater than PMR = 1 experience the same disruptions, and suffer even further in scenarios where the primary path experiences more than two consecutive timeouts.

5.4 Scenario 4

The first packet sent to the primary destination and the first packet sent to the alternate destination following TSN 1's timeout are both lost.

- **PMR = 0** TSN 1's retransmission times out first, and TSN 1 is re-retransmitted to the primary destination. When TSN 1 is acked, the failover is cancelled and normal data transfer continues to the primary destination from this point. Note that the heartbeat times out later, but does not affect the data transfer.
- **PMR = 1** TSN 1's retransmission times out first, and TSN 1 is re-retransmitted to the primary destination. When TSN

1 is acked, the failover is cancelled, but the sender cannot send any new data until TSN 2 times out. Once TSN 2 times out, the sender retransmits it to the alternate destination, and sends TSN 3 to the primary destination. From this point, normal data transfer continues to the primary destination.

Similar to Scenario 3, this scenario shows that the performance gap between PMR settings widens when the primary path experiences consecutive timeouts. Scenarios 2-4 reveal that the alternate path loss rate alone does not affect the performance gap between PMR settings, but it does influence how much the performance gap is widened during primary path loss events. The lower the alternate path loss rate, the more data that can be transferred during failover events. Thus, lower PMR settings do not degrade overall performance and sometimes improve performance.

6 CONCLUSION AND FUTURE WORK

We investigated the affects of lowering SCTP's failure detection threshold, Path.Max.Retrans (PMR), to less than 5 consecutive timeouts. We found PMR $= \{0, 1, 2, 3, 4, 5\}$ effective at accurately ($\leq 2\%$ error) detecting failure for primary path loss rates up to and including 1%, 2%, 3%, 5%, 7%, and 8%, respectively.

Most unexpectedly, we found that spurious failovers do not degrade the performance of bulk file transfers. We found that the PMR setting has little affect on the goodput for primary path loss rates less than 7%. Significant differences were observed at higher loss rates. At the higher primary path loss rates, lower PMR settings improve overall performance (even when the loss rate is higher on the alternate path). When PMR is aggressively tuned to PMR $= 0$, the goodput is never worse and is often better than that of transfers with PMR $= \{1, 2, 3, 4, 5\}$! Furthermore, in the case of an actual failure, PMR $= 0$ can detect failure and failover in a single timeout.

Given the surprising result that PMR $= 0$ performs best, we believe the PMR settings should be evaluated with different dormant state behaviors, and in network topologies that have different primary and alternate path bandwidth-delay configurations. The degree of multihoming should be increased beyond two per endpoint to ensure that the trends remain the same. Mobile ad-hoc networks that implement reactive routing protocols have route calculation overheads when idle paths are used. We have expanded our model to take these overheads into consideration. Future work will include these overheads in our evaluation.

ACKNOWLEDGEMENTS

The authors acknowledge Ryan Bickhart, Mark Hufe, Janardhan Iyengar, and Sourabh Ladha of the University of Delaware's Protocol Engineering Lab for their valuable comments and suggestions.

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] UC Berkeley, LBL, USC/ISI, and Xerox Parc. ns-2 documentation and software, Version 2.26, 2003. http://www.isi.edu/nsnam/ns.

[2] A. Caro. *End-to-End Fault Tolerance Using Transport Layer Multihoming*. PhD Dissertation, CISC Dept, University of Delaware. (in progress).

[3] A. Caro, P. Amer, J. Iyengar, and R. Stewart. Retransmission Policies with Transport Layer Multihoming. In *ICON 2003*, Sydney, Australia, September 2003.

[4] A. Caro, P. Amer, and R. Stewart. Retransmission Schemes for End-to-End Failover with Transport Layer Multihoming. In *GLOBECOM 2004*, Dallas, TX, November 2004. (to appear).

[5] A. Caro and J. Iyengar. ns-2 SCTP module, Version 3.4, August 2003. http://pel.cis.udel.edu.

[6] A. Caro, J. Iyengar, P. Amer, S. Ladha, G. Heinz, and K. Shah. SCTP: A Proposed Standard for Robust Internet Data Transport. *IEEE Computer*, 36(11):56–63, November 2003.

[7] J. Iyengar, K. Shah, P. Amer, and R. Stewart. Concurrent Multipath Transfer Using SCTP Multihoming. In *SPECTS 2004*, San Jose, California, July 2004.

[8] E. Kholer, M. Handley, and S. Floyd. Datagram Congestion Control Protocol (DCCP). draft-ietf-dccp-spec-07.txt, July 2004. (work in progress).

[9] R. Stewart, I. Arias-Rodriguez, K. Poon, A. Caro, and M. Tuexen. Stream Control Transmission Protocol (SCTP) Implementer's Guide. draft-ietf-tsvwg-sctpimpguide-10.txt, November 2003. (work in progress).

[10] 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, October 2000.

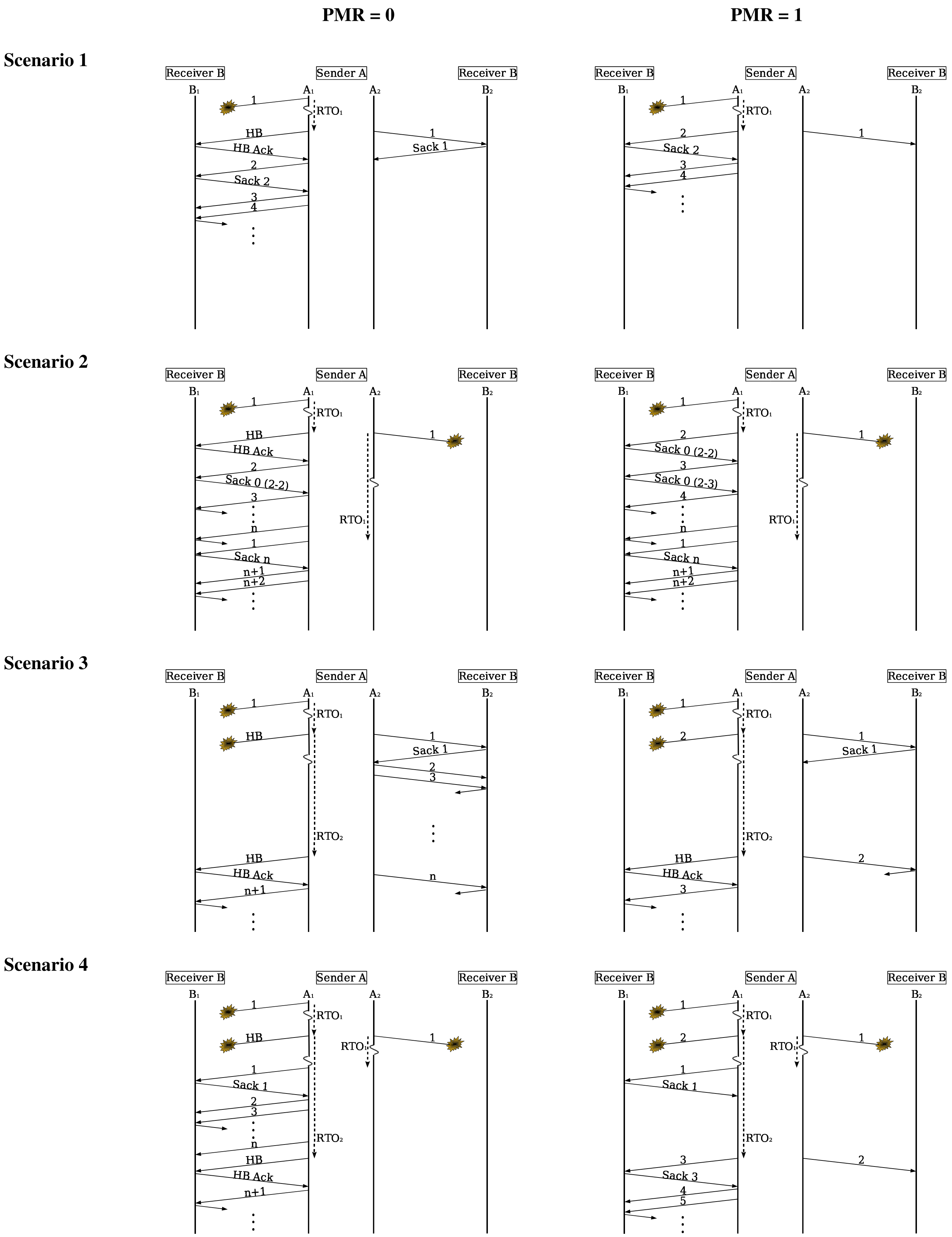

Figure 6: Timeout scenarios