

Elsevier Editorial System(tm) for Computer Communications  
Manuscript Draft

Manuscript Number: COMCOM-D-09-00278R1

Title: Concurrent Multipath Transfer During Path Failure

Article Type: Fast Track: Hot Topics

Keywords: Stream Control Transmission Protocol (SCTP), Concurrent Multipath Transfer (CMT), Multihoming, Multipath transport, Path failure

Corresponding Author: Ms. Preethi Natarajan,

Corresponding Author's Institution:

First Author: Preethi Natarajan, PhD

Order of Authors: Preethi Natarajan, PhD; Paul Amer, PhD

**Abstract:** We investigate how path failure influences Concurrent Multipath Transfer (CMT) using SCTP multihoming. We show that CMT suffers from significant "receive buffer blocking" which degrades performance during both permanent and short-term failure. To improve performance, we introduce a new "Potentially-Failed" (PF) destination state, and revise CMT's failure detection and (re)transmission policies to include the PF state. Using simulation, we demonstrate that the modification called CMT-PF outperforms CMT during failure – even with aggressive failure detection thresholds and varying receive buffer constraints. In non-failure scenarios, CMT-PF performs on par or better but never worse than CMT. Finally, we confirm these simulation results using FreeBSD implementations of CMT and CMT-PF. Based on our findings, we recommend CMT-PF be used in existing and future CMT implementations and RFCs.

Suggested Reviewers:

Opposed Reviewers:



This is the cover letter for the manuscript titled "Concurrent Multipath Transfer during Path Failure".

Currently, this manuscript has not been submitted for publication anywhere, and it will not be submitted until a decision has been made by Computer Communications.

The manuscript presents original material which has not previously published, completely or in part, in another journal.

Some of the material has been previously published in conference proceedings.

[Natarajan 2006] and [Natarajan 2008] introduce the rbuf blocking problem in CMT during failures, and discuss the solution -- CMT-PF. Also, these papers analyze CMT vs. CMT-PF using ns-2 simulations.

Post [Natarajan 2008] we implemented CMT-PF in FreeBSD. Using emulations we analyzed CMT vs. CMT-PF during failure and congestion scenarios. This manuscript includes a discussion of the emulation results.

[Natarajan 2006] P. Natarajan, J. Iyengar, P. Amer, R. Stewart, "Concurrent Multipath Transfer using Transport Layer Multihoming: Performance under Network Failures," IEEE MILCOM, Washington D. C., USA, Oct 2006.

[Natarajan 2008] P. Natarajan, N. Ekiz, P. Amer, J. Iyengar, R. Stewart, "Concurrent Multipath Transfer using Transport Layer Multihoming: Introducing the Potentially-failed Destination State," IFIP-TC6 Networking, Singapore, May 2008.

E-component(s)

[Click here to download E-component\(s\): Natarajan2006.pdf](#)

E-component(s)

[Click here to download E-component\(s\): Natarajan2008.pdf](#)

This is a detailed response to the reviewers' comments for the manuscript titled "Concurrent Multipath Transfer during Path Failure".

All of the reviewers' comments were addressed as follows:

1. In the measurements and simulation you seem to use a single stream for all DATA chunks and all these chunks are transferred in-ordered. This should be made clear.

Clarified in simulation/emulation setup in Sections IV, VA, VI.

2. As an editorial nit: multiple line breaks are where they should not be. Please make sure that you avoid these.

We tried to take care of this and hope that the unwanted line breaks do not appear anymore. If they do, can you please specify the location(s)? Thanks.

3. Since there are also currently activities to add multi path support to TCP, it might be useful to also abstract from the SCTP specifics and provide the results in a generalized way such they might also be applicable to others who work on transport protocols using multiple paths simultaneously.

Included a paragraph in Section IID, describing how failure-induced rbuf blocking can be possible in multipath TCP. Section IIIB also notes that the CMT-PF framework can be adapted to multipath TCP.

# Concurrent Multipath Transfer During Path Failure

Preethi Natarajan<sup>1</sup>, Nasif Ekiz<sup>2</sup>, Paul D. Amer<sup>2</sup>, Randall Stewart<sup>3</sup>

<sup>1</sup>Cisco Systems, prenatar@cisco.com

<sup>2</sup>CIS Dept., University of Delaware, {ekiz,amer}@cis.udel.edu

<sup>3</sup>Researcher, randall@lakerest.net

**Abstract**—We investigate how path failure influences Concurrent Multipath Transfer (CMT) using SCTP multihoming. We show that CMT suffers from significant “receive buffer blocking” which degrades performance during both permanent and short-term failure. To improve performance, we introduce a new “Potentially-Failed” (PF) destination state, and revise CMT’s failure detection and (re)transmission policies to include the PF state. Using simulation, we demonstrate that the modification called CMT-PF outperforms CMT during failure – even with aggressive failure detection thresholds and varying receive buffer constraints. In non-failure scenarios, CMT-PF performs on par or better but never worse than CMT. Finally, we confirm these simulation results using FreeBSD implementations of CMT and CMT-PF. Based on our findings, we recommend CMT-PF be used in existing and future CMT implementations and RFCs.

**Keywords**—Stream Control Transmission Protocol (SCTP), Concurrent Multipath Transfer (CMT), Multihoming, Multipath transport, Path failure.

## I. INTRODUCTION

Multihoming among networked machines and devices is a technologically feasible and increasingly economical proposition. Unlike TCP and UDP, the Stream Control Transmission Protocol (SCTP) [RFC4960] 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, providing network interface redundancy and improved end-to-end fault tolerance.

Multihomed nodes may also be connected via multiple end-to-end paths. For instance, users may be simultaneously connected through dial-up/broadband, or via multiple wireless technologies (e.g., 802.11b, GPRS). Concurrent Multipath Transfer (CMT) [Iyengar 2006] is an experimental SCTP extension that assumes multiple *independent* paths, and exploits these paths for *simultaneous* transfer of new data between end hosts. This work investigates CMT performance during path failures. We also highlight the conditions under which our findings would be pertinent to other transports that support multipath transfer (e.g. multipath TCP [Wischik 2008]).

This paper is organized as follows. Section II motivates this research by discussing the commonness of link failures in the Internet. Section II also overviews CMT’s failure detection process, and how CMT’s throughput degrades during failure detection. Section III outlines our proposed improvement – CMT with a “Potentially-Failed” destination state (CMT-PF). Using ns-2 simulations, Sections IV and V evaluate CMT vs. CMT-PF in failure and congestion scenarios, respectively. Section VI validates the simulation results with empirical experiments using FreeBSD implementations of CMT and CMT-PF. Section VII concludes this work.

## II. PROBLEM DESCRIPTION

This section discusses the prevalence of Internet path failures, and outlines how CMT’s performance degrades during path failures.

- 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.

### 1 A. Prevalence of Path Failures

2 Internet paths fail when a router or a link connecting two routers fails due to planned maintenance or unplanned accidents such  
3 as hardware malfunction or software error. Ideally, the routing system detects unplanned link failures, and reconfigures routing  
4 tables to avoid trying to route traffic via a failed link. Using data from an ISP's routing logs, [Markopoulou 2004] observes that  
5 link failures are part of everyday operation. Around 80% of the failures are unplanned, and the time-to-repair for any particular  
6 failure can be on the order of hours. Using probes [Feamster 2003] observes that Internet paths fail often. The failures appear  
7 everywhere and are not confined to a few bad links.

8 Existing research also highlights problems with Internet backbone routing that result in long route convergence times.  
9 [Labovitz 2000] shows that Internet's interdomain routers may take as long as tens of minutes to reconstruct new paths after a  
10 failure. During these delayed convergences, end-to-end Internet paths experience intermittent loss of connectivity in addition to  
11 increased packet loss, latency, and reordering. Using probes, [Paxson 1997] and [Zhang 2000] find that "significant routing  
12 pathologies" prevent selected pairs of hosts from communicating about 1.5% to 3.3% of the time. Importantly, the authors also  
13 find that this trend has not improved with time. In [Dahlin 2003], the authors use probes to confirm that failure durations are  
14 heavy-tailed, and report that 5% of detected failures last more than 2.75 hours, and as long as 27.75 hours. The pervasiveness of  
15 path failures in practice motivates us to study their impact on CMT.

### 16 B. Failure Detection in CMT

17 Since CMT is an extension to SCTP, CMT retains SCTP's failure detection process. A CMT sender employs a tunable failure  
18 detection threshold called *Path.Max.Retrans (PMR)* [RFC4960]. As shown in the finite state machine of Figure 1, a destination is  
19 in one of the two states – *Active* or *Failed* (inactive). A destination is active as long as acks come back for data or heartbeats  
20 (probes) sent to that destination. When a sender experiences more than PMR consecutive timeouts while trying to reach a specific  
21 active destination, that destination is marked as failed. Only heartbeats (i.e., no data) are sent to a failed destination. A failed  
22 destination returns to active when the sender receives a heartbeat ack. RFC4960 proposes a default PMR = 5, which translates to  
23  $\geq 63$  seconds (6 consecutive timeouts) for failure detection.

24 Figure 1: Failure Detection in CMT

### 25 C. Receive Buffer Blocking in CMT

26 [Iyengar 2005] explores the "receive buffer blocking" problem in CMT, where Transport Protocol Data Unit (TPDU) losses  
27 throttle data transmission once the CMT receiver's buffer (rbuf) is filled with out-of-order data. Even though the sender's cwnd  
28 would allow new data to be transmitted, rbuf blocking (i.e., flow control) stalls the sender, causing throughput degradation.

29 Rbuf blocking problem cannot be eliminated in CMT [Iyengar 2005]. To reduce rbuf blocking's negative impact during  
30 congestion, [Iyengar 2005] proposes different retransmission policies that use heuristics for faster loss recovery. These policies  
31 consider different path properties such as loss rate and delay, and try to reduce rbuf blocking by sending retransmissions on a  
32 path with lower loss or delay. In practice, a path's loss rate can only be estimated, so [Iyengar 2005] proposed the  
33 RTX\_SSTHRESH policy, where retransmissions are sent on the path with the largest slow-start threshold. Since  
34 RTX\_SSTHRESH outperformed other retransmission policies during congestion, [Iyengar 2005] recommended the  
35 RTX\_SSTHRESH policy for CMT. However, [Iyengar 2005] did not consider CMT performance during path failures, which is  
36 the focus of this work.

### 37 D. CMT Performance during Path Failures

38 CMT's failure-induced rbuf blocking is explained via the timeline diagram shown in Figure 2. The CMT sender (A) has two  
39 interfaces –  $A_1$  and  $A_2$ , and transmits data to receiver (B) with two interfaces –  $B_1$  and  $B_2$ . All four addresses are bound in the  
40 CMT association such that the sender employs the 2 independent paths – path 1 and path 2, for data transmission.

41 Figure 2: CMT's Rbuf Blocking during Path Failure

42 In Figure 2, the initial cwnd for each path=2 MTUs. Each TPDU consists of an MTU-sized data chunk, and is assigned a  
43 unique Transmission Sequence Number (TSN) [RFC4960].  $C_i$  and  $O_i$  denote the cwnd in number of MTUs, and the number of  
44 outstanding TPDU, respectively, on path  $i$ . A SACK labeled  $\langle Sa, b-c; Rd \rangle$  cumulatively acknowledges all TSNs up to and  
45 including  $a$ , selectively acknowledges TSNs  $b$  through  $c$  (missing report for TSNs  $a+1$  through  $b-1$ ), and advertises a receiver  
46 window capable of buffering  $d$  more TSNs. In Figure 2's example, the transport layer receive buffer can hold a maximum of 5  
47 TSNs, and its contents are listed after the reception of every TSN.

48 Both forward and reverse paths between  $A_1$  and  $B_1$  fail just after TSN 2 enters the network. Hence, TSN 2 and the SACK for  
49 TSN 1 are presumed lost. TSNs 3 and 4 are received out of order and stored in the receive buffer. Each of these TSNs triggers a  
50 SACK to the sender. The CMT sender uses the *Cwnd Update for CMT* (CUC) algorithm [Iyengar 2006] to decouple a path's  
51 cwnd evolution and data ordering. On receiving the SACK triggered by TSN 3, the sender uses CUC to increment  $C_2$  to 3, and  
52



decrement O1 and O2 to 1. The available receive buffer space for new data, calculated as advertised receive window (=4) – total outstanding (=2), allows the sender to transmit two TSNs, 5 and 6, on path 2. On path 1, even though 1 MTU worth of new data could be transmitted ( $C1 > O1$ ), rbuf blocking, i.e., flow control, throttles data transmission. On receiving the SACK triggered by TSN 4, the sender increases  $C2=4$ , and decreases  $O2=2$ . However, lack of rbuf space inhibits transmission of new data on path 2.

Since  $O2 < C2$ , the SACKs triggered by TSNs 5 and 6 do not increment  $C2$  [RFC4960] (discussed later). But these SACKs decrement  $O2$ . Even though  $O2 < C2$ , flow control continues to prevent data transmission on path 2.

When path 1's retransmission timer expires, the sender detects the loss of TSN 2. Note that this timeout is the first of the 6 (when  $PMR = 5$ ) consecutive timeouts needed to detect path 1's failure. After this timeout,  $C1=1$ ,  $O1=0$ , and path 1's RTO value is doubled [RFC4960]. The CMT sender employs the RTX\_SSTHRESH policy and retransmits TSN 2 on path 2.

On receiving TSN 2, the CMT receiver delivers data from TSNs 2-6 to the application. The corresponding SACK advertises a receive window of 5 TSNs. Once receive buffer becomes available, the sender transmits TSN 7 on path 1, and TSNs 8-11 on path 2. Due to path 1 failure, TSN 7 is lost, and TSNs 8-11 are received out-of-order and fill the receiver's buffer. Once again, flow control inhibits data transmission until the sender experiences a timeout on path 1, detects loss of TSN 7, and successfully retransmits TSN 7 on path 2.

To generalize, a CMT sender continues to transmit data on a failed path until the corresponding destination is *marked* failed, i.e., until the sender experiences  $(PMR + 1)$  consecutive timeouts on the failed path. During failure detection, out-of-order data received via the non-failed path(s) fill the receive buffer and stall data transmission until the lost TPDU is successfully retransmitted. Since losses on a failed path are detected only after a timeout, the length of an rbuf blocking instance is proportional to the failed path's RTO. Until failure detection, the sender keeps transmitting data on the failed path, resulting in successive instances of rbuf blocking. Each instance is exponentially longer than the previous instance due to the exponential backoff of RTO values.

Reference [Wischik 2008] discusses the advantages of resource pooling and argues the need for multipath-capable TCP (a.k.a. multipath TCP) to facilitate resource pooling. Multipath TCP functions similar to CMT – multipath TCP extends TCP to support multihoming, and sets up multiple subflows to simultaneously transmit data over multiple paths. Reference [Wischik 2008] acknowledges that a single receive window can interact badly with multipath TCP connections. Data received out-of-order on one subflow (path) may fill the receive window and flow control would stall data transmission on other subflows (paths). Note that this problem is conceptually similar to CMT's congestion-induced rbuf blocking. Consequently, when multipath TCP's subflows share a single receive window, the multipath TCP connection will endure failure-induced rbuf blocking and degraded throughput.

In CMT, rbuf blocking results in the following side-effects that further degrade throughput:

*Preventing congestion window growth:* Note that rbuf blocking prevents the sender from fully utilizing the cwnd. When the amount of outstanding data is less than the cwnd, RFC4960 prevents the sender from increasing the cwnd for future SACKs. For example, in Figure 2, when the sender receives the SACKs for TSNs 5, 6, 9-11, the sender cannot increment  $C2$ .

*Reducing congestion window:* To reduce burstiness in data transmission, an SCTP sender employs a congestion window validation algorithm similar to RFC2861. During every transmission, the sender uses the *MaxBurst* parameter (recommended value of 4) as follows:

$$\text{If } ((\text{outstanding} + \text{MaxBurst} * \text{MTU}) < \text{Cwnd})$$

$$\text{Cwnd} = \text{outstanding} + \text{MaxBurst} * \text{MTU}$$

This algorithm decays the cwnd after an idle period so that at the next sending opportunity, the sender will not transmit more than  $(\text{MaxBurst} * \text{MTU})$  bytes of data. During rbuf blocking, the amount of outstanding data can become less than the cwnd. In such cases, the above rule is triggered and further reduces the cwnd. In Figure 2, when the SACK triggered by TSN 11 arrives at the sender,  $O2$  decrements to 0. The window validation algorithm causes  $C2$  to be reduced to 4 ( $O2 (=0) + \text{MaxBurst} (=4)$ ).

### III. SOLUTION

[Caro 2005] recommends lowering the PMR value for SCTP flows in Internet-like environments. Correspondingly, lowering the PMR for CMT flows reduces the number of rbuf blocking episodes during failure detection. However, lowering the PMR is an incomplete solution to the problem since a CMT flow may be rbuf blocked for any  $PMR > 0$  (Section IV). Also, a tradeoff exists on deciding the value of PMR – a lower value reduces rbuf blocking but increases the chances of spurious failure detection, whereas a higher PMR increases rbuf blocking and reduces spurious failure detection in a wide range of environments.

#### A. Details of CMT-PF

To mitigate failure-induced rbuf blocking, our proposed solution introduces a new “Potentially-Failed” state to the FSM of Figure 1. The expanded FSM, shown in Figure 3, assumes 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 “Potentially-Failed” (PF). A PF destination is not used for data transmission or retransmission. CMT's retransmission policies are augmented to include the PF state. CMT with the new set of retransmission policies is called CMT-PF [Natarajan 2006]. Details of CMT-PF are:

- If a TPDU loss is detected by RFC4960's threshold number of missing reports, one of CMT's current retransmission policies, such as `RTX_SSTHRESH`, is used to select an active destination for "fast" retransmission.
- If a TPDU loss is detected after a timeout, the corresponding destination transitions to the PF state (Figure 3). The sender never transmits data to a PF destination. However, when all destinations are in the PF state, the sender transmits data to the destination with the fewest consecutive timeouts. In case of a tie, data is sent to the last active destination. This exception ensures that CMT-PF does not perform worse than CMT when all paths have potentially failed (discussed further in Section V).
- Heartbeats are sent to PF destination(s) with an exponential backoff of RTO after every timeout until either (i) a heartbeat ack transitions the destination back to the active state, or (ii) an additional PMR consecutive timeouts confirm the path failure, upon which the destination transitions to the failed state, and heartbeats are sent with a lower frequency as described in RFC4960.
- If ever a heartbeat ack indicates a PF destination is alive, that destination's cwnd is set to either 1 MTU (CMT-PF1), or 2 MTUs (CMT-PF2), and the sender follows the slow start algorithm to transmit data to this destination. Detailed analysis on the cwnd evolution of CMT-PF1 vs. CMT-PF2 can be found in Section V.
- Acks for retransmissions do not transition a PF destination back to the active state, since a sender cannot disambiguate whether the ack was for the original transmission or the retransmission(s).

Figure 3: Failure Detection in CMT-PF ( $PMR > 0$ )

#### B. CMT-PF Data Transfer during Failure

Figure 4 depicts an analogous CMT-PF timeline for the scenario described in Figure 2. All events are identical between the two figures up to the first timeout on path 1. After this timeout, the CMT-PF sender transitions path 1 to the PF state, transmits a heartbeat on path 1, and retransmits TSN 2 on path 2. The heartbeat loss on the failed path (path 1) is detected on the next timeout. This timeout is the second of  $(PMR + 1)$  consecutive timeouts required to detect path 1 failure. Meanwhile, receiver buffer space is released once the retransmitted TSN 2 is received on path 2. From this point onwards, data is transmitted only on path 2, without further rbuf blocking.

Section IID discussed how other multipath transports (e.g. multipath TCP [Wischik 2008]) may suffer from rbuf blocking during path failures. We note that the CMT-PF framework can be adapted to such multipath transports to mitigate the negative consequences of failure-induced rbuf blocking.

Figure 4: CMT-PF Reduces Rbuf Blocking during Path Failure

## IV. EVALUATIONS DURING FAILURE

CMT-PF is evaluated using the University of Delaware's ns-2 SCTP/CMT module [NS-2, Ekiz 2007]. The failure experiments discussed in this section use a simple topology and loss model (Figure 5). In the next section, we employ a more realistic (and complex) simulation topology and loss model to evaluate CMT and CMT-PF during congestion.

Figure 5: Simulation Topology for Failure Experiments

Figure 6: CMT vs. CMT-PF during Permanent Path Failure

Figure 7: CMT vs. CMT-PF under Varying PMR Values

In Figure 5, the multihomed sender, A, has two independent paths to the 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 coast-to-coast delays experienced by significant fraction of the flows in the Internet [Shakkottai 2004]. The sender A transfers an 8MB file to receiver B using both path 1 and path 2. This file transfer uses a single-streamed CMT or CMT-PF association and data is always delivered in-order to the receiving application. Path 2 fails during the file transfer; this failure is simulated by bringing down the bidirectional link between routers  $R_{20}$  and  $R_{21}$ . Unless stated otherwise, the  $PMR=5$ ,  $rbuf=64KB$ , and both paths experience Bernoulli losses with low loss rate ( $<1\%$ ).

#### A. Evaluations during Permanent Path Failure

In the following experiments, path 2 fails permanently 5 seconds after the file transfer begins.

##### 1) Single Permanent Path Failure

Neither path experiences congestion in this experiment, which helps to highlight how CMT and CMT-PF differ during failure detection.

Path 2's failure causes back-to-back timeouts at the sender. Both senders (CMT and CMT-PF) experience the first timeout on path 2 at ~6 seconds, and detect the failure after 6 back-to-back timeouts (PMR=5), at ~69 seconds (Figure 6). During the failure detection period, CMT continues to transmit data on path 2, experiencing consecutive timeouts and recurring rbuf blocking instances. CMT's throughput suffers until 69 seconds (i.e., until failure detection), after which CMT uses path 1 alone and completes the file transfer at around 80 seconds. On the other hand, CMT-PF transitions path 2 to PF state after the first timeout, and then transmits only heartbeats on path 2 avoiding further rbuf blocking. Reduced rbuf blocking helps CMT-PF to complete the file transfer (~15 seconds) using path 1 alone, even before path 2's failure is detected.

## 2) Varying Failure Detection Thresholds (PMR Values)

To achieve faster yet robust failure detection, [Caro 2005] argues for varying the PMR based on a network's loss rate, and suggests PMR=3 for the Internet. Since the sender detects a permanent path failure after (PMR+1) consecutive timeouts, CMT's failure-induced rbuf blocking varies as the PMR varies. Let,

$T_f$  = time when path 2 fails

$T_d$  = time when the sender detects path 2 failure (after PMR+1 consecutive timeouts)

The goodput during failure detection ( $G$ ) is defined as,

$$G = (\text{application data received during } T_f \text{ to } T_d) \div (T_d - T_f)$$

Figure 7 plots CMT vs. CMT-PF average goodput ( $G$ ) with 5% error margin. The dashed line in Figure 7 denotes the maximum attainable goodput of an SCTP file transfer (application data received  $\div$  transfer time) using path 1 alone.

When the failure detection threshold is most aggressive (PMR=0), both CMT and CMT-PF detect path 2 failure after the first timeout. Effectively, CMT-PF's failure detection never engages the PF state transition (Figure 3), and is equivalent to CMT's failure detection procedure. Both senders experience identical rbuf blocking, and perform similarly (Figure 7). As PMR increases, the number of rbuf blocking instances during failure detection increases, resulting in increasing performance benefits with CMT-PF. In Figure 7 as PMR and the failure detection period increase, 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 goodput. To conclude, *during path failures, CMT-PF performs as well as CMT for PMR=0, and better than CMT for PMR > 0.*

## B. Evaluations during Short-Term Failure

In the following experiments, path 2 in Figure 5 fails temporarily during the file transfer between A and B. The link connecting routers  $R_{20}$  and  $R_{21}$  goes down after 5 seconds from the start of file transfer, and is restored 5 seconds later.

### 1) During Single Short-term Failure

Neither path experiences congestion in this experiment, which helps to highlight the differences between CMT and CMT-PF during a short-term failure.

Figure 8: CMT vs. CMT-PF during Short-term Path Failure

Figure 9: CMT vs. CMT-PF under Varying Receive Buffer Sizes

The short-term failure is long enough for the sender (CMT or CMT-PF) to experience three back-to-back timeouts on path 2. As in the failure case, CMT transmits data on path 2 after each of these timeouts, while CMT-PF does not. Therefore, CMT suffers from consecutive rbuf blocking and lower throughput than CMT-PF (Figure 8). Once path 2 recovers at time=10, CMT's data and CMT-PF's heartbeat transmissions on the path (after the 3rd timeout - ~12.5 seconds) are successful, and both CMT and CMT-PF complete the file transfer without further rbuf blocking.

### 2) Varying Receive Buffer Sizes

This second short-term failure experiment analyzes CMT vs. CMT-PF for varying levels of receive buffer constraints (receive buffer sizes). Let

$T_f$  = time when path 2 fails

$T_r$  = time when path 2 is restored

The goodput during the short-term failure ( $G$ ) is defined as,

$$G = (\text{application data received during } T_f \text{ to } T_r) \div (T_r - T_f)$$

Figure 9 plots CMT vs. CMT-PF average goodput ( $G$ ) with 5% error margin. As the receive buffer becomes more constrained, i.e., as rbuf size decreases, the chances of rbuf blocking increases. *Consequently, CMT-PF's ability to alleviate rbuf blocking is more valuable for smaller rbuf sizes.*

## V. EVALUATIONS DURING CONGESTION

In this section we investigate how CMT-PF performs when timeouts are caused by non-failure scenarios such as congestion [Natarajan 2008]. Consider the case when a timeout on path  $p$  is due to congestion rather than failure. Depending on the rbuf size and the different paths' characteristics, the transport sender may or may not be rbuf blocked, leading to the following two scenarios:

*Sender is rbuf-limited:* Both CMT and CMT-PF senders cannot transmit new data until the rbuf blocking is cleared, i.e., until after successful retransmission(s) of lost data. The only difference is that CMT considers  $p$  for retransmissions, whereas CMT-PF transmits a heartbeat on  $p$ , and tries to retransmit lost data on other active paths. (If all destinations are in the PF state, the CMT-PF sender transitions the destination with the fewest consecutive timeouts to the active state (Section IIIA), and retransmits lost data to this destination.)

*Sender is not rbuf-limited:* Assume that SCTP PDUs (data or heartbeats) transmitted after the first timeout on path  $p$  are successfully received. In CMT, the cwnd allows 1 MTU of new data transmission on  $p$  (Figure 10a), and the corresponding SACK increments path  $p$ 's cwnd by 1 MTU. After 1 RTT after the timeout (shown by point A in Figure 10a), (i)  $p$ 's cwnd is 2 MTUs, and (ii) 1 MTU of new data has been successfully sent on  $p$ .

CMT-PF transmits a heartbeat on  $p$  and new data on other active path(s). (Note: if all destinations are marked PF, the CMT-PF sender transitions one of the PF destinations to the active state.) Path  $p$  is marked active when the heartbeat ack reaches the sender. Therefore, after 1 RTT from the timeout (point B in Figure 10b), (i)  $p$ 's cwnd is 1 MTU (CMT-PF1), and (ii) no new data has been sent on  $p$ . Comparing points A and B in Figures 10a and 10b, respectively, it can be seen that CMT has a 1 RTT "lead" in  $p$ 's cwnd growth. Assuming no further losses on  $p$ , after  $n$  RTTs, the cwnd on  $p$  will be  $2^n$  with CMT, and  $2^{n-1}$  with CMT-PF1.

To avoid the 1 RTT lag in CMT-PF1's cwnd evolution, we propose CMT-PF2 which initializes  $p$ 's cwnd to 2 MTUs after receiving a heartbeat ack (shown by point C in Figure 10c). Assuming that today's Internet router queues deal with packets rather than bytes, the successful routing of a heartbeat PDU is equivalent to the successful routing of a data PDU. Hence, a heartbeat ack can be used to clock the transport layer sender in the same way as a data ack. (Under similar reasoning, even an SCTP sender could initialize a destination's cwnd to 2MTUs after transitioning the destination from failed to active state, but that recommendation is outside the scope of this paper.) Since CMT-PF2 is clearly superior, we no longer consider CMT-PF1, and in the following sections, any reference to CMT-PF implies CMT-PF2.

(a): CMT

(b): CMT-PF1

(c): CMT-PF2

Figure 10: CMT vs. CMT-PF during Congestion

#### A. Simulation Setup

In the dual-dumbbell topology shown in Figure 11, 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 background cross-traffic that instigates bursty periods of congestion, and hence bursty congestion losses at the routers. Their last-hop propagation delays are uniformly distributed between 5-20 ms, resulting in end-to-end one-way propagation delays ranging ~35-65ms [Shakkottai 2004]. All links (both edge and core) have a buffer size twice the link's bandwidth-delay product, a reasonable setting in practice. Based on intuition, we believe that the final conclusions regarding CMT vs. CMT-PF are independent of the actual bandwidth and delay configurations used in the topology, as long as these configurations are similar on both paths.

Figure 11: Topology for Non-failure Experiments

Each single-homed edge node has eight traffic generators, introducing cross-traffic with a Pareto distribution [CAIDA, Fraleigh 2003]. The cross-traffic flows start at random times during the initial 5 seconds of the simulation. After an initial warm-up period, at time=10 seconds, sender A begins transmitting a 32MB file to receiver B over paths 1 and 2. This file transfer uses a single-streamed CMT or CMT-PF association and data is always delivered in-order to the receiving application. For both CMT and CMT-PF flows, rbuf=64KB, PMR=5, and loss rates are controlled by varying the cross-traffic load. The graphs in the subsequent discussions plot the average goodput (file size ÷ transfer time) of CMT vs. CMT-PF with 5% error margin.

#### B. Evaluations during Symmetric Loss Conditions

In the symmetric loss case, the aggregate cross-traffic load on both paths are similar, and vary from 40%-100% of the core link's bandwidth.

Both CMT and CMT-PF perform similarly (Figure 12) during low loss rates (i.e., low cross-traffic), since, most of the TPDU losses are recovered via fast retransmits, not timeout recoveries. As the cross-traffic load increases causing the loss rate to

increase, the number of timeouts on each path increases and *CMT-PF performs on par or insignificantly better than CMT (Figure 12)*. This confirms that the PF state transition does not penalize CMT-PF performance during symmetric path loss conditions.

Figure 12: CMT vs. CMT-PF during Symmetric Loss

Table 1: CMT vs. CMT-PF Mean Consecutive Data Timeouts on Path 2

Figure 13: CMT vs. CMT-PF during Asymmetric Loss Conditions

Table 2: CMT vs. CMT-PF Mean Number of Transmissions

### C. Evaluations during Asymmetric Loss Conditions

In practice, different paths are likely to experience asymmetric loss rates. For the next set of experiments, paths 1 and 2 experience different cross-traffic loads. The aggregate background cross-traffic on path 1 is set to 50% of the core link bandwidth, while the background on path 2 varies from 50%-100% of the core link bandwidth.

Note that rbuf blocking depends on the frequency of loss events (loss rate), and the duration of loss recovery. As the 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 RTO. As path 2's cross-traffic load increases, the probability that a sender experiences back-to-back timeouts on path 2 increases. CMT suffers more consecutive timeouts on data (Table 1) resulting in longer rbuf blocking periods when compared with CMT-PF. Therefore, as the asymmetry of the paths increases, CMT-PF performs better than CMT (Figure 13).

The asymmetric loss experiment helps highlight an important difference in CMT vs. CMT-PF's transmission strategy. In CMT, RTX\_SSTHRESH is a retransmission policy, and is not applied to 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.*

## VI. CMT-PF IMPLEMENTATION

We extended the FreeBSD CMT implementation to incorporate CMT-PF. The following experiments were performed using this implementation with the intention of validating the simulation-based conclusions in Sections IV and V.

The experimental topology, shown in Figure 14, consists of a client and server node running FreeBSD 7 —, and an intermediate node running the Dummynet traffic shaper [Rizzo 1997] which emulates Bernoulli losses. The server and client are connected by two independent paths, with symmetric bandwidth and propagation delay characteristics. The forward and reverse queue sizes for both paths are set to 1000KB. The transport layer receive window=64KB, and PMR=5. At time=0, the server initiates a bulk file transfer to the client. This file transfer uses a single-streamed CMT or CMT-PF association and data is always delivered in-order to the receiving application.

Figure 14: Emulation Topology for CMT vs. CMT-PF Experiments

### A. Single Permanent Path Failure

This single permanent path failure scenario is similar to the scenario described in Section IVA1. Neither path experiences loss. At time=5s, path 2 fails; this failure is emulated by having Dummynet block all packets traversing via path 2 to and from the client and server, respectively. Figure 15 plots the cumulative bytes received at the client during this transfer.

As observed in the simulations (Figure 6), path 2's failure causes consecutive timeouts and rbuf blocking instances in CMT, which prevents data transmission until failure detection time=69s. After failure detection, CMT transmits data using only path 1, completing the file transfer at time=80s. The CMT-PF sender transitions path 2 to PF after the first timeout (at time=6.5s), and transmits only heartbeats on path 2. Data transmission continues on path 1 and the file transfer finishes at time=18s. This

experiment confirms that the *FreeBSD CMT-PF implementation behaves as expected during a path failure.*

### B. Symmetric and Asymmetric Loss Conditions

Similar to the congestion scenarios in Section V, the following two experiments compare CMT and CMT-PF when the paths experience symmetric and asymmetric congestion levels, respectively. Note that the emulation and simulation topologies assume different loss models. Therefore, to facilitate a reasonable comparison between the emulation and simulation results, the Bernoulli loss rates for paths in Figure 14 were adjusted to be similar to the mean loss rates observed in the cross-traffic simulations.

In the first experiment, both paths experience symmetric loss rates, varying from 0.3%-14%. Figure 16 plots the average goodput (file size ÷ transfer time) of CMT vs. CMT-PF with 5% error margin. As observed in the simulations (Figure 12), both CMT and CMT-PF perform similarly at low loss rates, and this trend continues at higher loss rates.

In the second experiment, paths 1 and 2 experience asymmetric loss rates; loss rate on path 1 is fixed at 0.3% while path 2's loss rate varies from 0.3%-14%. As observed in the simulations (Figure 13), (i) both CMT and CMT-PF perform similar under low loss rates (Figure 17), and (ii) as the paths become more asymmetric in their loss conditions, CMT-PF performs similarly or slightly better than CMT. However, the absolute goodput difference between CMT and CMT-PF in Figure 17 is not as significant as observed in Figure 13, likely because the emulations and simulations employ different loss models. Nonetheless, our emulation experiments confirm that *CMT-PF performs on par or better but never worse than CMT during congestion.*

Figure 15: CMT vs. CMT-PF during Permanent Path Failure

Figure 16: CMT vs. CMT-PF during Symmetric Loss Conditions

Figure 17: CMT vs. CMT-PF during Asymmetric Loss Conditions

## VII. SUMMARY

Using simulation, we demonstrated that retransmission policies using CMT with a "potentially-failed" destination state (CMT-PF) outperform CMT during permanent and short-term failures. During permanent failures, CMT-PF employs a better failure detection process even under aggressive failure detection thresholds and varying receive buffer constraints.

Investigations also revealed that the PF state transition does not penalize CMT-PF performance during congestion. CMT-PF performs as well as CMT when the paths experience symmetric congestion levels, and similar or better than CMT when the paths experience asymmetric congestion levels. These conclusions were confirmed using real implementations of CMT and CMT-PF in FreeBSD.

Since *our findings demonstrate CMT-PF performs better or similar but never worse than CMT*, we recommend CMT be replaced by CMT-PF in existing and future implementations and RFCs. Finally, we note that other multipath transports (e.g. multipath TCP [Wischik 2008]) may also suffer from rbuf blocking during path failures, and the CMT-PF framework can be adapted to such transports to alleviate the negative consequences of failure-induced rbuf blocking.

## ACKNOWLEDGMENT

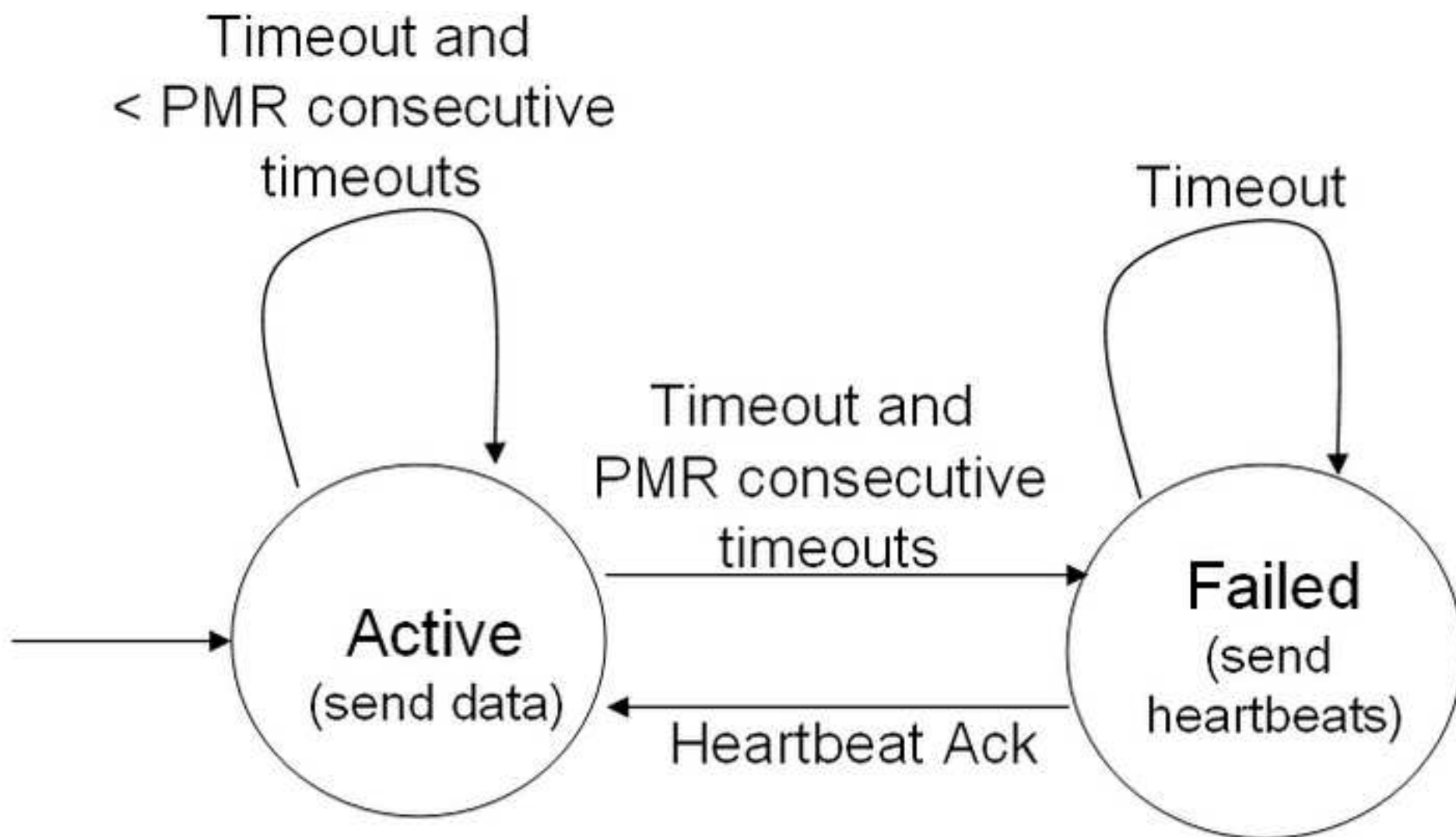
The authors thank Joseph Szymanski for helping implement CMT-PF on FreeBSD. The authors also thank Janardhan Iyengar, Jon Leighton and Ertugrul Yilmaz for their helpful comments and suggestions.

## REFERENCES

- [CAIDA] CAIDA: Packet Sizes and Sequencing, Mar 1998. [www.caida.org](http://www.caida.org)
- [Caro 2005] A. Caro, "End-to-End Fault Tolerance using Transport Layer Multihoming," PhD Dissertation, Dept. of Computer & Info Sciences, Univ. of Delaware, Aug 2005.
- [Dahlin 2003] M. Dahline, B. Chandra, L. Gao, A. Nayate, "End-to-End WAN Service Availability," IEEE/ACM Transactions on Networking, 11(2), 300-313, Apr 2003.
- [Ekiz 2007] N. Ekiz, P. Natarajan, J. Iyengar, A. Caro, "ns-2 SCTP Module," Version 3.7, Sep 2007. [pel.cis.udel.edu](http://pel.cis.udel.edu).
- [Feamster 2003] N. Feamster, D. Andersen, H. Balakrishnan, M. Kaashoek, "Measuring the Effects of Internet Path Faults on Reactive Routing," ACM SIGMETRICS, San Diego, Jun 2003.
- [Fraleigh 2003] 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, 17(6), 6-16, Nov 2003.
- [Iyengar 2005] J. Iyengar, P. Amer, R. Stewart, "Receive Buffer Blocking in Concurrent Multipath Transfer," IEEE GLOBECOM, St. Louis, Nov 2005.
- [Iyengar 2006] J. Iyengar, P. Amer, R. Stewart, "Concurrent Multipath Transfer using SCTP Multihoming over Independent End-to-end Paths," IEEE/ACM Transactions on Networking, 14(5), 951-964, Oct 2006.
- [Labovitz 2000] C. Labovitz, A. Abuja, A. Bose, F. Jahanian, "Delayed Internet Routing Convergence," ACM SIGCOMM, Stockholm, Aug 2000.
- [Markopoulou 2004] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C. Chuah and C. Diot, "Characterization of Failures in an IP Backbone," IEEE INFOCOM, Hong Kong, China, Mar 2004.
- [Natarajan 2006] P. Natarajan, J. Iyengar, P. Amer, R. Stewart, "Concurrent Multipath Transfer using Transport Layer Multihoming: Performance under Network Failures," IEEE MILCOM, Washington D. C., USA, Oct 2006.

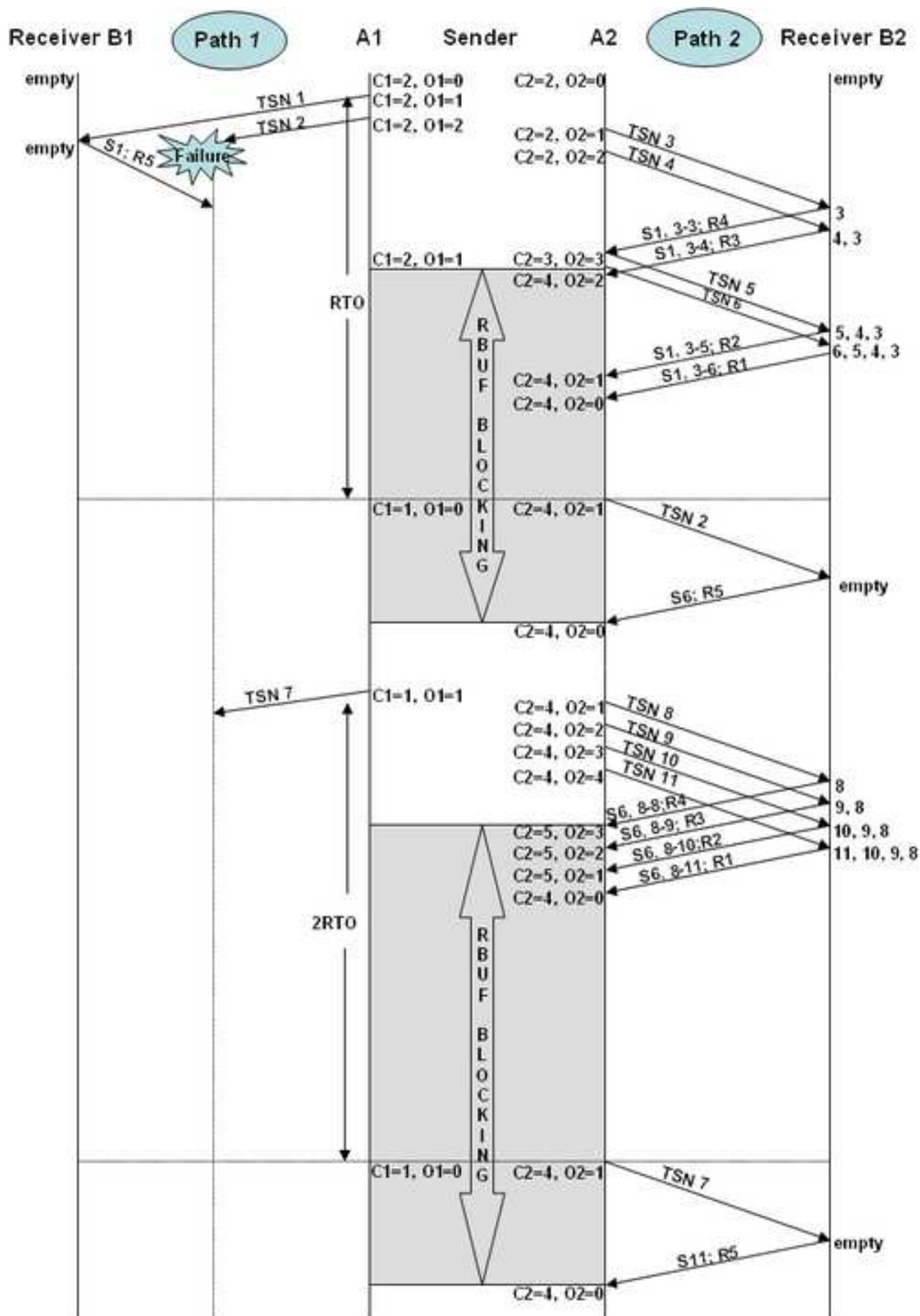
- [Natarajan 2008] P. Natarajan, N. Ekiz, P. Amer, J. Iyengar, R. Stewart, "Concurrent Multipath Transfer using Transport Layer Multihoming: Introducing the Potentially-failed Destination State," IFIP-TC6 Networking, Singapore, May 2008.
- [Natarajan 2009] P. Natarajan, "Leveraging Innovative Transport Layer Services for Improved Application Performance," PhD Dissertation, Dept. of Computer & Info Sciences, Univ. of Delaware, Feb 2009.
- [NS-2] ns-2 documentation and software, Version 2.33, Mar 2008. [www.isi.edu/nsnam/ns](http://www.isi.edu/nsnam/ns).
- [Paxson 1997] V. Paxson, "End-to-end Routing Behavior in the Internet," IEEE/ACM Transactions on Networking, 5(5), pp. 601–615, Oct 1997.
- [RFC2861] M. Handley, J. Padhye, S. Floyd, "TCP Congestion Window Validation," RFC 2861, Jun 2000.
- [RFC4960] R. Stewart, "Stream Control Transmission Protocol," RFC 4960, Sep 2007.
- [Rizzo 1997] L. Rizzo, "Dummynet: A Simple Approach to the Evaluation of Network Protocols," ACM Computer Communications Review, 27(1), 31–41, Jan 1997.
- [Shakkottai 2004] 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," Tech Report, Cooperative Association for Internet Data Analysis (CAIDA), Feb, 2004.
- [Wischik 2008] D. Wischik, M. Handley, M. Braun, "The Resource Pooling Principle," In ACM Computer Communications Review, 38(5), 47–52, Oct 2008.
- [Zhang 2000] Y. Zhang, V. Paxson, S. Shenker, "The Stationarity of Internet Path Properties: Routing, Loss, and Throughput," ACIRI Tech Report, May 2000.

Figure(s)  
[Click here to download high resolution image](#)

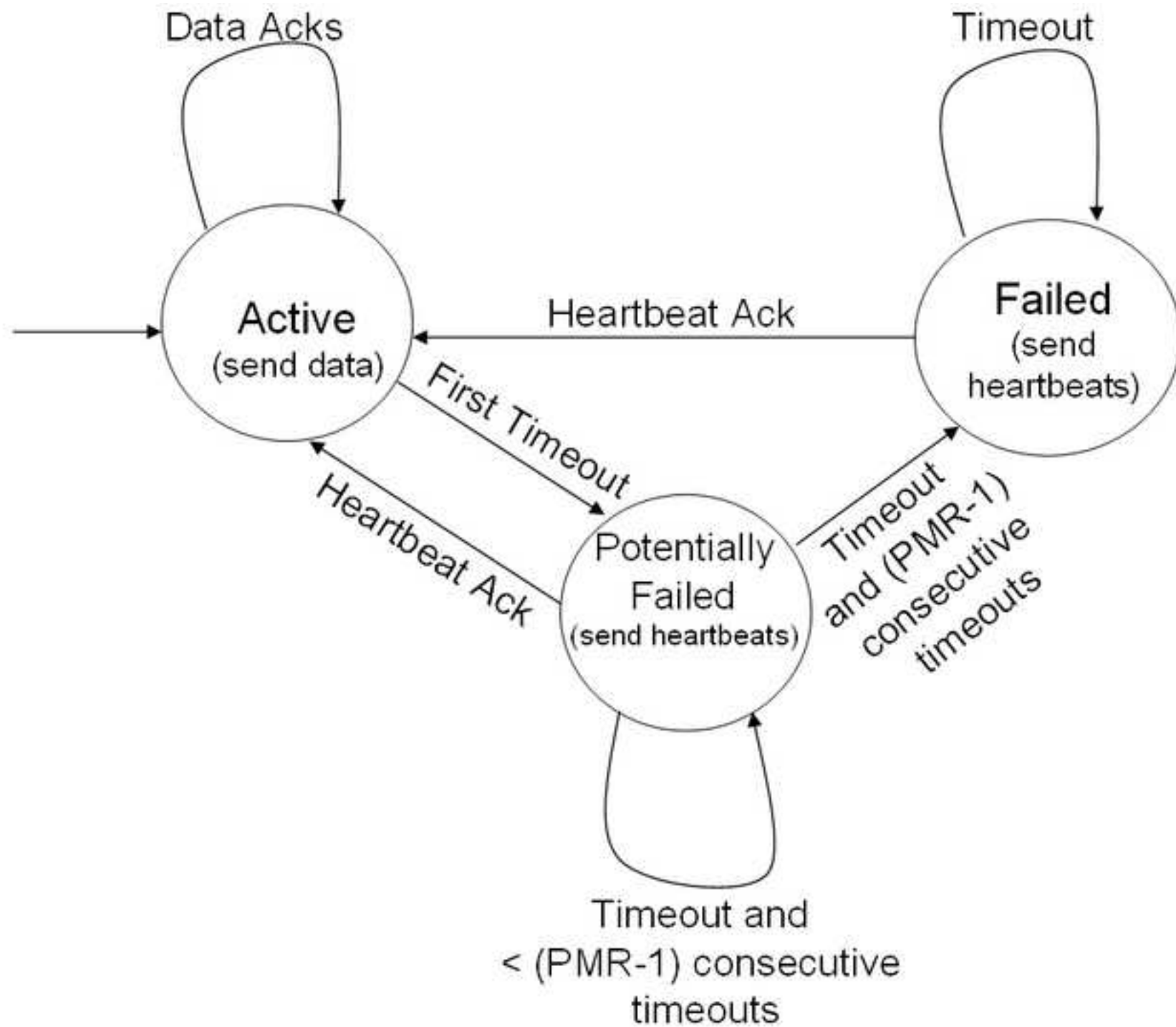




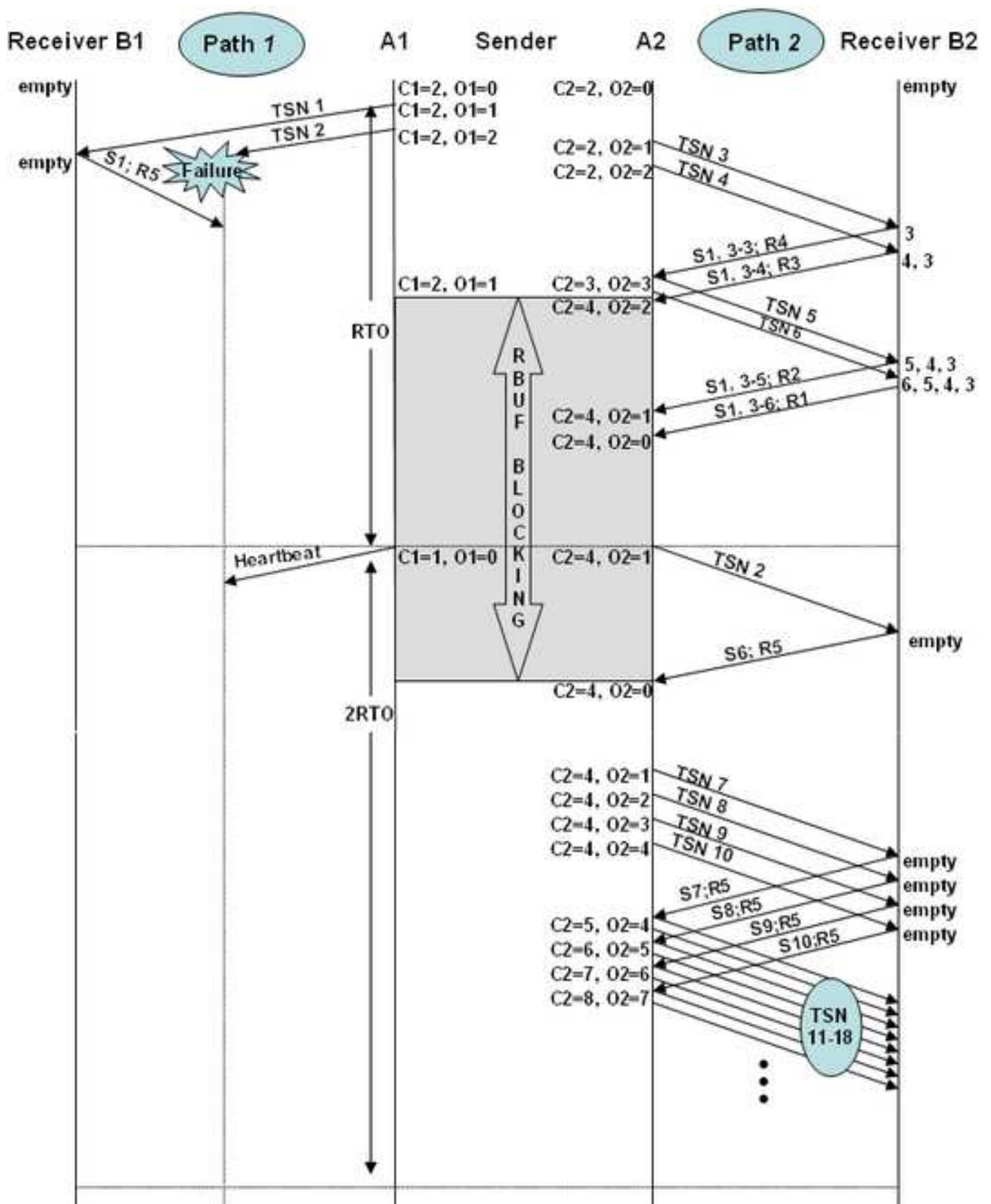
Figure(s)  
[Click here to download high resolution image](#)



Figure(s)  
[Click here to download high resolution image](#)

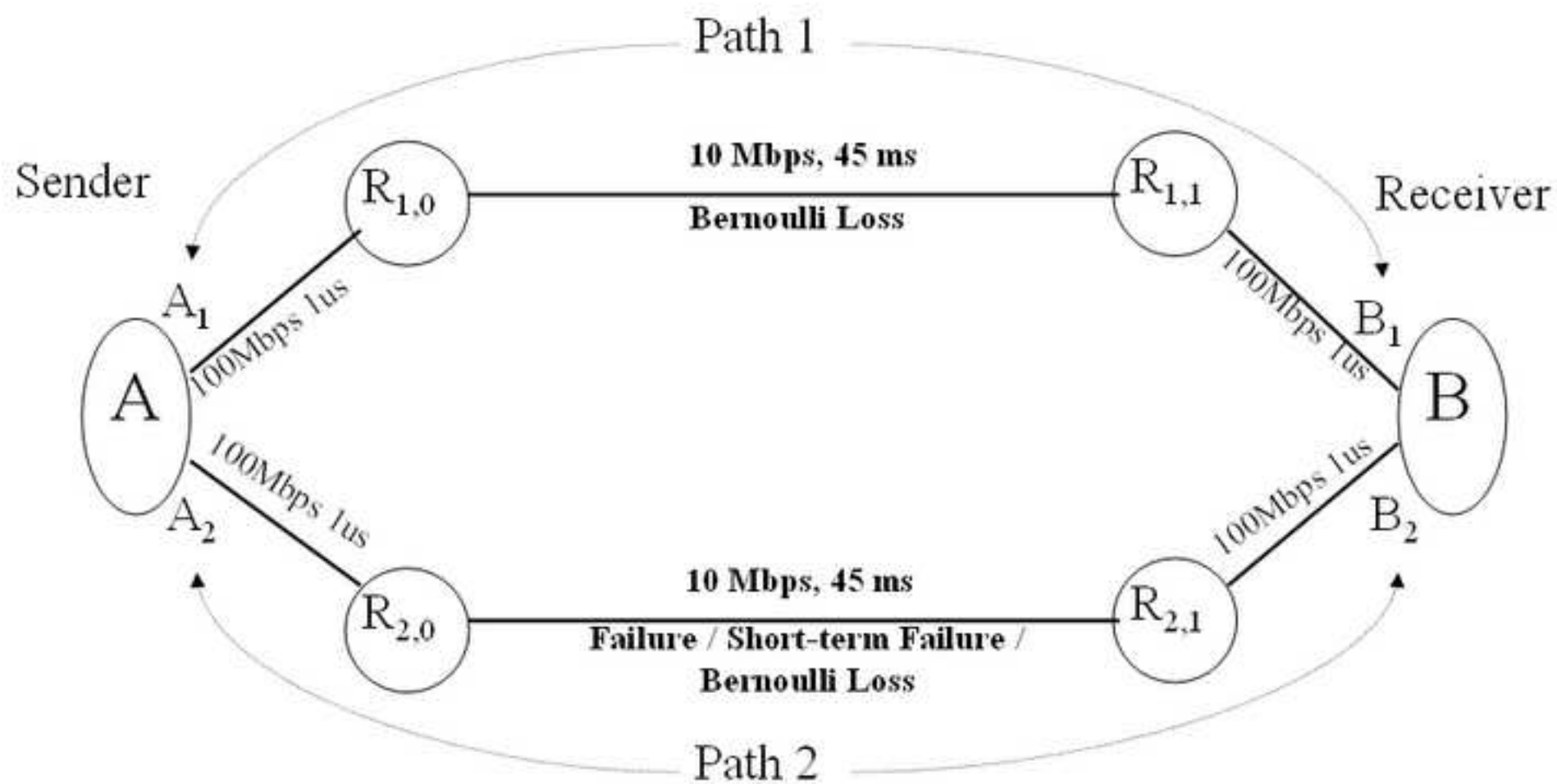


Figure(s)  
[Click here to download high resolution image](#)

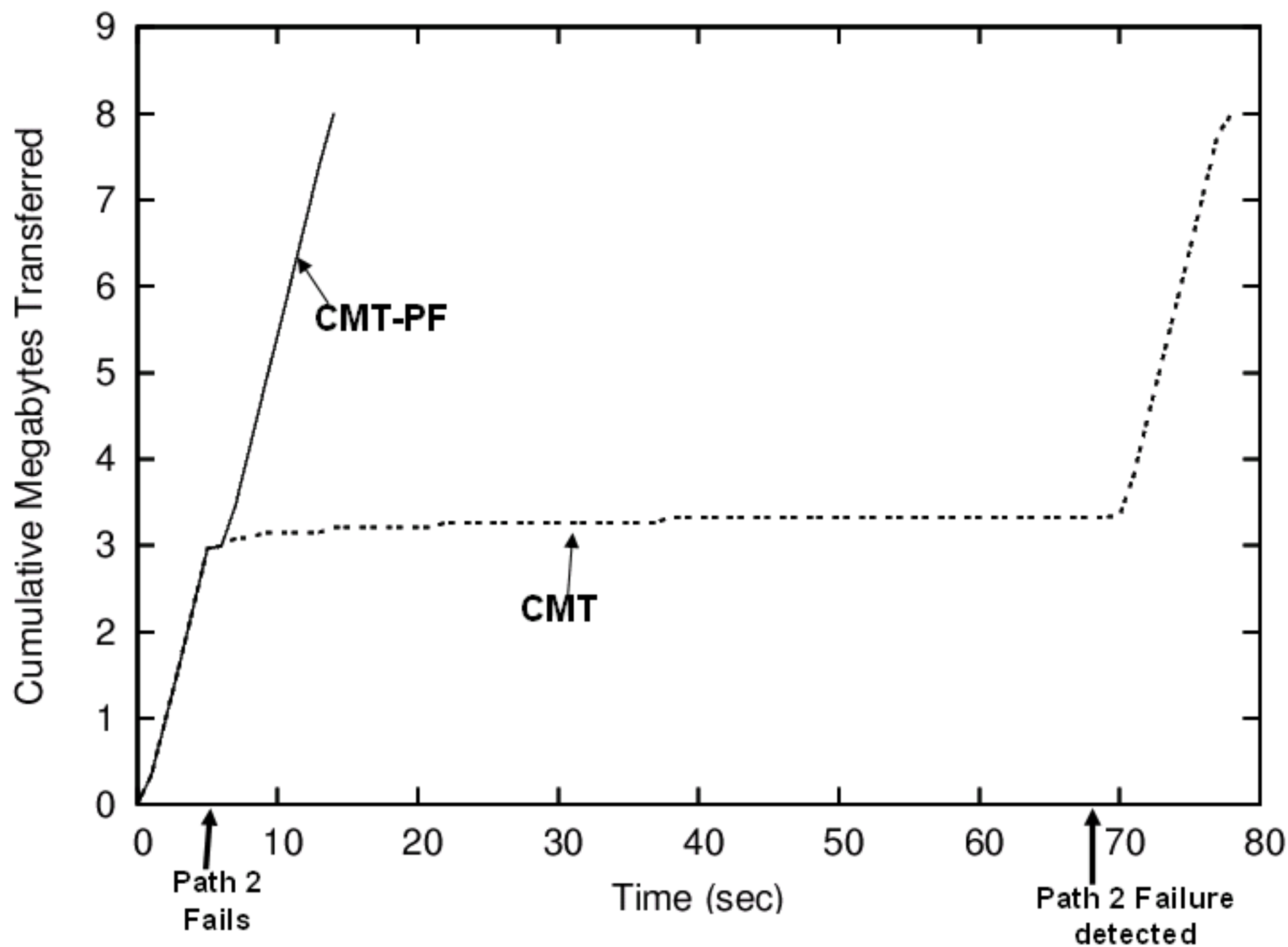


Figure(s)

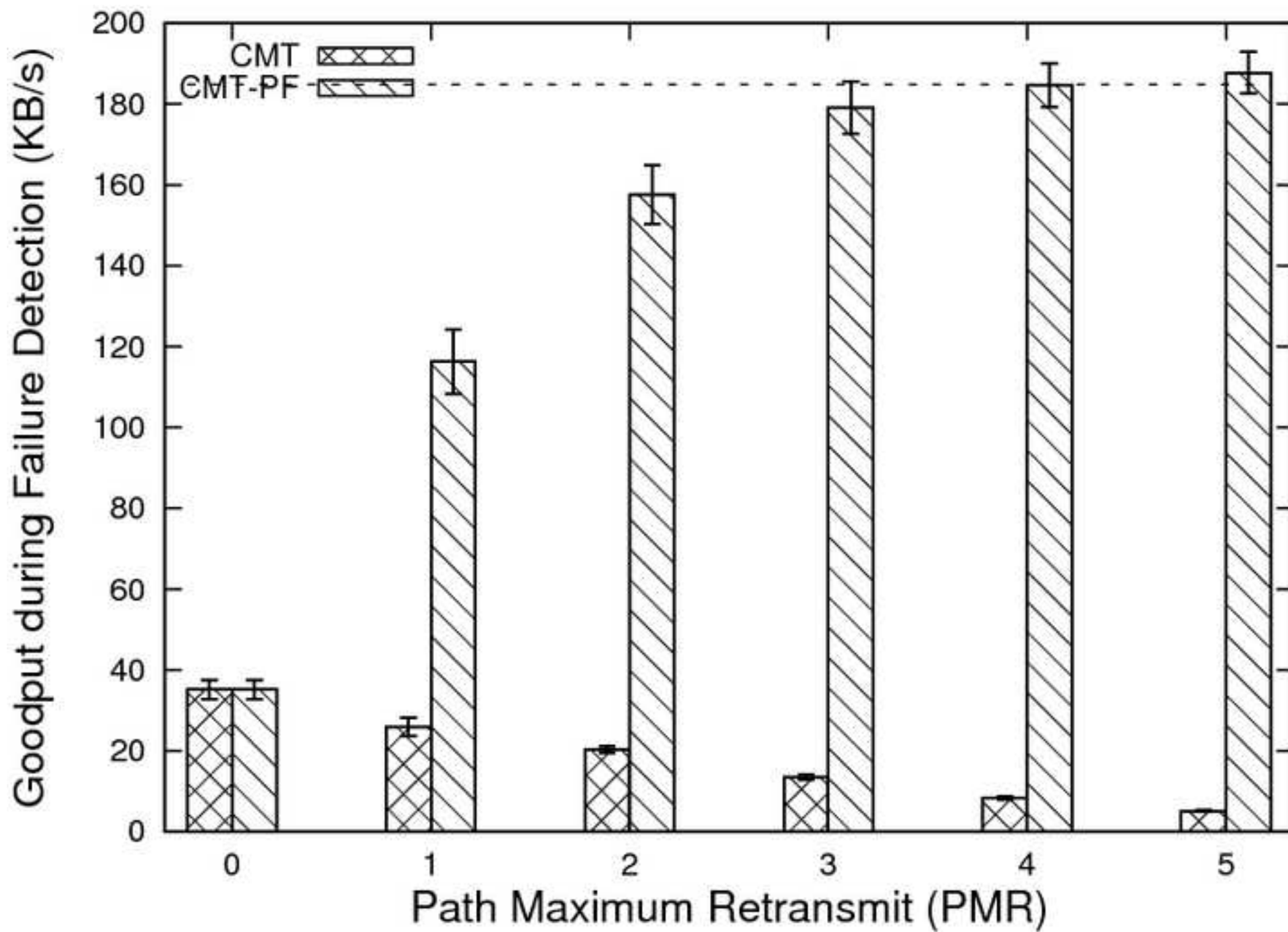
[Click here to download high resolution image](#)



Figure(s)  
[Click here to download high resolution image](#)

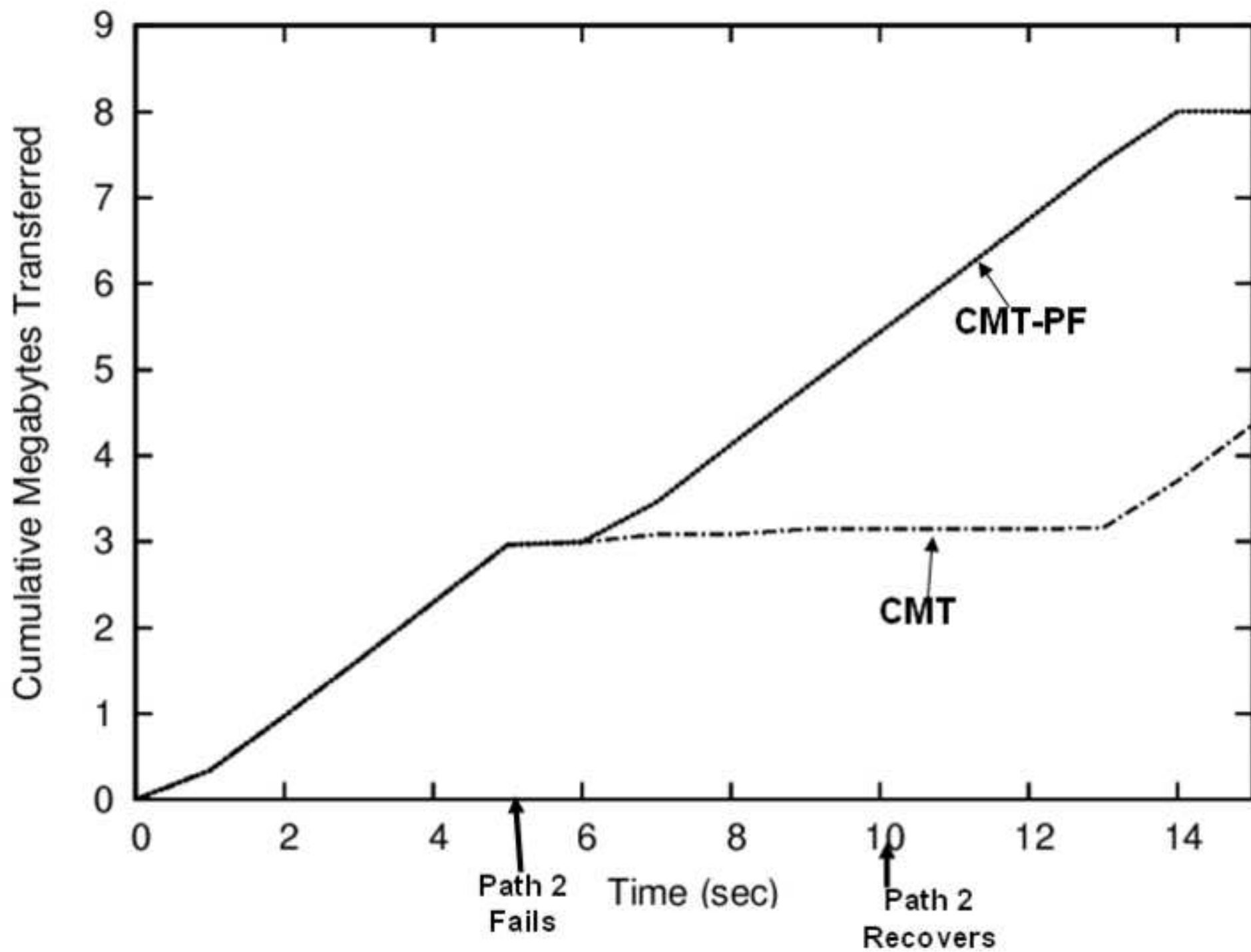


Figure(s)  
[Click here to download high resolution image](#)

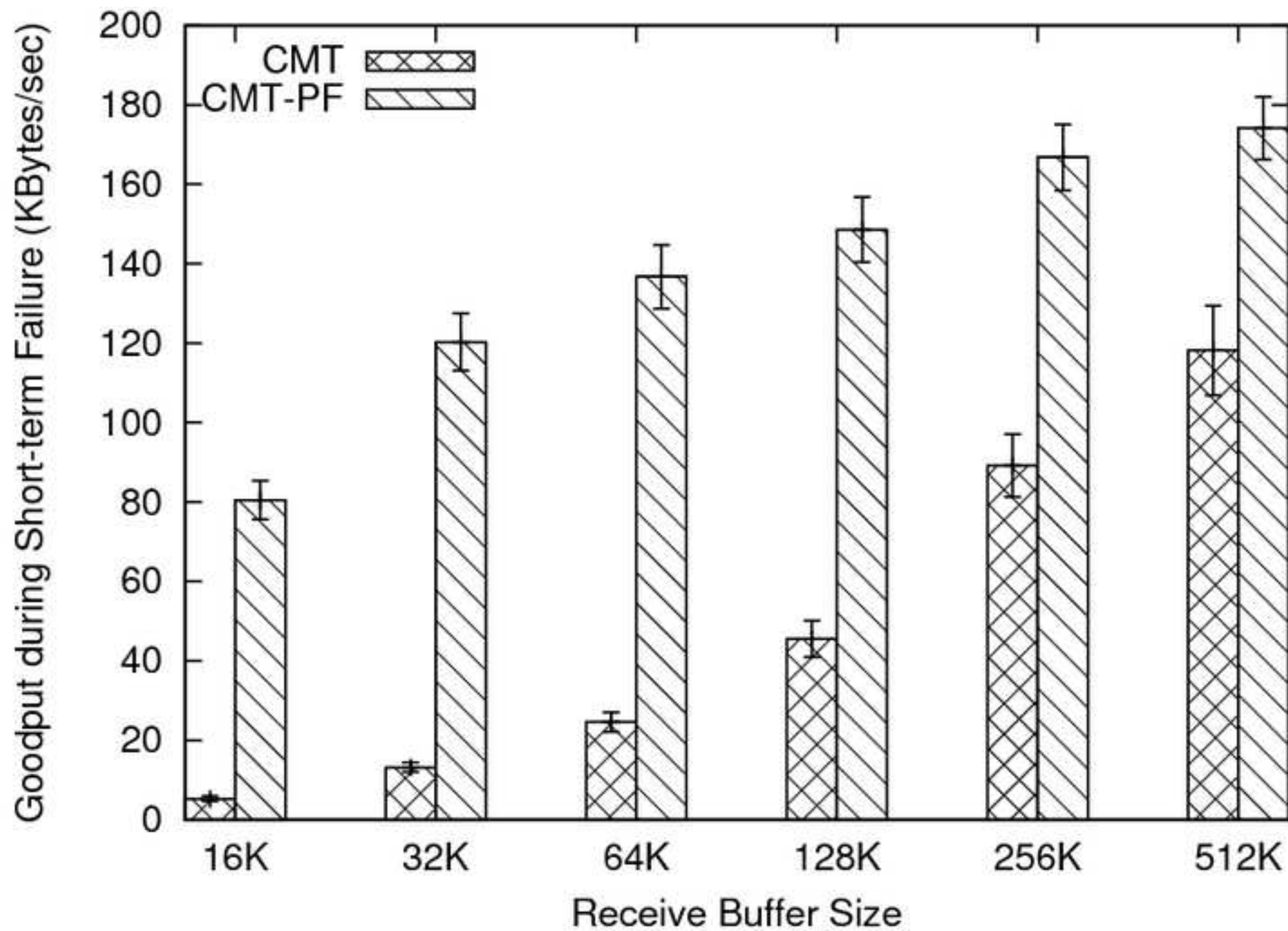




Figure(s)  
[Click here to download high resolution image](#)

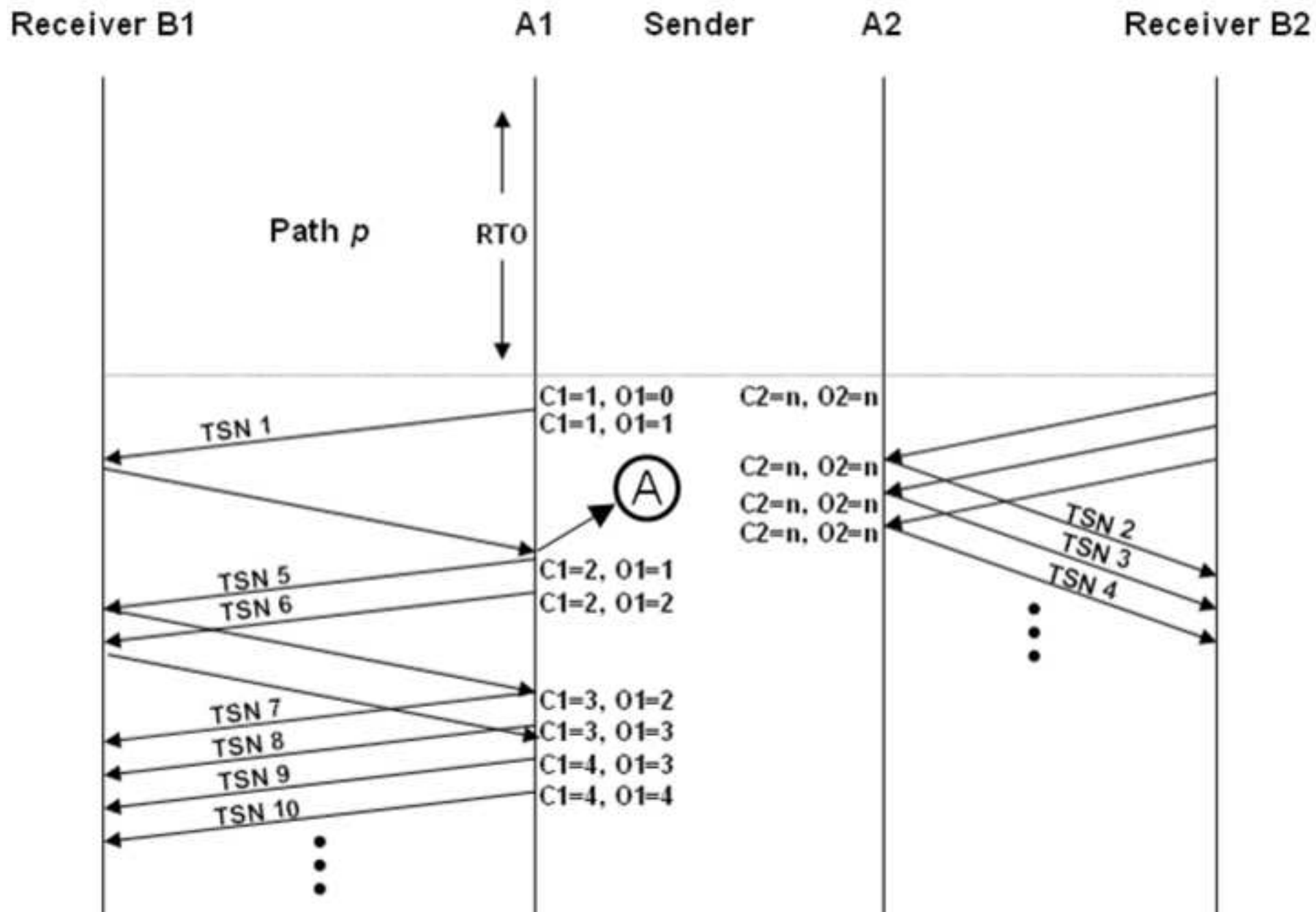


Figure(s)  
[Click here to download high resolution image](#)

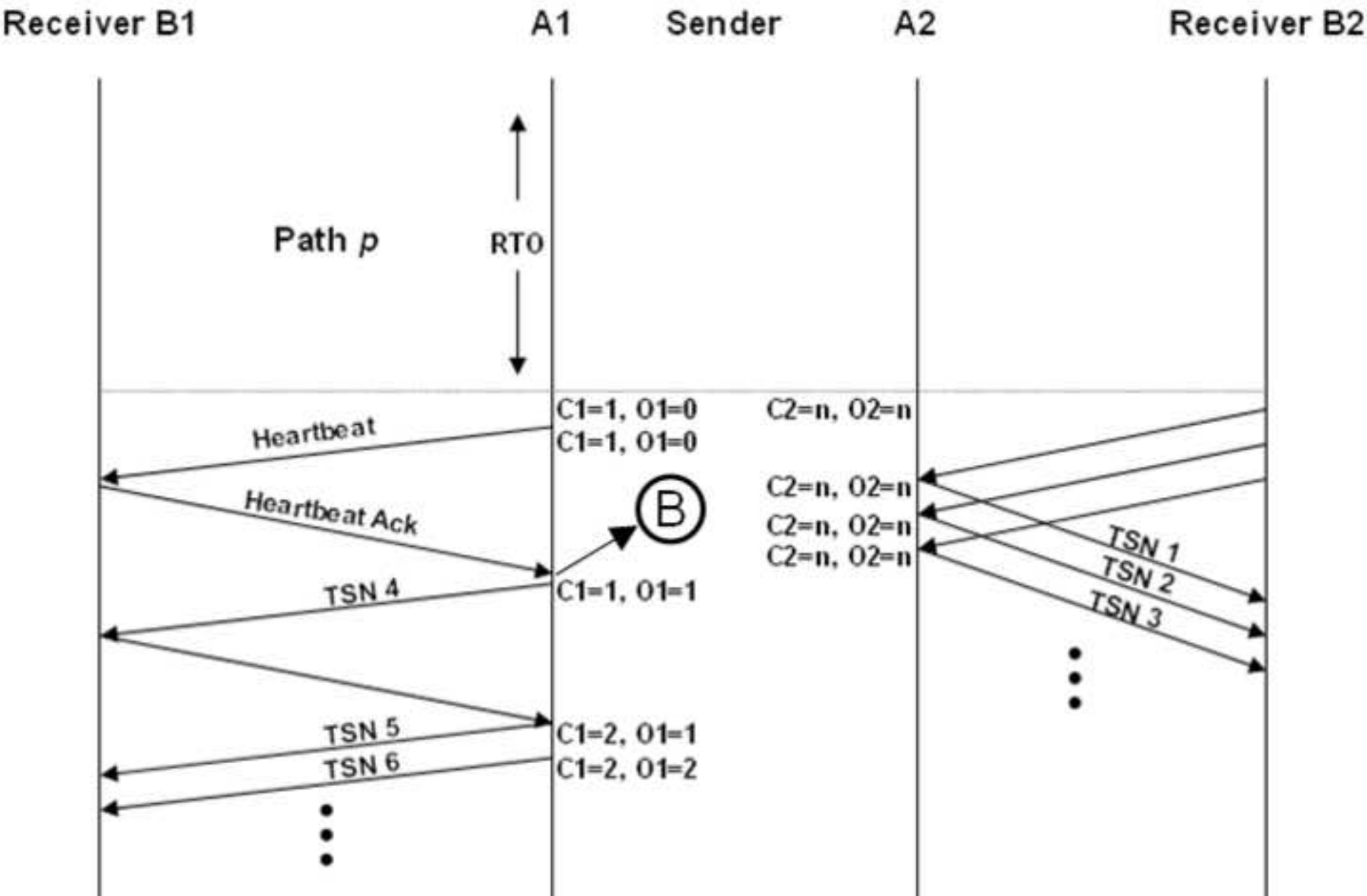




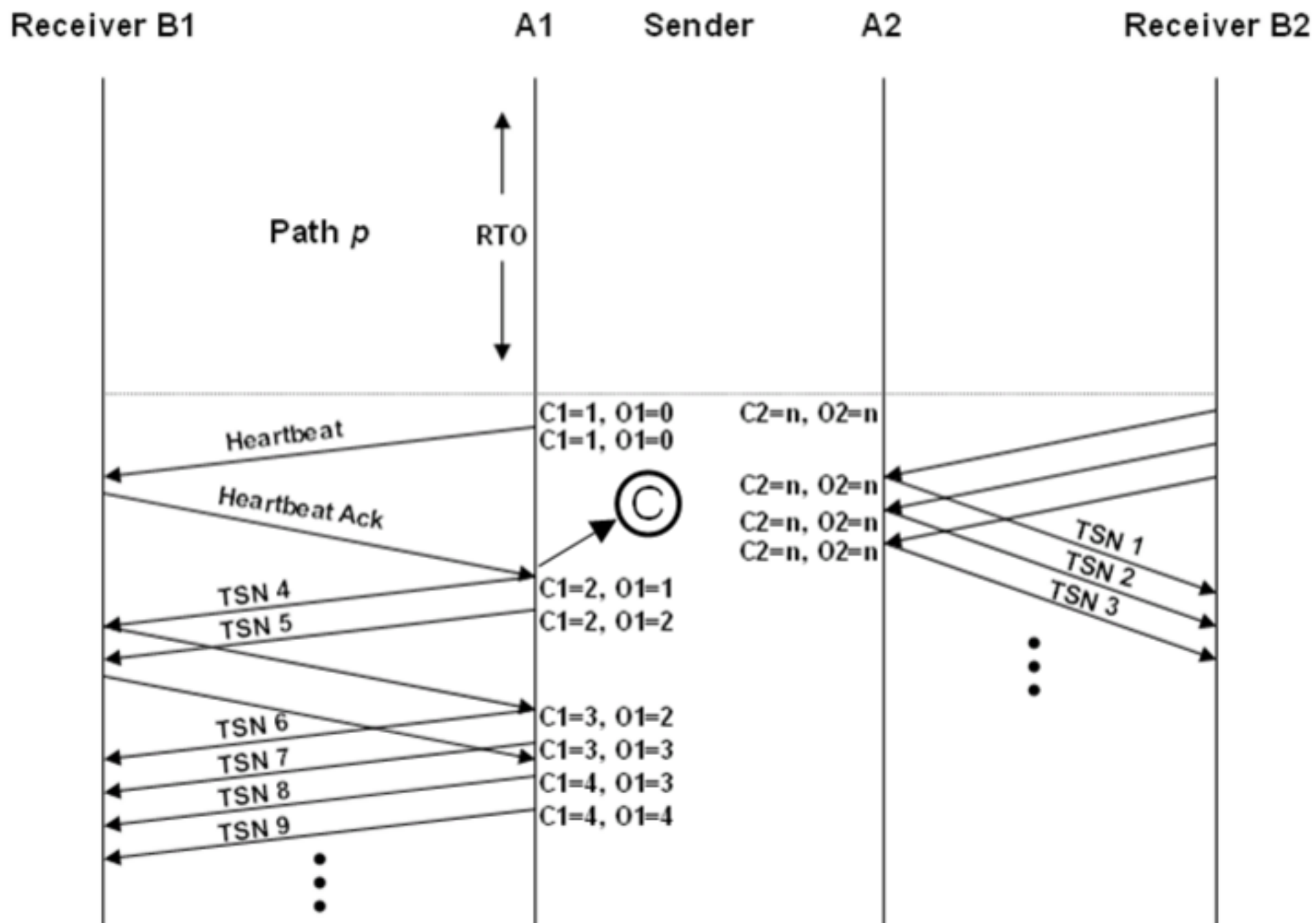
Figure(s)  
[Click here to download high resolution image](#)



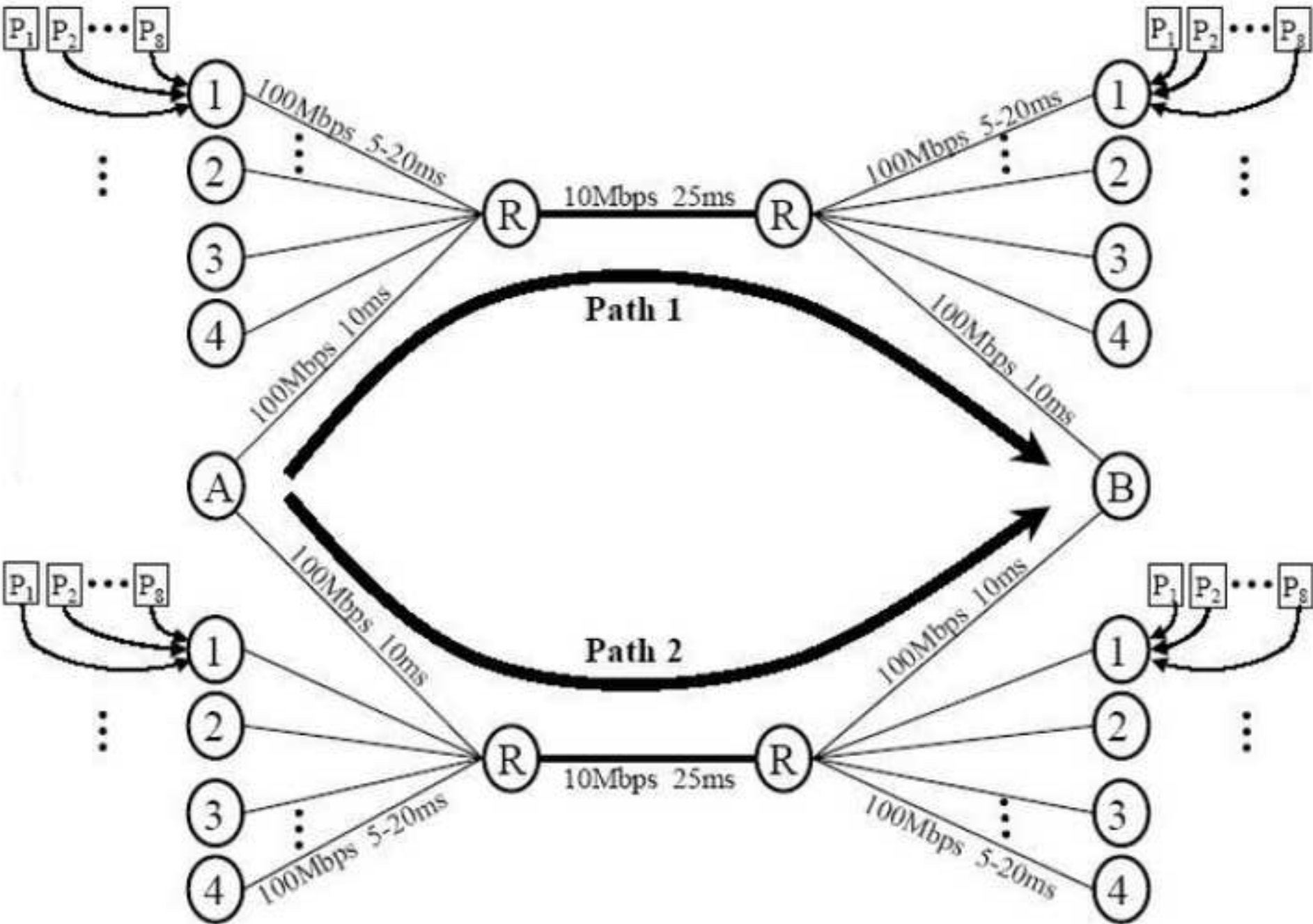
Figure(s)  
[Click here to download high resolution image](#)



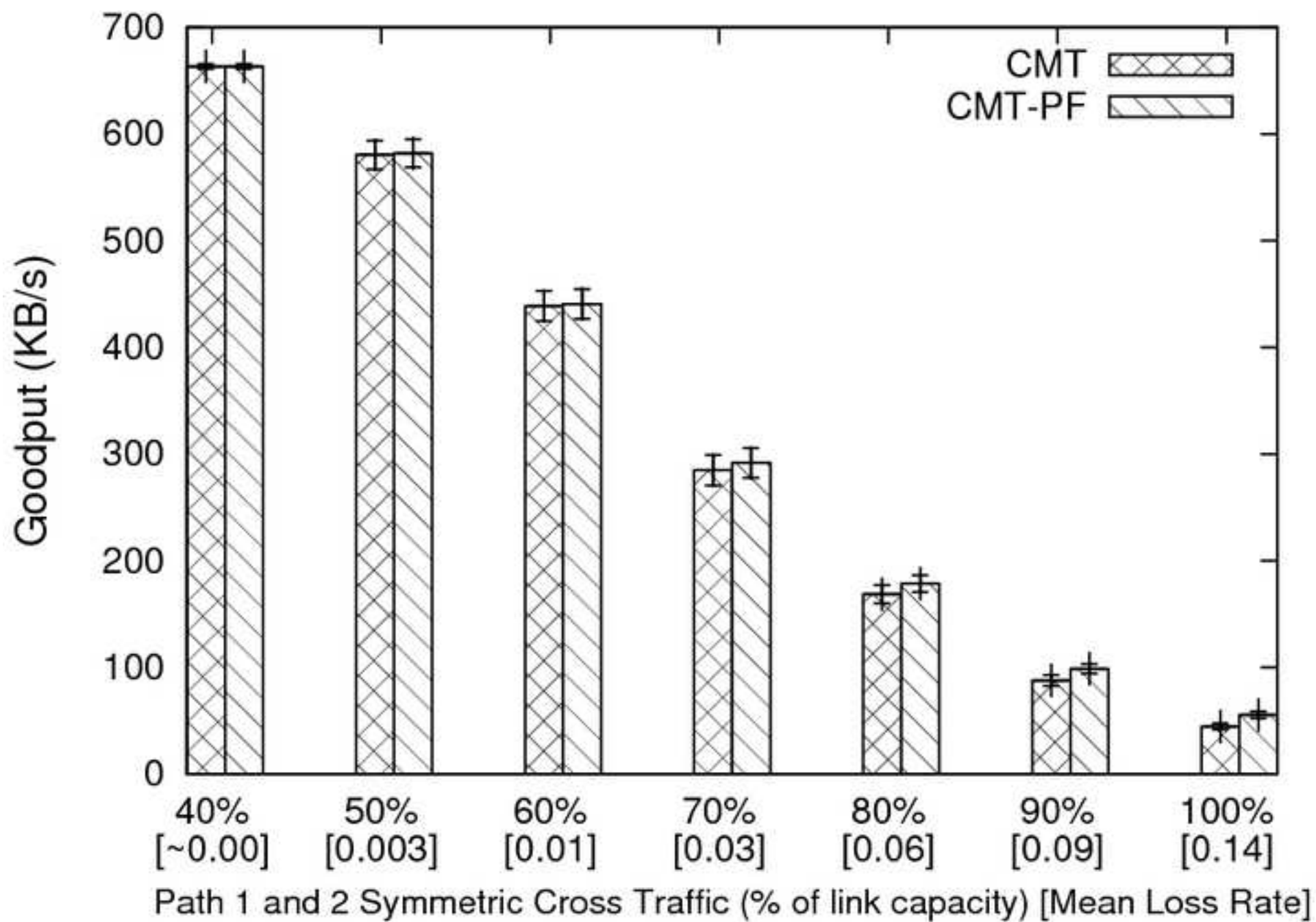
Figure(s)  
[Click here to download high resolution image](#)



Figure(s)  
[Click here to download high resolution image](#)

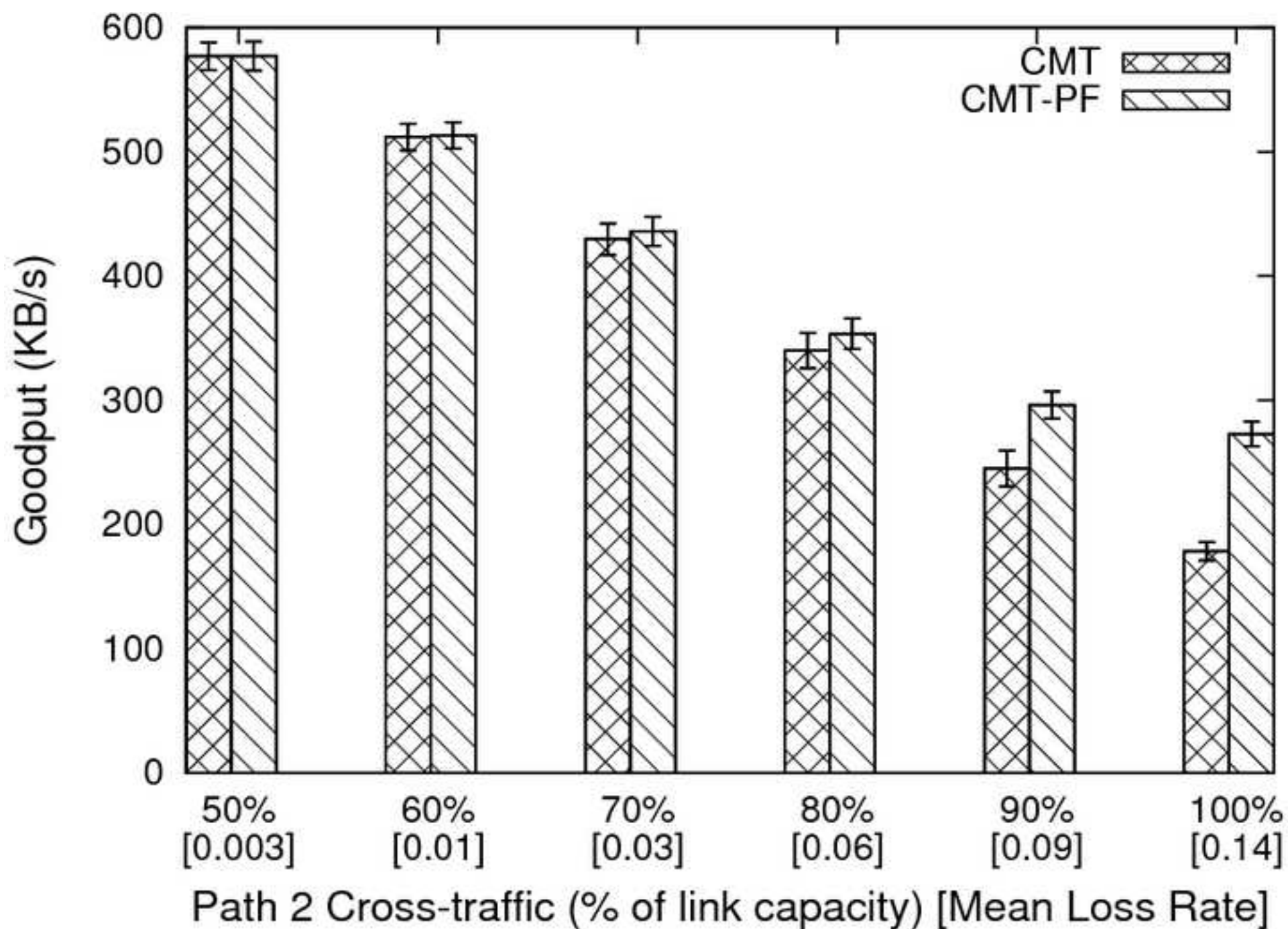


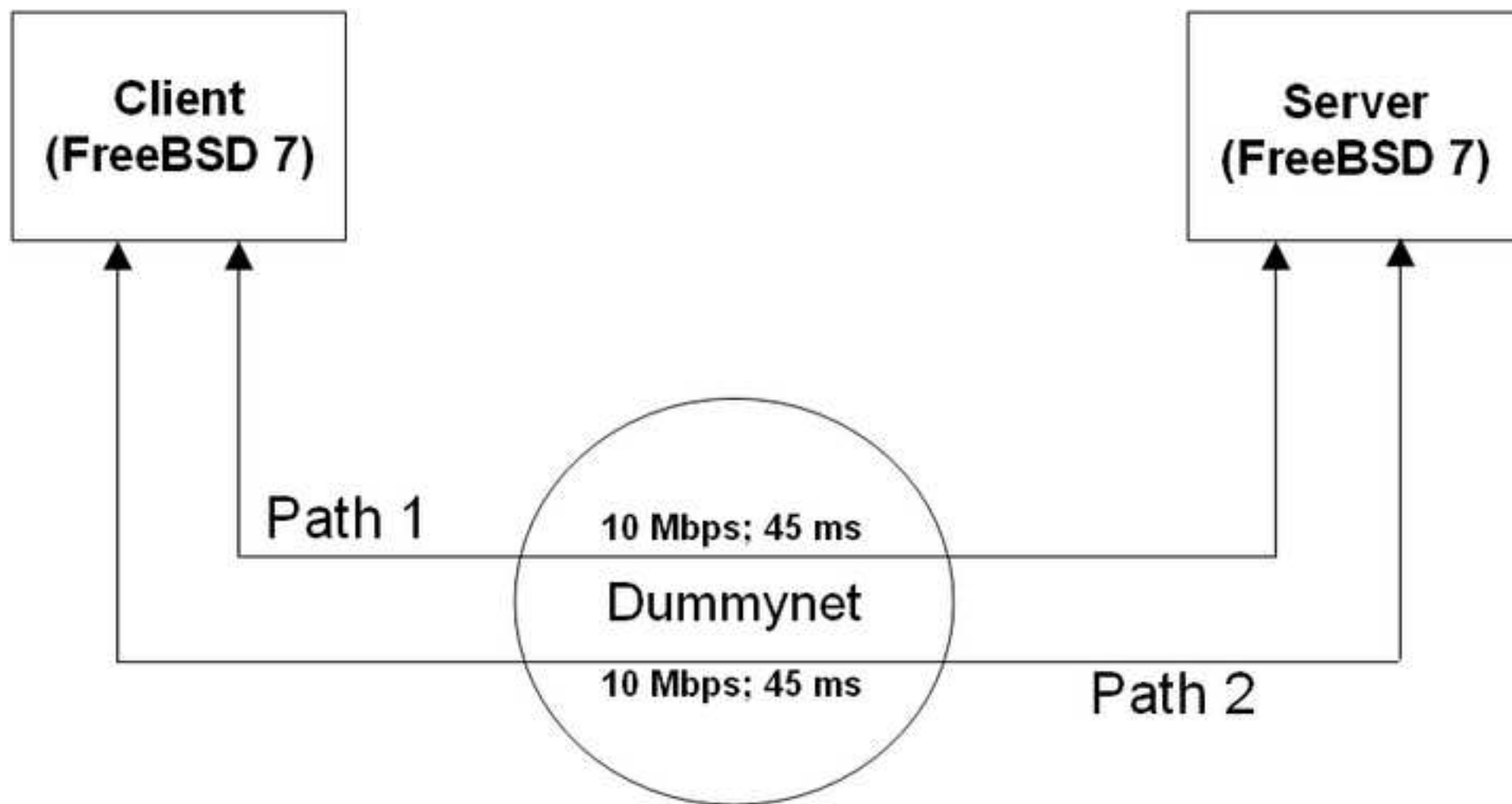
Figure(s)  
[Click here to download high resolution image](#)



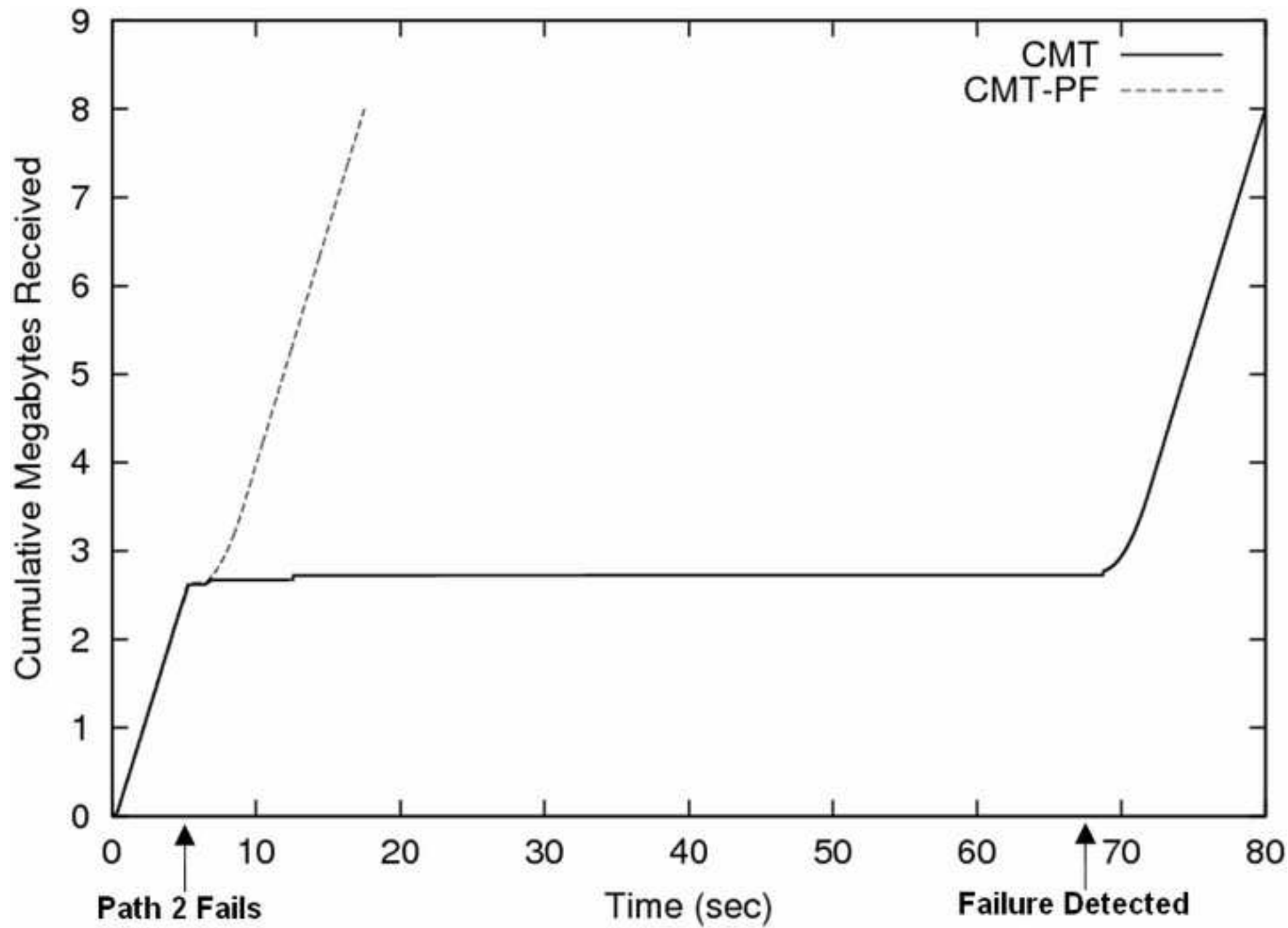


Figure(s)

[Click here to download high resolution image](#)

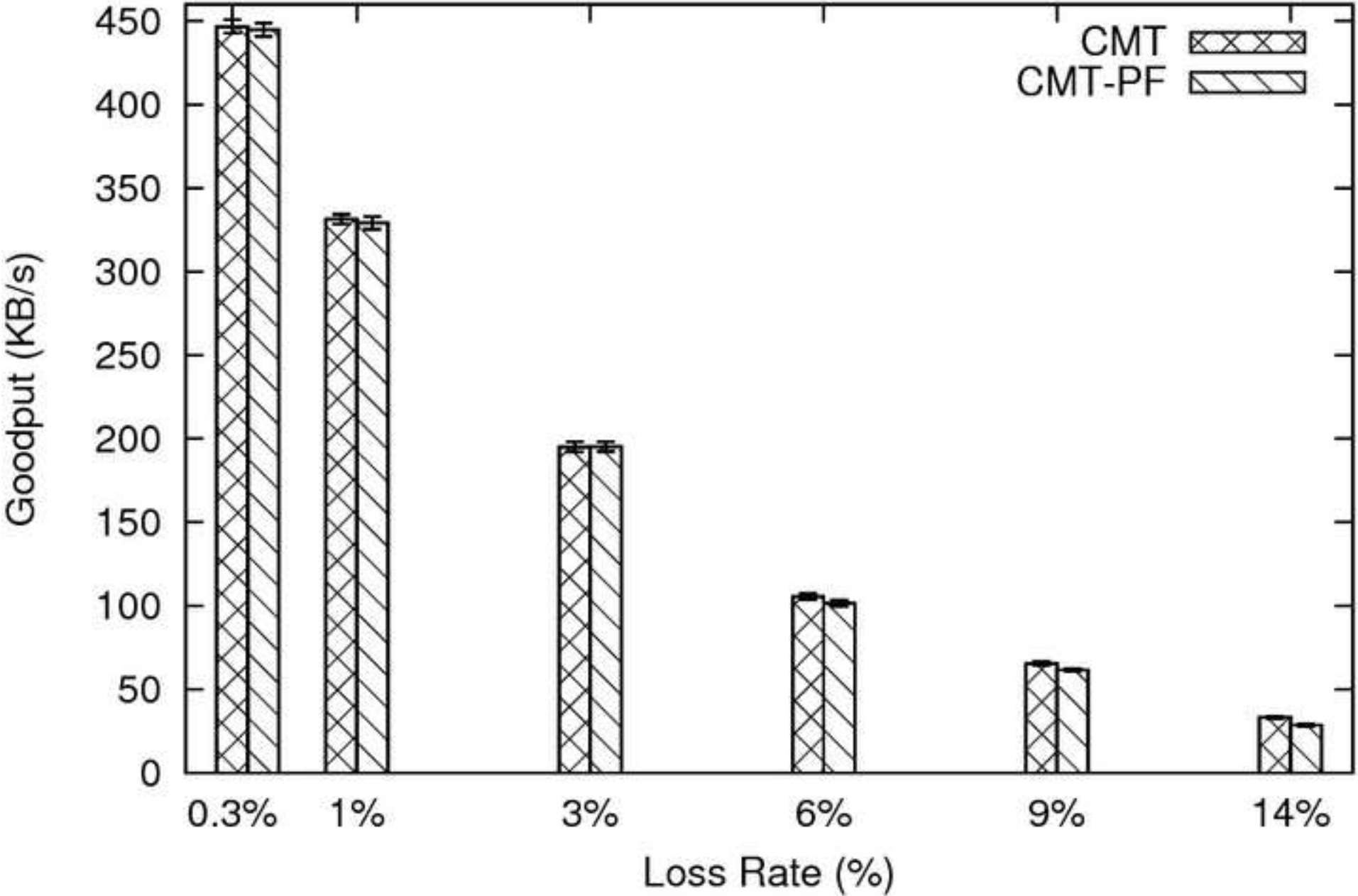


Figure(s)  
[Click here to download high resolution image](#)

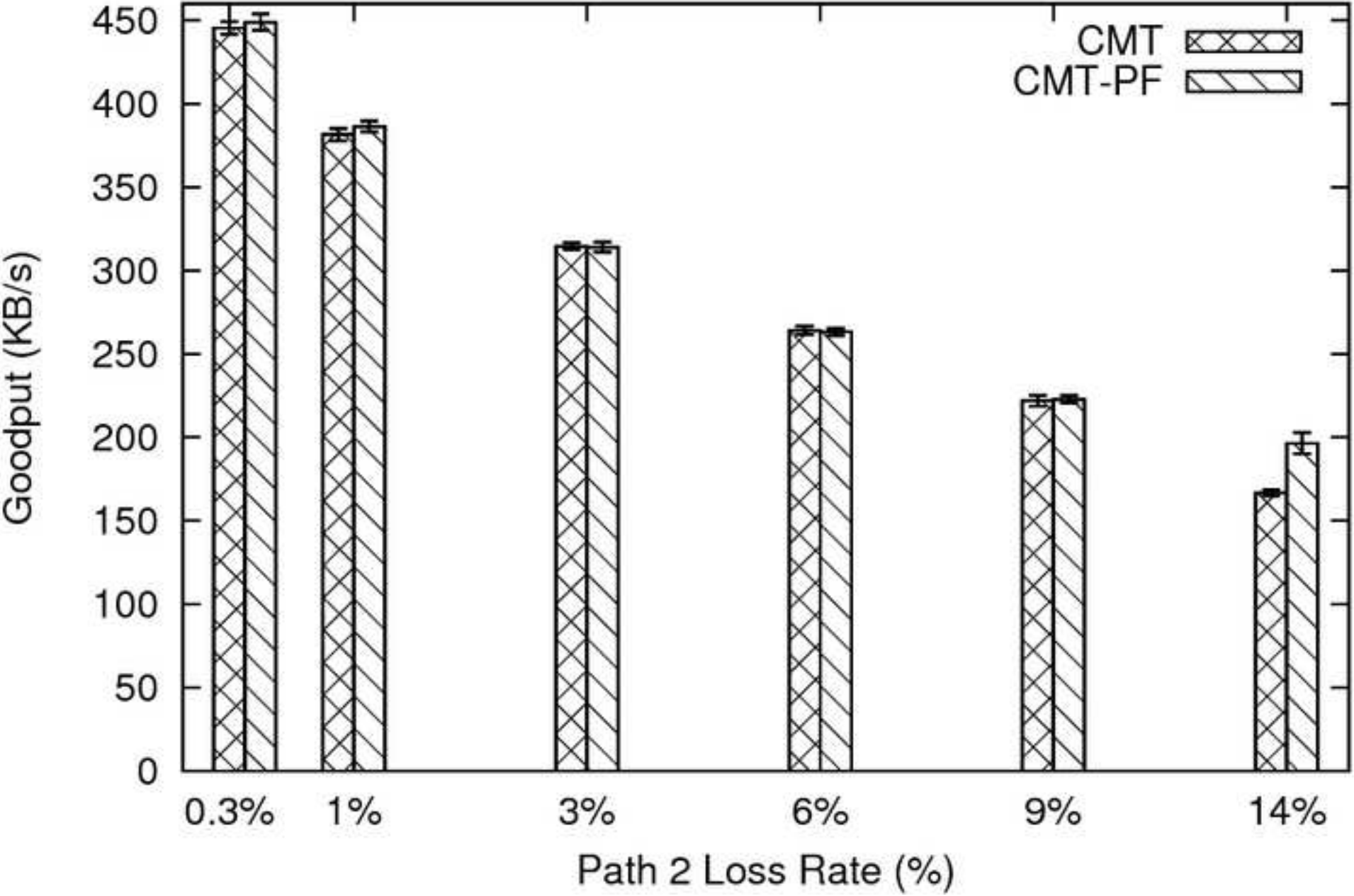




Figure(s)  
[Click here to download high resolution image](#)



Figure(s)  
[Click here to download high resolution image](#)



Variant	Path 2 Cross-traffic %	# of Consecutive Timeouts			
		2	3	4	5
CMT	70	0.49	0.02	0	0
CMT-PF		0	0	0	0
CMT	80	1.13	0.07	0	0
CMT-PF		0	0	0	0
CMT	90	3.73	0.60	0.09	0.02
CMT-PF		0.02	0.02	0	0
CMT	100	9.42	1.62	0.18	0.04
CMT-PF		0.04	0.04	0	0

Variant	Path 2 Cross-traffic %	Aggregate Transmissions		
		Path 1	Path 2	Path1/Path2
CMT	70	13,857	9,486	1.5
CMT-PF		14,002	9,344	1.5
CMT	80	15,530	7,902	2.0
CMT-PF		16,029	7,416	2.2
CMT	90	17,137	6,401	2.7
CMT-PF		18,153	5,362	3.4
CMT	100	18,093	5,508	3.3
CMT-PF		20,318	3,193	6.4