

Receive Buffer Blocking In Concurrent Multipath Transfer

Janardhan R. Iyengar **Paul D. Amer** **Randall Stewart**
 Protocol Engineering Lab, CIS Department Internet Technologies Division
 University of Delaware Cisco Systems
 {iyengar, amer}@cis.udel.edu rrs@cisco.com

Abstract—Previously, we studied the performance of **Concurrent Multipath Transfer using SCTP Multihoming (CMT)** under the assumption of an infinite receive buffer (rbuf). Here, we study CMT performance when a sender is constrained by the rbuf. We demonstrate using simulation that if two paths are used for CMT, the lower quality (i.e., higher loss rate) path degrades overall throughput of an rbuf-constrained CMT association by blocking the rbuf. We demonstrate that a wise retransmission policy can alleviate some of the throughput degradation by reducing the rbuf blocking problem. We present and discuss CMT performance using several retransmission policies and constrained rbuf values of 16KB, 32KB, 64KB, 128KB, and 256KB. While rbuf blocking cannot be eliminated, it can be reduced by choice of retransmission policy - a facility available to only the transport layer.

I. INTRODUCTION

A host is multihomed if it can be addressed by multiple IP addresses, as is the case when the host has multiple network interfaces. Multihoming is increasingly economically feasible and can be expected to be the rule rather than the exception in the near future, particularly when fault tolerance is crucial. Multihomed nodes may be simultaneously connected through multiple access technologies, and even multiple end-to-end paths to increase resilience to path failure. For instance, a mobile user could have simultaneous Internet connectivity via a wireless local area network using 802.11b and a wireless wide area network using GPRS.

Previous work proposed using *Concurrent Multipath Transfer (CMT)* between multihomed source and destination hosts to increase an application’s throughput [1], [2]. CMT is the concurrent transfer of new data from a source to a destination host via two or more end-to-end paths. The current transport protocol workhorses, TCP and UDP, do not support multihoming; TCP allows binding to only one network address at each end of a connection. At the time TCP was designed,

network interfaces were expensive components, and multihoming was beyond the ken of research. Changing economics and increasing emphasis on end-to-end fault tolerance have brought multihoming within the purview of the transport layer.

While concurrency can be arranged at other layers, the transport layer has the best knowledge to estimate end-to-end paths’ characteristics. Implementing multipath transfer at the application layer increases redundancy and room for error by requiring a separate implementation by each application programmer. We research CMT at the transport layer using the Stream Control Transmission Protocol (SCTP) [3], [4]. SCTP is an IETF standards-track protocol that natively supports multihoming at the transport layer. SCTP multihoming allows binding of one transport layer *association* (SCTP’s term for a connection) to multiple IP addresses at each end of the association. This binding allows a sender to transmit data to a multihomed receiver through different destination addresses. *Concurrent* transfer of *new* data to multiple destination addresses is currently not allowed in SCTP due primarily to insufficient research. Our research attempts to fill that need.

Previous CMT research developed three algorithms for SCTP resulting in CMT_{scd} - a protocol that uses SCTP’s multihoming feature for *correctly* transferring data between multihomed end hosts using multiple separate end-to-end paths [1]. Since CMT_{scd} ¹ allows concurrent transmission of data to multiple destinations, a sender can send retransmissions to one of several destinations that are receiving new transmissions. Previously, we proposed five retransmission policies, and protocol modifications to SCTP to allow for correct functioning of these retransmission policies [2]. But in [2], we operated under the strong and limiting assumptions that the receive buffer (rbuf) was infinite, and that the bottleneck queues on the end-to-end paths used in CMT were independent of each other.

In this paper, we drop the first assumption and investigate the effects of a bounded rbuf. That is, we analyze CMT performance using different retransmission policies in the presence of a finite rbuf. We continue to operate under the strong assumption that the bottleneck queues on the end-to-

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 University Research Program of Cisco Systems, Inc.

¹Henceforth, we will refer to “ CMT_{scd} ” as simply “CMT”

end paths used in CMT are independent of each other. Future work will relax this constraint.

Section II describes the simulation topology for our investigation, and reviews relevant concepts and terminology. Section III describes, by example, the rbuf blocking problem. Section IV reviews five retransmission policies for CMT. Section V presents our simulation results and a detailed analysis of the different retransmission policies with different rbuf sizes. Section VI concludes the paper.

II. PRELIMINARIES

For our simulations, we use the University of Delaware’s Sctp module for the ns simulator [5], [6], modified to incorporate CMT and the different retransmission policies. The topology is shown in Figure 1. The edge links represent the last hop, and the core links represent end-to-end conditions on the Internet. The end-to-end delays are 45ms on both paths, representing typical US coast-to-coast delays experienced by a significant fraction of the flows on the Internet [7]. The loss rate on Path 1 is maintained at 1%, and on Path 2 is varied from 1 to 10%. A loss rate of 1% means a forward path loss rate of 1%, and a reverse path loss rate of 1%.

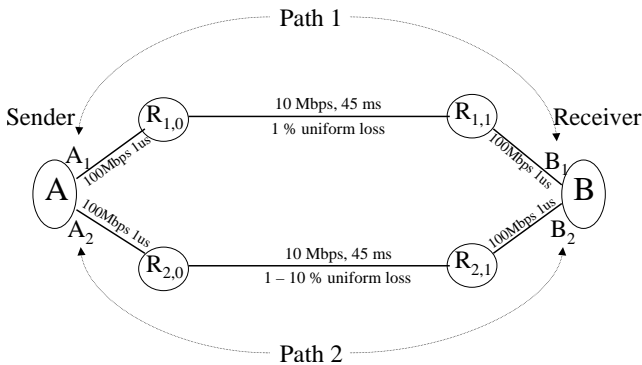


Fig. 1. Simulation topology

This simulation topology does not account for effects seen in the Internet such as network-induced reordering, delay spikes, etc.; these effects are beyond the scope of this study. Our simulation evaluation provides insight into the impact that a constrained rbuf poses to CMT, the fundamental differences between the retransmission policies, and their performance in a constrained rbuf environment. We chose a simple topology to avoid influence of other effects, and to focus on performance differences which we believe should hold true in a real environment as well².

We now review relevant concepts and terminology. A transport layer receiver maintains rbuf space for incoming data for two reasons: (i) to handle out-of-order data, and (ii) to

²The simulation topology is clearly simplistic. More realistic complex topologies involving variable cross-traffic were attempted, but required too much time to simulate. We believe that the *relative* performance of the evaluated policies will be the same for our simple topology as for a more complex one.

receive data at a rate higher than that of the receiving application’s consumption. In SCTP (and TCP), a receiver advertizes currently available rbuf space through window advertisements (normally accompanying acks) to a data sender. We refer to this value as the *advertized receive window (adv-rwnd)*. A sender computes a *peer-rwnd* to deduce how much *more* data can be buffered at the receiver. Beside the latest adv-rwnd received, the peer-rwnd takes into account data that has been sent but not yet acked by the receiver.

An SCTP receiver maintains a single rbuf per association. An SCTP sender, consequently, maintains a single peer-rwnd per association. Note that sender-side estimates such as congestion window (cwnd), slow start threshold (sssthresh) and roundtrip time (RTT) are maintained per destination - they represent the state of different network paths from a sender to each destination address. A sender has no reason to maintain separate rbufs or peer-rwnds per path since a receiver can consume data only in sequence, irrespective of the destination address they are sent to³. An SCTP sender’s sending rate is bound by both the peer-rwnd and the pertinent destination’s cwnd, i.e., $\min(\text{peer-rwnd}, \text{cwnd})$.

We define a *sub-association flow* as the set of transport protocol data units (PDUs) within an SCTP association that have the same destination address. In Figure 1, an SCTP association from the sender to the receiver spanning the two paths will have two sub-association flows - one consisting of PDUs with destination B_1 , and the other with destination B_2 .

A reference to “cwnd for destination X” means the cwnd maintained at the sender for destination X, and “timeout on destination X” refers to the expiration of a retransmission timer maintained for destination X at the sender. Since bottleneck queues on the end-to-end paths are assumed independent, each destination uniquely maps to an independent path. For instance in Figure 1, “cwnd for destination B_1 ” may be used interchangeably with “cwnd for path 1” (where path 1 ends at destination B_1).

CMT schedules new data to different destinations as bandwidth becomes available on corresponding paths, i.e., as corresponding cwnds allow. When cwnd space is available simultaneously for two or more destinations, data is sent to these different destinations arbitrarily. Assuming that a CMT sender will likely not know path properties *a priori*, our transmission policy is an intuitive one.

III. PROBLEM DESCRIPTION: RECEIVE BUFFER BLOCKING

A CMT receiver maintains a single rbuf which is shared across the sub-association flows in an association. With CMT, this buffer sharing can degrade throughput. To elaborate this point, we use an excerpt from a simulation of a CMT association (see Figure 2) using the topology shown in Figure 1. In this example, the rbuf is 16KB, Path 1 (A_1 to B_1) has a loss rate of 1%, and Path 2 (A_2 to B_2) has a loss rate of 10%.

³While SCTP supports *unordered* data delivery and *multistreaming* in an association [4], here we focus on ordered data delivery over a single stream.

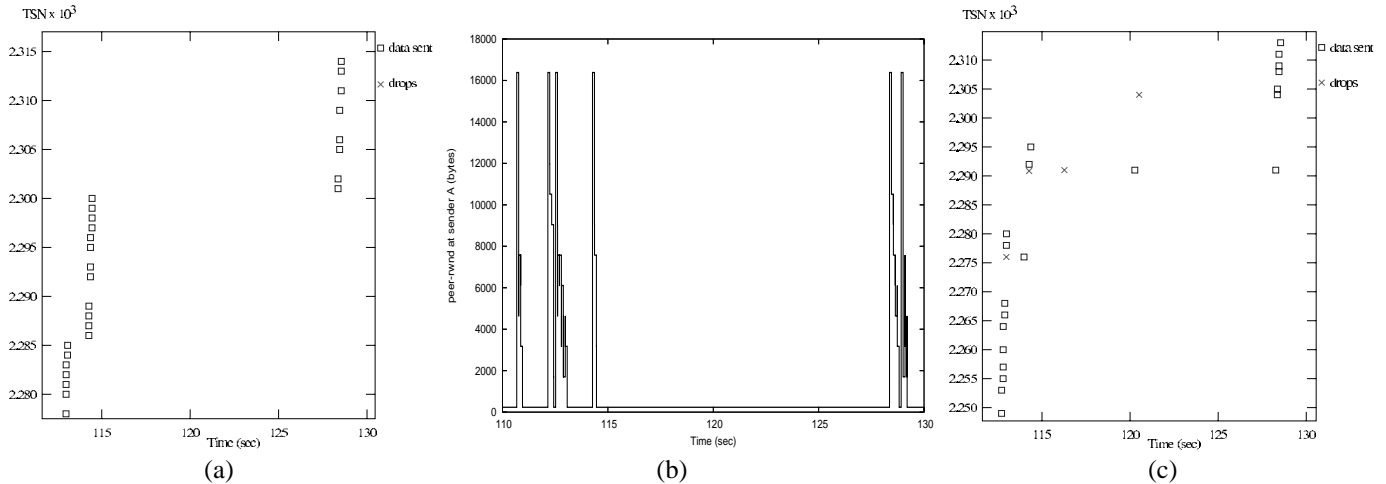


Fig. 2. Instantiation of rbuf blocking: (a) Progression of data sent to destination B₁ over Path 1 (loss rate 1%) over a select interval; (b) peer-rwnd value maintained at sender (endpoint A) over same interval; (c) Progression of data sent to destination B₂ over Path 2 (loss rate 10%) over same interval

Retransmissions are sent to the same destination as the original transmission. We call this retransmission policy *RTX-SAME* (we will revisit this retransmission policy decision later).

Figures 2(a) and (c) show Transmission Sequence Number (TSN) progression⁴ over Path 1 and Path 2, respectively, and Figure 2(b) shows peer-rwnd evolution at the sender (endpoint A) during an excerpted time interval. Figure 2(a) shows that data transmission over the lower loss rate path stops abruptly around 114.5 seconds and resumes around 128 seconds. This 13.5 second cessation can be explained with the help of Figure 2(b). At 114.5 seconds, the peer-rwnd at the sender abruptly reduces from 16384 bytes to 236 bytes, constraining the sender from transmitting any new data. The cause for this abrupt rbuf reduction is explained as follows.

During the same time interval from 114.5 seconds to 128 seconds, Figure 2(c) shows that the lower quality (i.e., higher loss rate) path undergoes congestion, and recovers from losses through repeated retransmission timeouts - the longest recovery time being 8 seconds for TSN 2304. During this entire period of 13.5 seconds when loss recovery repeatedly occurs on Path 2, the receiver waits for retransmissions to come through, and is unable to deliver subsequent TSNs to the application (some of which were sent over Path 1). These subsequent TSNs are held in the transport layer rbuf until the retransmissions are received, thus blocking the rbuf and the peer-rwnd. Path 2 thus causes blocking of the rbuf, preventing data from being sent on either path and reducing overall throughput.

This example demonstrates how a shared rbuf results in a sub-association flow on a higher quality (i.e., lower loss rate) path getting lower throughput than expected. We note that the exact numbers used in this example do not hold special relevance. This example presents a phenomenon which occurs,

⁴TSNs in SCTP serve the same purpose as sequence numbers do in TCP.

in lesser or greater degree, throughout a CMT association.

We argue that rbuf blocking is not specific to the transport layer; it applies to multipath transfer at other layers as well. rbuf blocking cannot be eliminated or reduced by moving CMT's functionality to a different layer. Specifically, if a single logical flow is distributed across multiple end-to-end paths by the application layer, and the application layer receiver (the final destination) has finite buffer space, then rbuf blocking will occur.

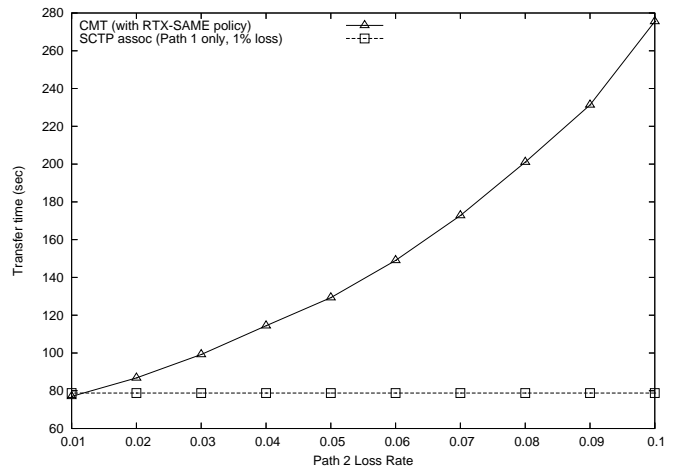


Fig. 3. rbuf blocking in CMT causes throughput degradation

Figure 3 shows two curves - time taken to transfer an 8MB file using (i) CMT (with RTX-SAME retransmission policy) with a 16KB rbuf, and (ii) a single SCTP association which uses *only* the better path (Path 1 with loss rate 1%). Intuitively, using two paths should provide higher overall throughput than using one path. However, Figure 3 demonstrates that using two paths performs worse than using only the better path if a finite

rbuf is shared across the paths⁵. This performance difference is due to rbuf blocking that occurs in CMT - an rbuf of 16KB does not constrain a single SCTP association (which uses one lower loss rate path) as much as it constrains CMT (which uses two paths with different loss rates).

After analyzing several traffic flows, we observe that chances of rbuf blocking are higher during periods of timeout recovery. Further, a larger timeout recovery period due to back-to-back timeouts with exponential backoff results in an even higher probability that a finite rbuf blocks a sender. We therefore hypothesize that reducing (i) the number of timeouts, and/or (ii) the number of back-to-back timeout retransmissions will alleviate the rbuf blocking problem. Consequently, we hypothesize that making an intelligent decision about which destination a retransmission should be sent to will help reduce rbuf blocking. As we mentioned earlier, retransmissions in the above example are sent to the same destination as the original transmission. We used this retransmission policy initially due to its simplicity. In the following section, we present several alternative “smarter” policies.

IV. CMT RETRANSMISSION POLICIES

Multiple paths present an SCTP sender with a choice where to send a retransmission of a lost transmission. But this choice is not well-informed since SCTP currently restricts sending new data [4], which can act as probes for information (such as available bandwidth, loss rate and RTT), to only one primary destination. Consequently, an SCTP sender has minimal information about alternate paths to a receiver.

On the other hand, since new data is regularly being sent to all destinations concurrently, a CMT sender maintains more current and more accurate information (e.g., RTT estimate) about all paths to a receiver. This information allows a CMT sender to make a more informed decision where to send a retransmission. We now investigate how CMT should make this decision in a realistic lossy environment.

We review five retransmission policies for CMT [2]. For four policies, a retransmission may be sent to a destination other than the one used for the original transmission.

- **RTX-SAME** - Once a new data chunk is scheduled and sent to a destination, all retransmissions of the chunk thereafter are sent to the same destination (until the destination is deemed *inactive* due to failure [4]).
- **RTX-ASAP** - A retransmission of a data chunk is sent to any destination for which the sender has cwnd space available at the time the retransmission needs to be sent. If the sender has available cwnd space for multiple destinations, one is chosen randomly.
- **RTX-CWND** - A retransmission of a data chunk is sent to the destination for which the sender has the largest cwnd. A tie is broken randomly.

⁵Presumably an SCTP sender does not have apriori knowledge about the better path and hence cannot always achieve best performance. We discuss *expected* SCTP performance in [8].

- **RTX-SSTHRESH** - A retransmission of a data chunk is sent to the destination for which the sender has the largest ssthresh. A tie is broken randomly.
- **RTX-LOSSRATE** - A retransmission of a data chunk is sent to the destination with the lowest loss rate path. If multiple destinations have the same loss rate, one is selected randomly.

Of the policies, RTX-SAME is simplest. Since an application that stripes data across multiple transport layer (SCTP or TCP) associations will not be able to move outstanding data from one association to another, such an application will use the RTX-SAME policy at the transport layer. RTX-SAME thus represents the performance of an data striping application. RTX-ASAP is a “hot-potato” retransmission policy - the goal is to retransmit as soon as possible (without regard to loss rate) so that a receiver does not have to wait longer for the lost TSN. RTX-CWND and RTX-SSTHRESH attempt to move retransmissions onto the path with estimated lowest loss rate. Since ssthresh is a slower moving variable than cwnd, the values of ssthresh may better reflect the conditions of the respective paths⁶. RTX-LOSSRATE uses information about loss rate provided by an “oracle” - information that RTX-CWND and RTX-SSTHRESH estimate. This policy represents a “hypothetically” ideal case; hypothetical since in practice, a sender typically does not know apriori which path has the lowest loss rate; ideal since the path with the lowest loss rate has highest chance of having a PDU delivered.

Retransmission policies that take loss rate into account (RTX-CWND, RTX-SSTHRESH, and RTX-LOSSRATE) attempt to reduce the chances of a retransmission getting dropped. Therefore, we initially hypothesized that these policies would best alleviate rbuf blocking by reducing the number of timeouts and the number of back-to-back timeouts.

We do not propose different policies for scheduling new transmissions - we assume that a sender does not know path properties apriori and can therefore only react to network events such as congestion losses (see Section II for our transmission policy). This assumption holds true particularly in the Internet where the path properties are changing.

V. EVALUATION OF CMT RETRANSMISSION POLICIES

We now evaluate the five retransmission policies for CMT operating under a constrained rbuf. Default rbuf values in commonly used operating systems today vary from 16KB to 64KB and beyond. We believe that today, when having $\frac{1}{2}$ GB of memory on a desktop computer is common, having an rbuf of at least 64KB should be reasonable. We first study and analyze performance of the different policies with an rbuf of 64KB in Section V-A. This section provides insight into the causes of the performance differences between the retransmission policies. We then summarize performance of the different policies under more and less constraining rbufs

⁶We assume that the reader is familiar with SCTP and TCP congestion control, the variables cwnd and ssthresh, and their dynamics [4], [9].

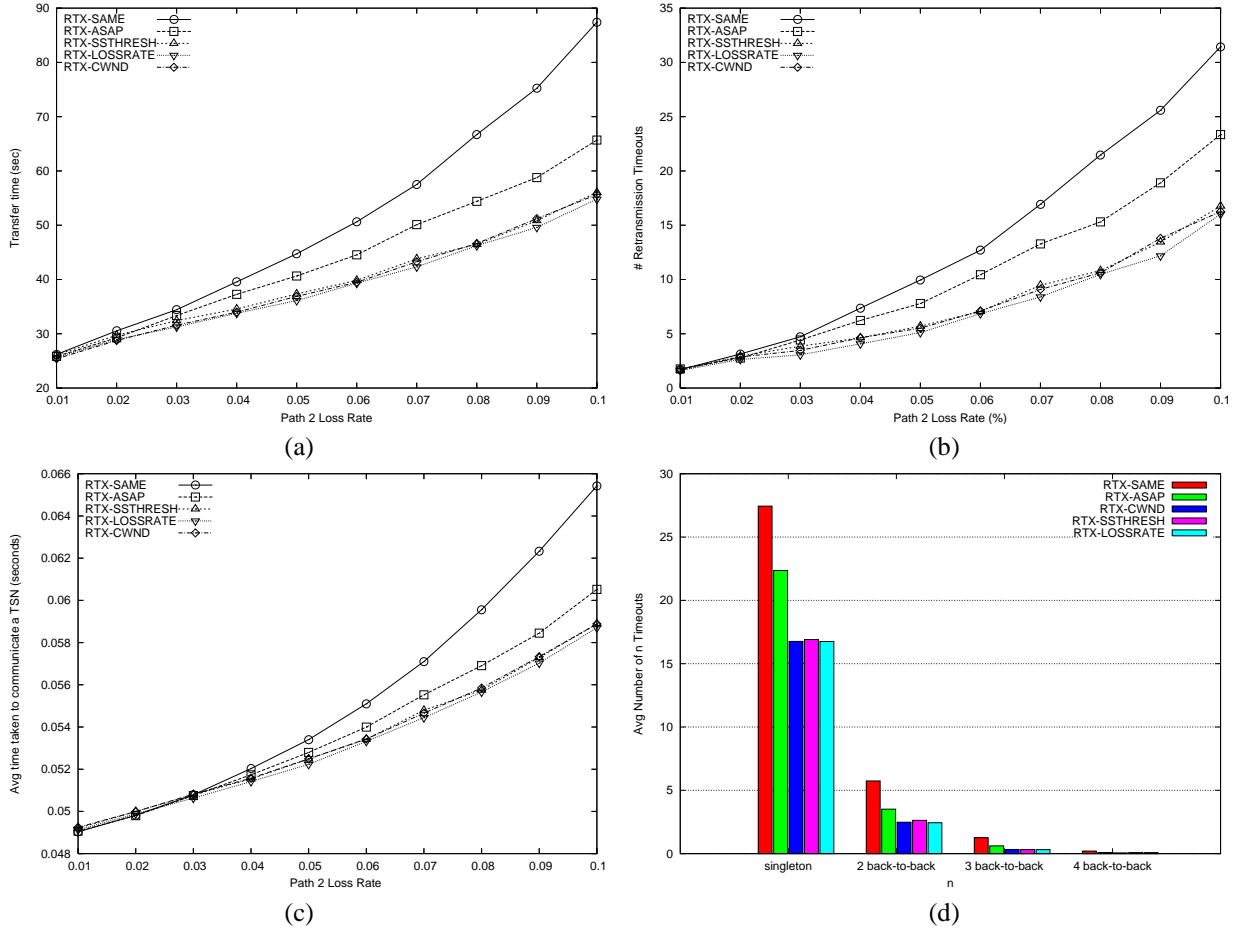


Fig. 4. With $rbuf = 64K$, and Path 1 loss rate = 1%.: (a) Time taken by CMT to transfer an 8MB file, (b) Number of retransmission timeouts for CMT with different retransmission policies, (c) Average time taken to successfully communicate a TSN with different retransmission policies, (d) Average number of back-to-back timeouts with different retransmission policies (Path 1 loss rate = 1%; Path 2 loss rate = 10%)

varying from 16KB to 256KB in Section V-B. This analysis provides us with an understanding of how much impact a constraining $rbuf$ (or the $rbuf$ blocking problem) has on the different policies. Finally in Section V-C, we study the impact of end-to-end delay on performance degradation due to $rbuf$ blocking.

A. Evaluation with $rbuf = 64KB$

Figure 4(a) shows the time taken for a CMT sender to transfer an 8MB file when the $rbuf$ is set to 64KB, using the five retransmission policies. Each plotted value is the mean of at least 100 simulation runs. RTX-SAME, the simplest to implement, performs worst. Its performance gap with the other policies increases as the loss rate on Path 2 increases. RTX-ASAP performs better than RTX-SAME, but still considerably worse than the three loss rate based policies which perform best. We present two causes for these differences.

Cause 1: Figure 4(b) shows the number of retransmission timeouts experienced when using the different policies. One

may conclude that improvement in using the loss rate based policies is due partly to fewer timeouts (and hence, timeout recovery periods). RTX-SAME does not consider loss rate and experiences the largest number of timeouts. RTX-ASAP does not consider loss rate and does better than RTX-SAME, but still experiences more timeouts than the loss rate based policies. This analysis supports our intuitive hypothesis - taking path loss rate into consideration while deciding the retransmission destination improves the chances of a retransmission getting through, and improves overall performance due to reduction of $rbuf$ blocking.

Cause 2: Figure 4(c) shows the average time taken to successfully communicate a TSN. This time is measured as the time taken from the first transmission of a TSN to the time when that TSN or one of its retransmissions finally reaches the receiver. RTX-SAME shows the highest average, suggesting that more retransmissions may be needed for a successfully communicating a TSN. Since recovery via fast

retransmission can happen only once for a given TSN⁷, the number of back-to-back timeouts may be higher with RTX-SAME than with the other policies. Each back-to-back timeout causes a sender's retransmission timeout value to double, thus doubling the timeout recovery period. Recall that the longer the timeout recovery period, the higher the probability and longer the duration for which rbuf blocking occurs. Thus, more back-to-back timeouts will degrade performance.

Figure 4(d) shows average number of n back-to-back timeouts for the different retransmission policies with Path 1 loss rate = 1%, and Path 2 loss rate = 10%. Overall, loss rate based policies experience about half the back-to-back timeouts that RTX-SAME does. Though the occurrences of $n=3$ and $n=4$ are few, their impact is significant - with our simulation parameters, at $n=3$ timeout recovery is 4 seconds, and at $n=4$ timeout recovery is 8 seconds. Thus, we argue that *performance degradation due to back-to-back timeouts can be significantly reduced by taking loss rate into account for making retransmission decisions.*

B. Evaluation with different rbufs

We also investigated the performance of the different retransmission policies using rbuf sizes of 16KB, 32KB, 128KB, and 256KB. The performance ranking of the different policies with these rbufs remains the same as with an rbuf of 64KB (Figure 4(a)); performance curves for these rbufs are therefore not shown⁸. We discuss a few salient points.

- With a large (i.e., minimally constraining) rbuf of 256KB, RTX-SAME still performs poorly due to a high number of timeouts, and the consequent throughput degradation. Each timeout causes cwnd reduction at a sender, and entails idle time with the sender not sending any data - causing throughput reduction.
- As the rbuf size decreases and becomes more of a constraint, degradation in CMT throughput occurs due to increased rbuf blocking as explained in Section III. All retransmission policies suffer in the face of a constrained rbuf. Even with a reasonably large rbuf of 128KB, some performance degradation occurs; i.e., even with large rbufs, rbuf blocking does occur, but to a lesser extent.
- As the rbuf becomes more of a constraint, degradation in throughput of RTX-SAME policy is markedly more than with the other retransmission policies. The reasons for this degradation are the same as described in Section V-A. Degradation is lowest in the loss rate based policies with an increasingly constraining rbuf.

In summary, retransmission policies that take loss rate into account perform better under the different rbuf values consid-

⁷The Multiple Fast Retransmit (MFR) algorithm allows recovery using fast retransmission multiple times on the same TSN [10]. The MFR algorithm has not been ported to CMT, and so the only recovery mechanism possible from the loss of a fast retransmission currently in CMT, as is the case currently in SCTP and TCP, is a timeout recovery.

⁸We refer the interested reader to the extended version of this paper [8] for these performance graphs.

ered. Of the loss rate based policies, the practical ones (RTX-CWND and RTX-SSTHRESH) perform similarly under all conditions considered. We arbitrarily select RTX-SSTHRESH as CMT's retransmission policy in further evaluations.

C. Evaluation under different end-to-end delays

Figure 5 shows relative performance degradation of CMT with RTX-SSTHRESH under different end-to-end delays - 10ms, 25ms, 45ms, 90ms, 180ms, and 360ms, yielding RTTs of 20ms, 50ms, 90ms, 180ms, 360ms, and 720ms, respectively. The delays on both paths to the receiver are symmetric (We are currently studying rbuf blocking with asymmetric paths). These values cover a range of RTTs experienced by majority of flows on the Internet [7]. Relative performance degradation is computed as the ratio

$$\frac{\text{CMT throughput with infinite (INF) rbuf}}{\text{CMT throughput with rbuf} = X}$$

as X varies from 16KB to 256KB along the X-axis (Note that along the Y-axis, smaller values are better).

The degradation curves for all end-to-end delays in Figure 5 show a knee at about 64KB. Also, as the end-to-end delay decreases, CMT's relative throughput degradation increases. Degradation is as much as 10 times when end-to-end delay is 10ms. We now explain why smaller delays have greater sensitivity to rbuf constraints.

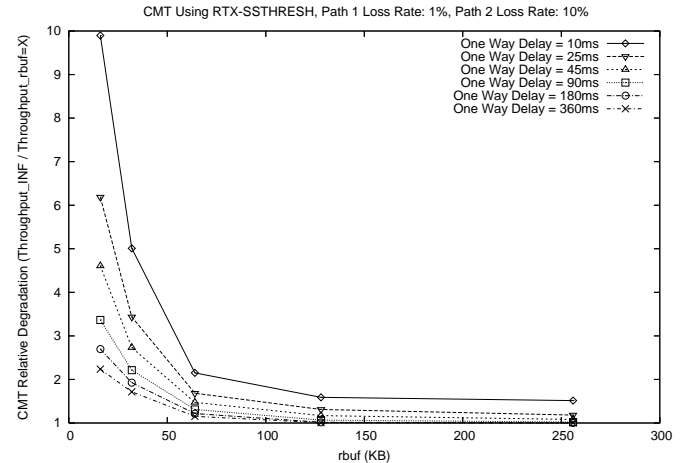


Fig. 5. Relative performance degradation of CMT with different end-to-end delays

Overall SCTP throughput, similar to TCP throughput, varies inversely with delay. As delay decreases, throughput increases. This relationship holds true for large rbuf conditions. Thus, in the relative performance degradation measure, the numerator (CMT throughput with infinite rbuf) increases as delay decreases.

As the rbuf size increasingly becomes a bottleneck, a different dynamic dominates. According to the SCTP specification [4] and the specification for computing TCP's retransmission timer [11], retransmission timeouts (RTOs) should have a (conservative) minimum value of 1 second to avoid spurious timeouts. These timeout recovery periods are thus independent of the end-to-end delays considered, since these delays are much lesser than 1 second. As rbuf increasingly constrains, the number of timeouts increases. Consequently, total time spent in timeout recovery (which is roughly the same irrespective of the end-to-end delay) increasingly dominates association lifetime. Thus, the denominator in the relative performance degradation measure (CMT throughput with $\text{rbuf}=X$, as X varies) does not increase as fast as the numerator with decreasing end-to-end delay, since the denominator is largely dictated by (constant) timeout recovery periods. Therefore, the influence of a constrained rbuf increases as end-to-end delay decreases. In summary, *CMT is more sensitive to rbuf constraints in environments (such as data centers [12]) with shorter end-to-end delay.*

We can thus see that rbuf blocking has a larger impact on associations with shorter end-to-end delay due to a minimum RTO value which is recommended [4], [11] and largely in use. We note that shorter minimum RTOs together with better RTT estimation algorithms [13], [14] may help remove this bias against shorter delay associations, but are beyond the scope of this work.

VI. SUMMARY AND DISCUSSION

We presented the rbuf blocking problem which causes degradation to a CMT sender by a constraining rbuf. We evaluated five retransmission policies for CMT under different rbuf constraints. Simulation results show that RTX-SAME, shown in [15] to work well in non-CMT environments, and whose performance represents that of a data striping application, performs poorest. Better performance results from any retransmission policy that takes loss rate into account. Of the practical loss rate based policies (RTX-CWND and RTX-SSTHRESH), we arbitrarily chose RTX-SSTHRESH as CMT's retransmission policy. Investigation under different end-to-end delays revealed that CMT is more sensitive to rbuf constraints in environments with shorter end-to-end delay.

We reemphasize that rbuf blocking is not specific to the transport layer; it applies to multipath transfer at other layers as well. rbuf blocking cannot be eliminated or reduced by moving the functionality of CMT to a different layer. *A significant benefit of CMT is that retransmission decisions can be made at the transport layer, thus considerably reducing rbuf blocking - a benefit that multipath transfer at any other layer does not have.*

We are currently investigating rbuf blocking and CMT performance with asymmetric paths. We are also investigating damage caused due to *stale acks*, and mechanisms to assist a sender in identifying such acks. Stale acks are acks that are received late on a higher delay return path with stale

information, and can affect sender estimates (e.g., RTT). With CMT, stale acks are received frequently by a data sender. Another area of future work is to investigate mechanisms to share available bandwidth when the end-to-end paths are not independent, and share a bottleneck.

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] J. Iyengar, K. Shah, P. Amer, and R. Stewart, "Concurrent Multipath Transfer Using SCTP Multihoming," in *SPECTS 2004*, San Jose, California, July 2004.
- [2] J. Iyengar, P. Amer, and R. Stewart, "Retransmission Policies For Concurrent Multipath Transfer Using SCTP Multihoming," in *ICON 2004*, Singapore, Nov. 2004.
- [3] 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, Nov. 2003, (work in progress).
- [4] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson, "Stream Control Transmission Protocol," RFC2960, Oct. 2000.
- [5] UC Berkeley, LBL, USC/ISI, and Xerox Parc, "ns-2 documentation and software," Version 2.1b8, 2001, www.isi.edu/nsnam/ns.
- [6] A. Caro and J. Iyengar, "ns-2 SCTP module," Version 3.2, December 2002, <http://pel.cis.udel.edu>.
- [7] S. Shakkottai, R. Srikant, A. Broido, and k. claffy, "The RTT Distribution of TCP Flows in the Internet and its Impact on TCP-based Flow Control," Tech. Rep., Cooperative Association for Internet Data Analysis (CAIDA), Feb. 2004.
- [8] J. Iyengar, P. Amer, and R. Stewart, "Receive Buffer Management For Concurrent Multipath Transport Using SCTP Multihoming," Tech Report TR2005-10, CIS Dept, University of Delaware, Jan. 2005.
- [9] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control," RFC2581, IETF, Apr. 1999.
- [10] A. Caro, P. Amer, J. Iyengar, and R. Stewart, "Retransmission Policies with Transport Layer Multihoming," in *ICON 2003*, Sydney, Australia, Sept. 2003.
- [11] V. Paxson and M. Allman, "Computing TCP's Retransmission Timer," RFC2988, IETF, Nov. 2000.
- [12] N. Jani and Krishna Kant, "SCTP Performance in Data Center Environments," Tech. Rep., Intel Corporation, 2005.
- [13] S. Bohacek, J. P. Hespanha, J. Lee, C. Lim, and K. Obraczka, "TCP-PR: TCP for Persistent Packet Reordering," in *IEEE ICDCS 2003*, Rhode Island, May 2003.
- [14] H. Ekstrom and R. Ludwig, "The Peak-Hopper: A New End-to-End Retransmission Timer for Reliable Unicast Transport," in *IEEE INFOCOM 2004*, Hong Kong, Mar. 2004.
- [15] A. Caro, P. Amer, and R. Stewart, "Transport Layer Multihoming for Fault Tolerance in FCS Networks," in *MILCOM 2003*, Boston, MA, Oct. 2003.