Concurrent Multipath Transfer using SCTP Multihoming: Introducing the Potentially-failed Destination State

Preethi Natarajan[1], Nasif Ekiz[1], Paul D. Amer[1], Janardhan R. Iyengar[2], and Randall Stewart[3]

[1]Protocol Engineering Lab, CIS Dept, University of Delaware
{nataraja, nekiz, amer}@cis.udel.edu
[2]Dept of Math & Computer Science, Connecticut College
iyengar@conncoll.edu
[3]Internet Technologies Division, Cisco Systems
rrs@cisco.com

Abstract. Previously, we identified the failure-induced receive buffer (rbuf) blocking problem in Concurrent Multipath Transfer using SCTP multihoming (CMT), and proposed CMT with a Potentially-failed destination state (CMT-PF) to alleviate rbuf blocking. In this paper, we complete our evaluation of CMT vs. CMT-PF. Using ns-2 simulations we show that CMT-PF performs on par or better than CMT during more aggressive failure detection thresholds than recommended by RFC4960. We also examine whether the modified sender behavior in CMT-PF degrades performance during *non-failure* scenarios. Our evaluations consider: (i) realistic loss model with symmetric and asymmetric path loss, (ii) varying path RTTs. We find that CMT-PF performs as well as CMT during non-failure scenarios, and interestingly, outperforms CMT when the paths experience asymmetric rbuf blocking conditions. We recommend that CMT be replaced by CMT-PF in future CMT implementations and RFCs[1].

Keywords: Stream Control Transmission Protocol (SCTP), Concurrent Multipath Transfer (CMT), Path failure, Receive buffer blocking.

1 Introduction

Unlike TCP and UDP, the Stream Control Transmission Protocol (SCTP) [RFC4960] natively supports multihoming at the transport layer. SCTP multihoming allows binding of a transport layer association (SCTP's term for a connection) to multiple IP addresses at each end of the association. This binding allows transmission of data to different destination addresses of a multihomed receiver. Multiple destination addresses imply the possibility of independent end-to-end paths. Concurrent

[1]Prepared through collaborative participation in the Communication and Networks Consortium sponsored by the US Army Research Lab under Collaborative Tech Alliance Program, Coop Agreement DAAD19-01-2-0011. The US Gov't is authorized to reproduce and distribute reprints for Gov't purposes notwithstanding any copyright notation thereon.
Supported by the University Research Program, Cisco Systems, Inc.

Multipath Transfer (CMT) [5] leverages SCTP's multihoming support, and increases an application's throughput via simultaneous transfer of new data over independent paths between multihomed source and destination hosts.

Reference [4] explores the receive buffer (rbuf) blocking problem in CMT, where transport protocol data unit (TPDU) losses throttle a sender once the transport receiver's buffer is filled with out-of-order data. Even though the congestion window would allow new data transmission, rbuf blocking (i.e., flow control) stalls the sender, causing throughput degradation. To reduce rbuf blocking effects during congestion, [4] proposes different retransmission policies that use heuristics for faster loss recovery, such as, sending retransmissions on the path with the largest congestion window (RTX_CWND policy), or the largest slow-start threshold (RTX_SSTHRESH policy). Both RTX_CWND and RTX_SSTHRESH perform equally well during congestion-induced rbuf blocking, and therefore [4] arbitrarily selected and recommended the RTX_SSTHRESH policy for CMT. In [9], we show how CMT with RTX_SSTHRESH policy suffers from consecutive instances of failure-induced rbuf blocking.

To mitigate rbuf blocking during path failures, we modified CMT's failure detection process to include a new "potentially-failed" (PF) destination state. CMT with a modified set of retransmission policies that consider the PF state is called CMT-PF. CMT-PF is based on the rationale that loss detected by a timeout implies either severe congestion or failure en route. After a single timeout on a path, a sender is unsure, and marks the corresponding destination as PF. A PF destination is not used for data transmission or retransmission. Only heartbeats are sent to the PF destination. If a heartbeat-ack returns, the PF destination returns to active state. Note that CMT-PF retransmission policies are applied only for timeout based loss recoveries. One of CMT's current retransmission policies, such as RTX_SSTHRESH, is applied for TPDU losses detected by RFC4960's threshold number of missing reports (fast retransmits). Details of CMT-PF can be found in [9].

2 Evaluation in Failure Scenarios

We implemented CMT-PF in the University of Delaware's SCTP/CMT ns-2 module [2]. Our previous evaluation [9] considered only the RTX_SSTHRESH variants of CMT and CMT-PF. Our current evaluations include the RTX_CWND variants as well since RTX_CWND appears to be a better policy than RTX_SSTHRESH during failure (more details follow). The four variations considered for evaluations are:

1) CMT-CWND: CMT with RTX_CWND retransmission policy.
2) CMT-SSTHRESH: CMT with RTX_SSTHRESH retransmission policy.
3) CMT-PF-CWND: CMT-PF with RTX_CWND retransmission policy for fast retransmissions.
4) CMT-PF-SSTHRESH: CMT-PF with RTX_SSTHRESH policy for fast retransmissions.

To achieve faster yet robust failure detection, [1] argues for varying Path.Max.Retransmit (PMR) based on a network's loss rate, and suggests PMR=3 for the Internet. Lowering the PMR for CMT flows reduces the number of rbuf blocking

episodes, and thus CMT's throughput degradation during failures. However, a tradeoff exists on deciding the value of PMR – a lower value reduces rbuf blocking but causes spurious failure detection, whereas a higher value increases rbuf blocking but avoids spurious failure detection for a wide range of environments.

We first investigate how CMT-PF improvements are affected by the failure detection threshold. We evaluate CMT vs. CMT-PF for PMR values ranging from 0 – 5. The simulation topology is shown in Figure 1. A multihomed sender, A, has two independent paths to a multihomed receiver, B. The edge links between A or B to the routers represent last hop link characteristics. The end-to-end one-way delay is 45ms on both paths, representing typical U.S. coast-to-coast delays experienced by a significant fraction of today's Internet flows [7].

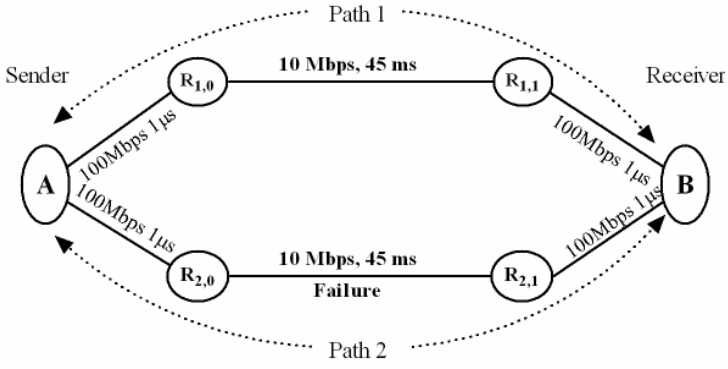


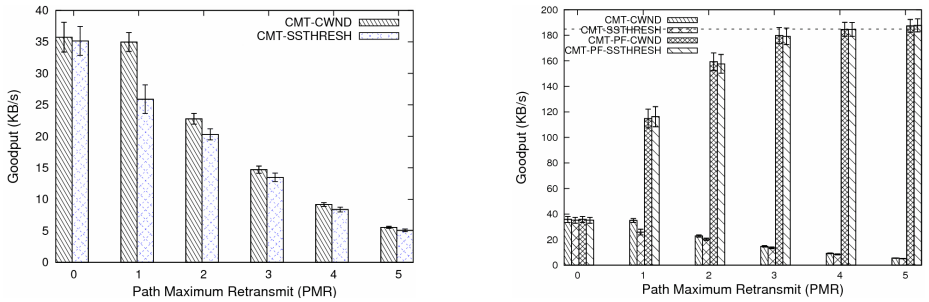
Figure 1: Topology with Bernoulli Loss Model

Each simulation run is a bulk file transfer from A to B. Rbuf=64KB and both paths experience 1% loss rate with Bernoulli loss model. Path 2 fails 10 seconds after the file transfer commences. Sender A detects this path failure after PMR+1 consecutive timeouts. Figure 2 plots the different variations' goodput during failure detection (application data received ÷ failure detection time), with a 5% error margin. Note that the failure detection period decreases with smaller PMR values.

As mentioned in Section 1, [4] arbitrarily recommends RTX_SSTHRESH policy over RTX_CWND during congestion-induced rbuf blocking. Interestingly, the two policies exhibit performance differences during failure-induced rbuf blocking (Figure 2a). Ssthresh is a more historic account of a path's congestion conditions than cwnd. When path 2's loss rate swiftly changes from 1% to 100% (failure), ssthresh converges slower to path 2's current conditions than cwnd. *For CMT, RTX_CWND appears to be a better policy than RTX_SSTHRESH during failure scenarios.*

The dashed line in Figure 2b denotes the maximum attainable goodput of an SCTP file transfer (application data received ÷ transfer time) using path 1 alone. When the failure detection threshold is most aggressive (PMR=0), all senders detect path 2 failure after the first timeout. The 4 variations experience similar rbuf blocking during this failure detection and perform almost equally (Figure 2b). As PMR increases, the number of rbuf blocking instances during failure detection increases, resulting in increasing performance benefits from CMT-PF. As seen in Figure 2b as PMR increases, CMT-PF's goodput increases, whereas CMT's goodput decreases. Starting from PMR=3, CMT-PF's goodput is comparable or equal to the maximum attainable

SCTP goodput. Unlike the two CMT variations, *both CMT-PF-SSTHRESH and CMT-PF-CWND perform equally well during failure, and better than CMT for PMR > 0.*



(a) CMT; CWND vs. SSTHRESH (b) CMT vs. CMT-PF

Figure 2: Goodput during Failure Detection for varying PMR values

3 Evaluation in Non-Failure Scenarios

Section 2 confirms that transitioning a destination to the PF state after timeout caused by path failure alleviates rbuf blocking, and improves CMT-PF performance. However, it is necessary to evaluate whether the PF state transition impairs performance during timeouts caused by *non-failure scenarios* such as congestion. In [9], our congestion experiments assumed a simple loss model with uniformly distributed loss rates. Here, we consider the more realistic topology and loss model shown in Figure 3.

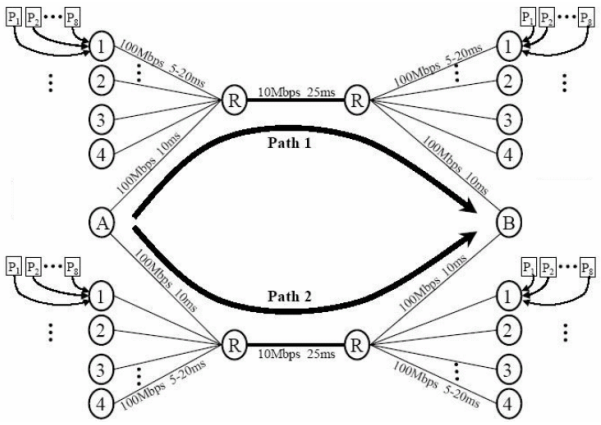


Figure 3: Bursty Loss Created by Cross-traffic

In this dual-dumbbell topology, each router, R, is attached to five edge nodes. Dual-homed edge nodes A and B are the transport sender and receiver, respectively. The other edge nodes are single-homed, and introduce cross-traffic that instigates

bursty periods of congestion and bursty congestion losses at the routers. Their last-hop propagation delays are randomly chosen from a uniform distribution between 5-20 ms, resulting in end-to-end one-way propagation delays ranging ~35-65ms [7]. All links (both edge and core) have a buffer size twice the link's bandwidth-delay product, which is a reasonable setting in practice. Each single-homed edge node has eight traffic generators, introducing cross-traffic with 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 [3, 8]. The result is a data transfer between A to B, over a network with self-similar cross-traffic, which resembles the observed nature of traffic on data networks [10]. We simulate a 32MB file transfer between sender A and receiver B, under different loss conditions. Rbuf=64KB, PMR=5, and loss rates are controlled by varying the cross-traffic load.

3.1 Symmetric Loss Conditions

The aggregate cross-traffic load on both paths are similar and vary from 40%-100% of the core link's bandwidth. Figure 4 plots the average goodput (file size ÷ transfer time), of the 4 variations with 5% error margin. The 4 variations exhibit similar performance during low loss rates (i.e., minimal cross-traffic), since most of the TPDU losses are recovered via fast retransmits as opposed to timeouts. As cross-traffic and hence loss rate increases, the number of timeouts on each path increases. Under such conditions, the probability and time duration that both paths are *simultaneously* marked "potentially-failed" increases in CMT-PF. To ensure that CMT-PF performs well even if all destinations get marked PF, CMT-PF transitions the destination with the smallest number of consecutive timeouts back to the active state, allowing data to be sent to that destination [9]. This modification guarantees that CMT-PF performs on par with CMT even when both paths experience high loss rates (Figure 4).

Under symmetric loss conditions, we study how a path's RTT affects the throughput differences between CMT and CMT-PF. Note that any difference between CMT and CMT-PF transpires only after a timeout on a path. Let us assume that a path experiences a timeout event, and the next TPDU loss on the path takes place after n RTTs. After the timeout, CMT slow starts on the path, and the number of TPDUs transmitted on the path at the end of n RTTs = $1 + 2 + 4 \ldots + 2n = (2(n+1) - 1)$. CMT-PF uses the first RTT for a heartbeat transmission, and slow starts with initial cwnd=2 after receiving the heartbeat-ack. In CMT-PF, the number of TPDUs transmitted by end of n RTTs on the path = $0 + 2 + 4 \ldots + 2n = (2(n+1) - 2)$. Thus, in n RTTs, CMT transmits 1 TPDU more than CMT-PF, and the 1 TPDU difference is unaffected by the path's RTT. Therefore, when paths experience symmetric RTTs (a.k.a. symmetric RTT conditions), we expect the performance ratio between CMT and CMT-PF to remain unaffected by the RTT value.

We now consider a more interesting scenario when the independent end-to-end paths experience symmetric loss rates, but *asymmetric* RTT conditions. That is, path 1's RTT=x sec, and path 2's RTT=y sec ($x \neq y$). How do x and y impact CMT vs. CMT-PF performance? More importantly, does CMT-PF perform worse when the paths have asymmetric RTTs? We performed the following Bernoulli loss model

experiment to gain insight (Bernoulli loss model simulations take much less time than cross-traffic ones, and both loss models resulted in similar trends between CMT and CMT-PF). We used the topology shown in Figure 1 to transfer an 8MB file from sender A to receiver B. Path 1's one-way propagation delay was fixed at 45ms while path 2's one-way delay varied as follows: 45ms, 90ms, 180ms, 360ms, and 450ms. Both paths experience identical loss rates ranging from 1%-10%.

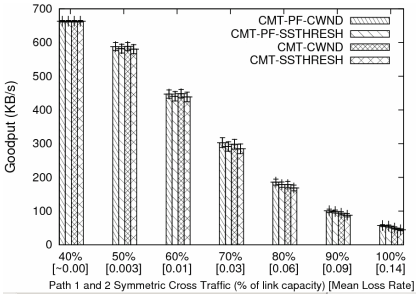


Figure 4: Goodput during Symmetric Loss Conditions

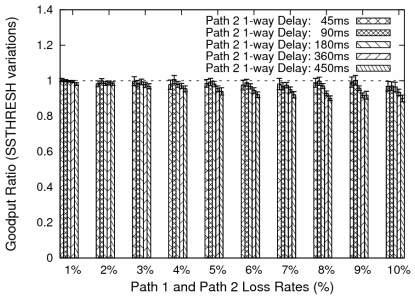


Figure 5: Goodput Ratio during Asymmetric Path RTTs

Figure 5 plots the ratio of CMT-SSTHRESH's goodput over CMT-PF-SSTHRESH's (relative performance difference) with 5% error margin. As expected, both CMT and CMT-PF perform equally well during symmetric RTT conditions. As the asymmetry in paths' RTTs increases, an interesting dynamic dominates and CMT-PF performs slightly better than CMT (goodput ratios < 1).

Note that rbuf blocking depends on the frequency of loss events (loss rate) and duration of loss recovery. Loss recovery duration increases with a path's RTT and RTO. As loss rate increases, the probability that a sender experiences consecutive timeout events on the path increases. After the first timeout, CMT-PF transitions the path to PF, and avoids data transmission on the path (as long as another active path exists) until a heartbeat-ack confirms the path as active. But, a CMT sender suffers back-to-back timeouts on *data* sent on the path, with exponential backoff of timeout recovery period. Consecutive timeouts increase path 2's RTO more at higher RTTs, and results in longer periods of rbuf blocking in CMT (shown in [9]). Therefore, as path 2's RTT increases, CMT's goodput degrades *more* than CMT-PF's, and the goodput ratio decreases (Figure 5). Similar trends are observed between CMT-CWND and CMT-PF-CWND (not shown due to space constraints) [9]. In summary, *during symmetric loss conditions, CMT and CMT-PF perform equally well when paths experience symmetric RTT conditions. As the RTT asymmetry increases, CMT-PF demonstrates a slight advantage at high loss rates.*

3.2 Asymmetric Loss Conditions

In our asymmetry experiments, paths 1 and 2 experience different cross-traffic loads. The aggregate cross-traffic load on path 1 is set to 70% of the core link bandwidth, while on path 2 the load varies from 70%-100% of the core link bandwidth. Figure 6 plots the average goodput of the 4 variations with 5% error margin. As discussed in

the previous sub-section, as path 2's cross-traffic load increases, the probability that a sender experiences back-to-back timeouts on path 2 increases. CMT suffers a higher number of consecutive timeouts on *data* (Table 1) resulting in extended rbuf blocking periods than CMT-PF. Therefore, as path 2's cross-traffic load increases, CMT-PF performs better than CMT (Figure 6). This result holds true for both RTX_CWND and RTX_SSTHRESH variants.

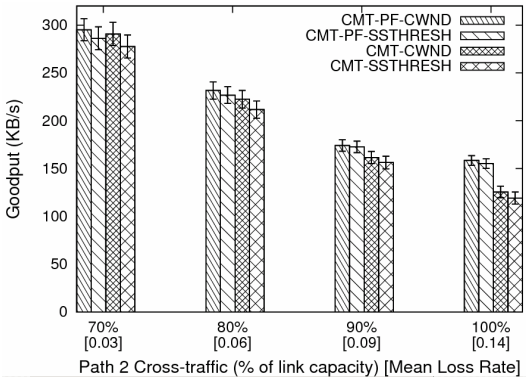


Figure 6: CMT vs. CMT-PF Goodput during Asymmetric Loss Conditions (Path 1 Cross-traffic Load = 70%)

Table 1: Mean Consecutive Data Timeouts (Path 2)

Variant	Path 2 Cross-traffic	# of Consecutive			
		2	3	4	5
CMT-CWND	90	3.89	0.49	0.20	0.02
CMT-PF-CWND		0.22	0.07	0	0
CMT-CWND	100	8.80	1.36	0.24	0.16
CMT-PF-CWND		0.22	0.11	0	0
CMT-SSTHRESH	90	5.44	0.56	0.02	0.02
CMT-PF-SSTHRESH		0.29	0.02	0	0
CMT-SSTHRESH	100	9.44	1.80	0.51	0.09
CMT-PF-SSTHRESH		0.36	0.07	0	0

Table 2: Mean Number of Transmissions

Variant	Path 2 Cross-traffic %	Aggregate Transmissions		
		Path 1	Path 2	Path1/Path2
CMT-CWND	90	15,186	8,822	**1.7**
CMT-PF-CWND		16,115	7,944	**2.0**
CMT-CWND	100	16,842	7,256	**2.3**
CMT-PF-CWND		18,843	5,233	**3.6**
CMT-SSTHRESH	90	15,096	8,902	**1.7**
CMT-PF-SSTHRESH		16,113	7,940	**2.0**
CMT-SSTHRESH	100	16,670	7,421	**2.2**
CMT-PF-SSTHRESH		18,983	5,097	**3.7**

In CMT, RTX_CWND and RTX_SSTHRESH are *retransmission* policies, and are not applied during new data transmissions. In CMT-PF, a path is marked PF after a timeout, and as long as active path(s) exist, CMT-PF avoids retransmissions on the PF path. Once the retransmissions are all sent, CMT-PF's data transmission strategy is applied to new data, and CMT-PF *avoids new data transmissions* on the PF path. As shown in Table 2, when compared to CMT, CMT-PF reduces the number of (re)transmissions on the higher loss rate path 2 and (re)transmits more on the lower loss rate path 1. This transmission difference (ratio of transmissions on path 1 over path 2) between CMT-PF and CMT increases as the paths become more asymmetric in their loss conditions. In summary, *CMT-PF does not perform worse than CMT during asymmetric path loss conditions. In fact, CMT-PF is a better transmission strategy than CMT, and performs better as the asymmetry in path loss increases.*

4 Conclusion & Future Work

Using simulation, we demonstrated that (re)transmission policies using CMT with a "potentially-failed" destination state (CMT-PF) outperform CMT during failures – even under aggressive failure detection thresholds. Investigations during symmetric loss conditions revealed that CMT-PF performs as well as CMT during symmetric path RTTs, and slightly better when the paths experience asymmetric RTT conditions. The simulation results conclude that CMT-PF (i) reduces rbuf blocking during failure scenarios, and (ii) performs on par or slightly better than CMT during non-failure scenarios. In light of these findings, we recommend CMT be replaced by CMT-PF in SCTP implementations and RFCs. We have implemented CMT-PF in FreeBSD SCTP/CMT stack, and are performing emulation experiments comparing CMT vs. CMT-PF. As future work, we will evaluate CMT vs. CMT-PF when end-to-end paths share a bottleneck and therefore are no longer independent.

5 Disclaimer

6 References

[1] A. Caro, "End-to-End Fault Tolerance using Transport Layer Multihoming," PhD Dissertation, CIS Dept, U of Delaware, 08/05.

[2] A. Caro, J. Iyengar, "ns-2 SCTP Module," Version 3.6, 06/06. pel.cis.udel.edu.

[3] CAIDA: Packet Sizes and Sequencing, 03/98. www.caida.org.

[4] J. Iyengar, P. Amer, R. Stewart. "Performance Implications of Receive Buffer Blocking in Concurrent Multipath Transfer," Computer Communications, 2/07, 30(4), pp 818-829.

[5] J. Iyengar, P. Amer, R. Stewart, "Concurrent Multipath Transfer using SCTP Multihoming over Independent End-to-end Paths," IEEE/ACM Trans on Networking, 10/06, 14(5), pp 951-964.

[6] P. Natarajan, J. Iyengar, P. Amer, R. Stewart, "CMT using Transport Layer Multihoming: Performance under Network Failures," MILCOM 2006, Washington D. C., 10/06.

[7] S. Shakkottai, R. Srikant, A. Broido, K. Claffy, "The RTT distribution of TCP Flows in the Internet and its Impact on TCP-based flow control," TR, CAIDA, 02/04.

[8] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, C. Diot. "Packet-level Traffic Measurements from the Sprint IP Backbone," IEEE Network, 11/03, 17(6), pp 6-16.

[9] P. Natarajan, N. Ekiz, P. Amer, R. Stewart, "CMT using SCTP Multihoming: Transmission Policies using a Potentially-failed Destination State," TR 2007/338, CIS Dept, U of Delaware, 02/07.

[10] W. Leland, M. Taqqu, W. Willinger, D. Wilson, "On the Self-similar Nature of Ethernet Traffic," ACM SIGCOMM, San Francisco, 09/93.