Building Multihoming Awareness into FAST TCP: A Simulation Analysis

M. Junaid Arshad, M. Saleem Mian

Department of Computer Science
University of Engineering and Technology
Lahore, Pakistan
Email: junaidarshad@uet.edu.pk; drmsaleem@uet.edu.pk
Tel: 92-42-9029260

Abstract: The dramatic evolution of high-speed network infrastructures and communication patterns (from point-to-point connections or multipoint-to-point or multihomed structures) means that an increasing number of Grid applications and distributed computing are demanding enhanced and diverse transport-level functionality. An end-to-end transport layer multihoming using concurrent multipath transfer (CMT) of data is an efficient approach that will be able to meet the demand for required bandwidth and connectivity, but the current multihomed-aware protocols (like SCTP or pTCP) are not designed for high-capacities and large-latencies networks, they often have performance problems transferring large data files over shared long-distance WANs. It has been shown that SCTP-CMT is more sensitive to receiver buffer (rbuf) constraints, and this rbuf blocking problem causes significant throughput degradation when multiple paths are used concurrently. In this research paper, we demonstrate the weakness of SCTP-CMT rbuf constraints and, we then identify that rbuf blocking problem in SCTP multihoming is mostly due to its loss-based nature for detecting network congestion. We present a simulation-based performance comparison of FAST TCP versus SCTP in high-speed networks. The objective of this article is threefold: to discuss rbuf blocking problems in SCTP-CMT; to describe some proposed transport protocols (like FAST TCP) that solve a number of throughput issues; and finally, to gain new insight into these protocols and thereby suggest avenues for future research. [The Journal of American Science. 2009;5(1):14-24]. (ISSN: 1545-1003).

Keywords: FAST TCP, SCTP-CMT, Multihoming, Transport Protocols, Receiver Buffer

1. Introduction

The Internet is a worldwide-interconnected computer network that transmits data by packet switching based on the TCP/IP protocol suite. The huge success of the Internet is achieved with improving designs and enriching protocols. While keeping pace with the advances in communication technology and non-stop demand for additional bandwidth and connectivity, the Internet continuously experiences changes and updates in almost all aspects.

As the application data has grown at a tremendous and accelerated rate in the past few years, by reason of continued advances in computing, communication, and storage technologies, combined with the development of national and global Grid systems, thus requiring a proper transport protocol with some enhanced features, in order to realize the vision of distributed collaboration, such as transfer large amount of data and access remote resources (computing facility and storage space) across a high-speed wide area networks.

In this regard, first it would be significant to address the deficiencies in the current loss-based congestion control protocols (like TCP (Allman et al., 1999) and SCTP (Stewart et al., 2000; Iyengar et al., 2004)) when doing large data transfers. The key challenge we face, and intend to overcome, is that the current loss-based congestion control protocols do not scale to this regime i.e., they are ill suited for the future high-speed networks, which motivates the design of new distributed algorithms for large bandwidth-delay product networks (i.e., FAST TCP (Jin et al., 2003)). The congestion control algorithms used by TCP and SCTP are based on RFC 2581 (Allman et al., 1999) and RFC 2960 (Stewart et al., 2000), respectively. Their key mechanisms are Slow Start and congestion avoidance phase. All these congestion control algorithms exploit the additive increase and multiplicative decrease (AIMD) paradigm (Jacobson, 1988), which additively increases the congestion window (cwnd) to grab the available bandwidth and suddenly decreases the congestion window when network capacity is hit and congestion is experienced via segment losses (thus obtaining a poor utilization of the bottleneck link).

But this limitation is avoided by the delay-based approach of FAST TCP; it is one of such algorithms that are designed for high-speed long-distance networks, it aims to rapidly stabilize networks into steady, efficient and fair operating points. FAST TCP congestion control mechanism reacts to both queuing-delay and packet loss, since packet loss only provides a single bit of information about the congestion level, whereas delay is a continuous quantity and in principal provides more information about the network (which in turn provides efficient link utilization).

Another approach that is used for improving the end-to-end throughput and link redundancy is a transport layer multihoming (Ohta, 2002). Multihoming is the ability of a host or site to access remote destination via more than one upstream connection, usually from different providers. SCTP supports concurrent multipath transfer (CMT) (Iyengar et al., 2004) of data between the multihomed hosts, but the existing TCP (Allman et al, 1999) and its variant (such as FAST TCP) do not support multihoming. Wide spread use of multihoming was infeasible during the early days of the Internet due to cost constraints; today, network interfaces have become commodity items. Cheaper network interfaces and cheaper Internet access motivate content providers to have simultaneous connectivity through multiple ISP's for added flexibility and fault tolerance.

Thus, we believe that a transport protocol right for large data transfers over high-speed networks such as in Grid computing should have at least the following properties:

- A transport protocol that has both high performance and robustness, not only under local transfer of small files but also in high-speed long-distance transfer of extremely large files, along with the capability for independent up-gradation of its components.
- A transport protocol that has the ability for transferring of data through concurrent multipath using multihoming or some other means.
- A transport protocol that could run on the same Internet infrastructure (with minimum modifications) we have today.

The protocols performance measurements are included in many papers (Floyd, 2003; Kelly, 2002; Jin et al., 2003; Bullot et al.) proposing modifications to the standard TCP AIMD congestion control algorithm, but with the large deployment of optical networks and grid applications, there is still continuing studies to evaluate the current set of high-speed transport protocols to determine the best overall protocol.

In this study, our focus is not to choose a winner since many of the protocols are under intensive and rapid progress. Rather, we hope to achieve in-depth understanding of the environments in which various protocols would perform well, and the causes of poor performance.

The remainder of this paper is organized as follows. In Section 2, first we present an overview of the protocols (SCTP and FAST TCP) and then we present our experimental setup and progressively analyze the behavior of FAST TCP compared to SCTP by using a simple network topology in ns-2 (VINT Project, Network Simulator). In Section 3, we delineate the rbuf blocking problem in SCTP-CMT (Iyengar et al., 2005) and identify the dilemma degrading its performance in the presence of a bounded receive buffer, which is evaluated in Section 3.2.1 through simulations. Finally in Section 4, we present the conclusions of this work.

2. FAST TCP vs. SCTP in High-Speed Networks

SCTP (Stewart et al., 2000) is a new reliable session-oriented transport protocol operating on top of the Internet Protocol (IP). SCTP and TCP use the basic AIMD algorithm to adjust their windows sizes.

These loss-based protocols achieve congestion control successfully in the current low speed networks. However, they can perform poorly in networks with high bandwidth-delay product (BDP) paths; because the AIMD algorithm, being very conservative, is not designed for large window size flows. First, it takes too long time for a large window size user to recover after a back-off and the bandwidth is not effectively utilized (Floyd, 2003), secondly it detects congestion only when the packet is lost in the network.

FAST TCP (Jin et al., 2003) is a modification to the standard TCP congestion control algorithm for high-speed long-distance connections. The delay-based congestion control algorithm of FAST TCP is fundamentally different from AIMD; it uses queueing delay for congestion control and its advantage over loss-based approach is small at low speed, but decisive at high speed.

The window update algorithm of FAST TCP determines the right congestion window size based on the current estimation of queueing delay whenever reliable RTT measurements are available (i.e., qdelay = avgRTT – baseRTT).

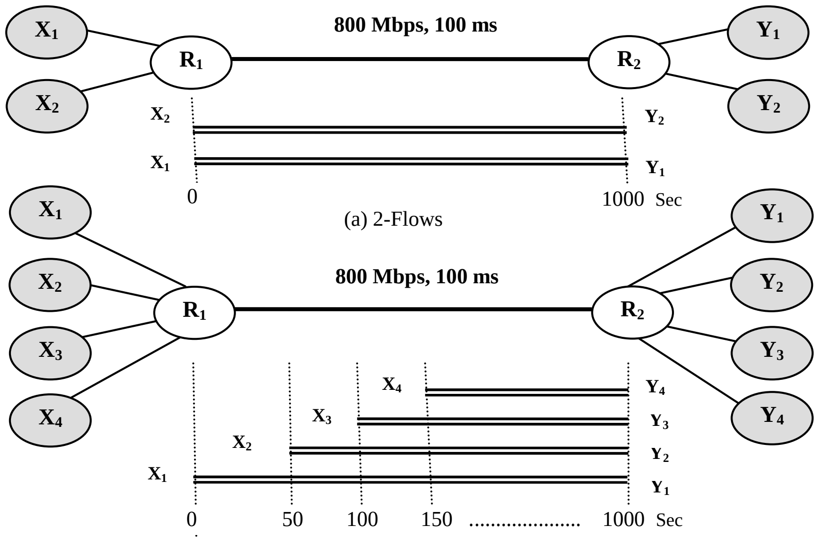
FAST TCP periodically updates the congestion window based on the average RTT and average queueing delay provided by its estimation component, according to (1) as described in (Jin et al., 2003).

$$w \leftarrow \min\left\{2w, (1-\gamma)w + \gamma\left(\left(\frac{baseRTT}{RTT}\right)w + \alpha(w, qdelay)\right)\right\} \qquad (1)$$

In the next subsections, we describe the setup of our experimental comparison of the protocol's performance in terms of application throughput, queuing-delay and packet loss during file transfers.

2.1 Experimental Setup

In this section, we briefly describe the experiments carried out to compare the performance of FAST TCP and SCTP protocols in single-homed high-speed networks. We used *ns-2* network simulator as the basis for our protocols comparison and performance evaluation. We used FAST TCP simulator module for *ns-2* (Cui et al.), version 1.1 (SACK introduced) and for SCTP, we used the University of Delaware's module (Caro et al.).
We conducted two set of simulations to compare the protocol's performance based on the network topology: i) with the two (flows) sender and receiver pairs ($X_1 \leftrightarrow Y_1, X_2 \leftrightarrow Y_2$) shown in Figure 1(a) and, ii) with four (flows) sender and receiver pairs ($X_1 \leftrightarrow Y_1, X_2 \leftrightarrow Y_2, X_3 \leftrightarrow Y_3, X_4 \leftrightarrow Y_4$) shown in Figure 1(b).



(b) 4-Flows

Figure 1: Network topology used in the simulations with active periods of the flows

To see the difference between FAST TCP and SCTP, we simulated same link with different number of flows having the bottleneck link capacity of 800Mbps with drop-tail queueing, and the buffer size of 3000 packets with a fixed packet length of 1500 bytes. A router monitor's module recorded the queue size every 0.2 second and packet loss was set to 0%. We ran each set of simulations for 1000 seconds and data transfer was done using FTP. For FAST TCP, in all of our experiments the parameter value alpha (α) was set to 200 packets for each flow.

2.2 Results and Discussions

In this section, we present a performance comparison of both the protocols through simulation results and discuss the protocols behavior. In the first set of simulations, there were two flows (sources) sharing a router with a common propagation delay of 100ms, which started and terminated at the same times as shown in Figure 1(a). In the second set of simulations, there were four flows with the same propagation delay of 100ms, which joined and departed according to the schedule in Figure 1(b).

We ran each set of simulations under each of two protocols (SCTP and FAST TCP) and presented the aggregate throughput, congestion window, the queue occupancy and the total number of packets lost at the bottleneck. Figure 2 and Figure 3 show the simulation results for SCTP and FAST TCP, respectively, when two flows were used. Similarly, the simulation results in Figure 4 and Figure 5 show the aggregate throughput, congestion window, the queue occupancy and the total number of packets lost at the bottleneck for SCTP and FAST TCP, respectively, when four flows were used.

SCTP's trajectories in Figure 2 (2-flows) show that during the initial slow-start phase, there is no a priori knowledge of the available bandwidth that can be used to stop the exponential growth of the SCTP's windows. Thus, we see that SCTP increases its congestion windows until the available bandwidth is exceeded and, it uses more and more buffers in the router until it losses packets by overrunning the bottleneck queue. All these losses experienced are due to congestion at the routers; no loss is due to bit errors (i.e., loss in our simulations (only) occurs due to congestion, we do not set the loss rate).

We also observe that, as more number of SCTP competing sources join the network, stability becomes worse for this loss-based protocol that produce more oscillations in its congestion windows and queue size, and increase packet loss in the network as shown in Figure 4.

On the other hand under similar conditions, FAST TCP consistently outperforms SCTP in terms of throughput, stability with zero packet loss at the bottleneck. For FAST TCP, each source tries to maintain the same number of packets in the queue in equilibrium and each competing source get an equal share of the bottleneck bandwidth as shown in Figure 3 and Figure 5. These figures show clearly that FAST achieves a better aggregate throughput, since they can keep the system around full utilization.

As a comparison, SCTP's flows (Figure 2 and Figure 4) purposely generate packet losses and oscillate between full utilization and under utilization. This fact can be explained by the following description in Figure 4 (4-flows), as the first SCTP flow $X_1 \leftrightarrow Y_1$ was started at time zero, during the initial slow-start, there was no a prior knowledge of the available link bandwidth, so this exponential growth of the window stop too early (when the network is far from congestion) and it will take a long time by using the linear increase to arrive at the optimal congestion window size (if there was only one flow), but at time 50 and 100 seconds the two more SCTP flows $X_2 \leftrightarrow Y_2$ and $X_3 \leftrightarrow Y_3$ joined the network and started their windows increasing too, as a consequence, increase the sending rates blindly (when the network is close to congestion) and the congestion windows will frequently grow until the available bandwidth is exceeded.

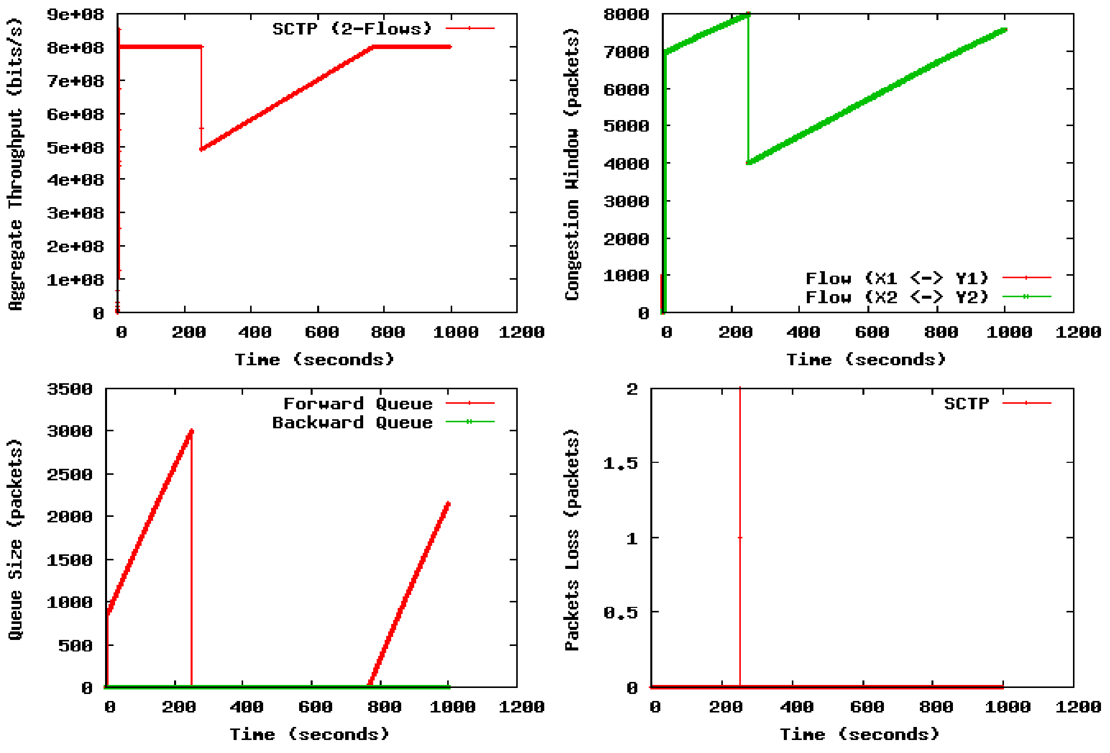


Figure 2: SCTP's trajectories with 2-flows

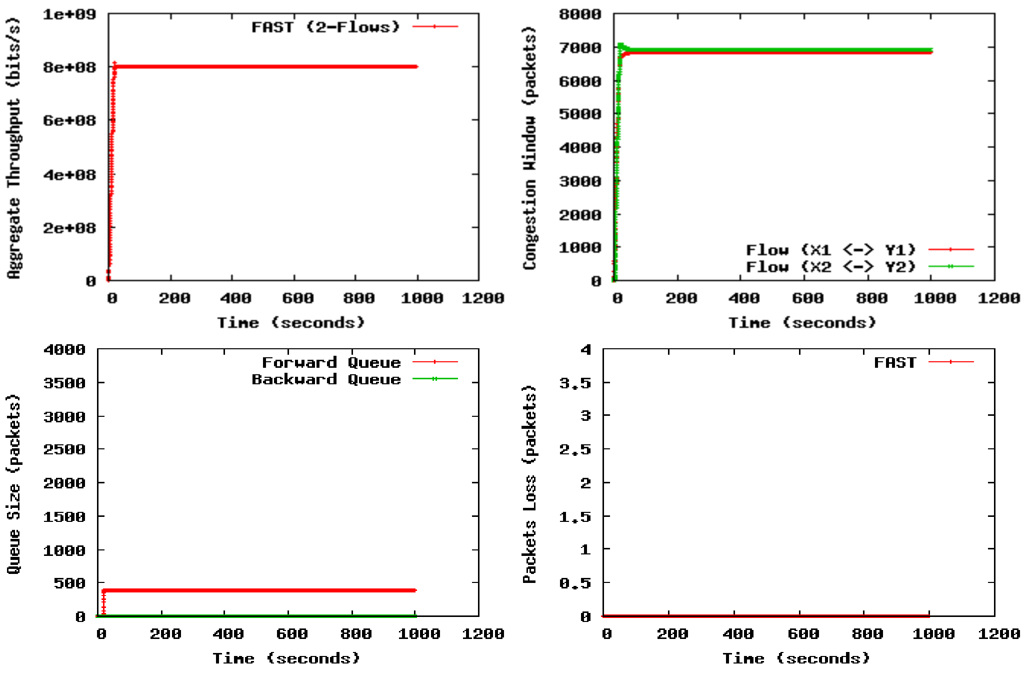


Figure 3: FAST TCP's trajectories with 2-flows

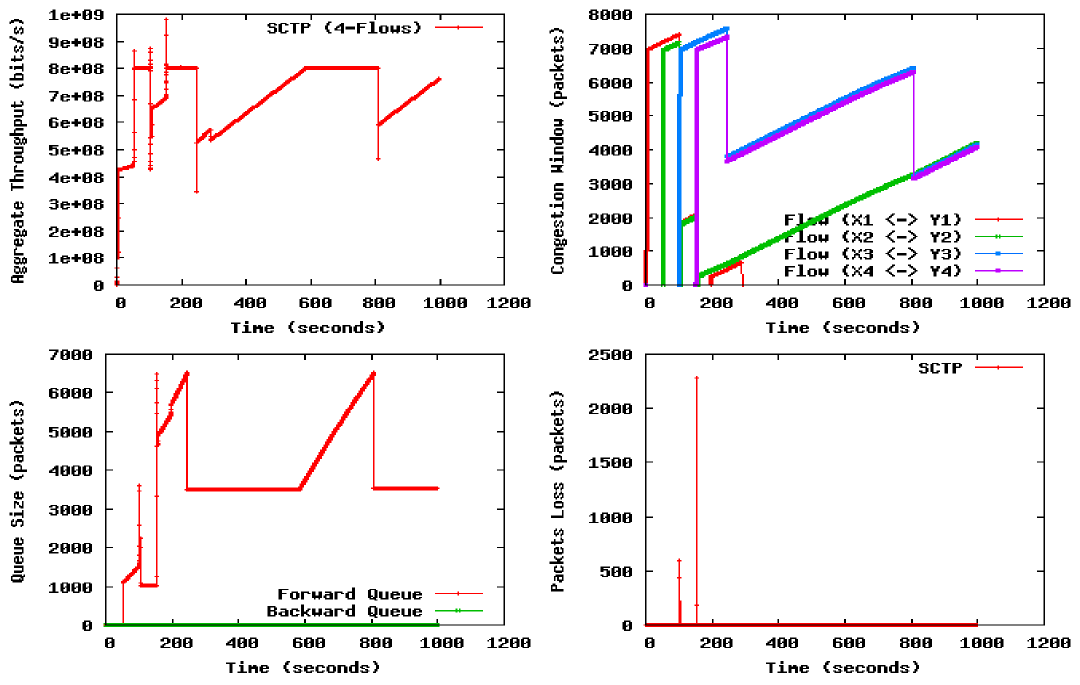


Figure 4: SCTP's trajectories with 4-flows

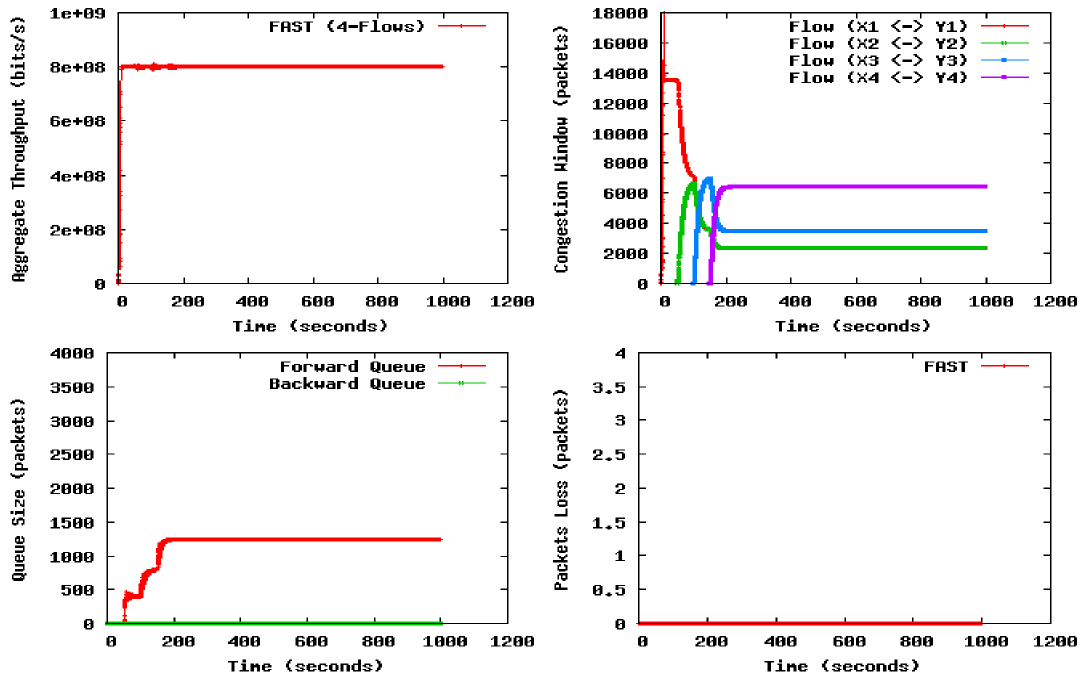


Figure 5: FAST TCP's trajectories with 4-flows

As the windows size increase we wait for the aggregate throughput to also increase, but the aggregate throughput cannot increase further than the available bandwidth, this is because, beyond this point any increase in the window size only results in the segments taking up buffer space at the bottleneck router. And hence it then started packets drop at the network when the queue size exceeded the available router buffer capacity as shown in Figure 4 in the region between 0 and 102.8 seconds. Similar, cwnds reductions are observed at time 200 seconds (when last SCTP flow $X_4 \leftrightarrow Y_4$ at time 150 seconds joined the network) and later at time 800 seconds.

3. Receive Buffer Blocking in SCTP-CMT Multihoming

3.1 Problem Description

SCTP is an IETF standards-track protocol that natively supports multihoming at the transport layer. SCTP is relatively new; it has not yet been widely deployed in the Internet despite its many advantages over standard TCP and UDP, though the research on extending SCTP to support concurrent multipath transfer using transport layer multihoming to increase the association bandwidth is still in progress (Iyengar et al., 2004).

In SCTP-CMT, the receiver maintains a single rbuf which is shared across all the paths (flows) and it consumes data only in sequence, irrespective of the destination address they are sent to.

An SCTP sender's sending rate is bounded by both the peer receiver-window (rwnd) and the pertinent destination's cwnd. It has been shown in (Iyengar et al., 2005) that if two paths are used for CMT: (i) the lower quality (i.e., higher loss rate) path degrades overall throughput of a receiver buffer constrained CMT association by blocking the rbuf or peer-rwnd and, (ii) it also degrades performance increasingly with increasing difference in end-to-end delay combinations on the paths used for CMT and, (iii) it is more sensitive to rbuf constraints in environments with shorter end-to-end delay.

This rbuf blocking problem causes significant throughput degradation when multiple paths are used concurrently. Moreover, larger the difference between the paths (due to delays and/or loss-rates differences) also increases the rbuf blocking in SCTP-CMT.

3.2 Impact of Receive Buffer Blocking on CMT due to Network Congestion-Based Losses

When one path used in CMT experiences failure (due to congestive losses or non-congestive losses), data outstanding on the failed path has to be recovered through a timeout, resulting in rbuf blocking for the period of the timeout, thus the chances of rbuf blocking are higher during periods of missing packet's recovery through retransmissions. Since each timeout causes congestion window reduction at a sender, and entails idle time (i.e., sender not transmitting data), that ultimately causing throughput reduction.

In this research work, our purpose is to study the impact of network congestion-based losses on rbuf blocking during the concurrent multiple path transfer of data. Although, several retransmissions policies (Iyengar et al., 2004; Iyengar et al., 2005) are suggested to reduce the rbuf-blocking problem in SCTP-CMT at transport layer, but rbuf blocking problem cannot be eliminated. We also demonstrate this problem in the next Section 3.2.1, through network simulations and study the performance of Concurrent Multipath Transfer using SCTP-CMT in the presence of a bounded receive buffer (rbuf). We then identify that reducing (or eliminating) the number of packet losses will reduce the rbuf blocking problem in SCTP-CMT, *but in the real Internet it is not possible for the SCTP-CMT to avoid from these losses (mostly due to congestion) due to its loss-based congestion detection mechanisms.* We then will identify and come to the conclusion that rbuf blocking problem in SCTP-CMT multihoming is mostly due to its loss-based nature for detecting congestion.

As the packet loss can be caused by a number of factors, including signal degradation over the network medium, oversaturated network links, corrupted packets rejected in-transit, faulty networking hardware (etc). But the most common reason for packet loss is the network congestion and this congestion is sensed by the loss-based protocols through the packet loss indication. It means the network congestion is only sensed when packet is actually lost by the loss-based protocols (like SCTP), as we have shown in the single-path scenario (Figure 2 – Figure 5) and we also observed that such kinds of problems are far away from FAST TCP, because it uses queuing-delay instead of loss probability as a congestion signal.

3.2.1 Experimental Setup

For our simulations, we use the University of Delaware's SCTP module (Caro et al.), which is now part of the latest ns simulator distribution (VINT Project, Network Simulator). Figure 6 demonstrates the simple network topology simulated: a dual-dumbbell topology whose core links have a bandwidth of 10Mbps and a one-way propagation delay of 50ms. The router pairs (R_1, R_2) and (R_3, R_4) are attached to three and five edge nodes, respectively. One of these edge nodes is a dual-homed node for an SCTP endpoint, while the other two/four nodes are single-homed and introduce background-traffic over the forward paths that creates loss for the SCTP traffic.

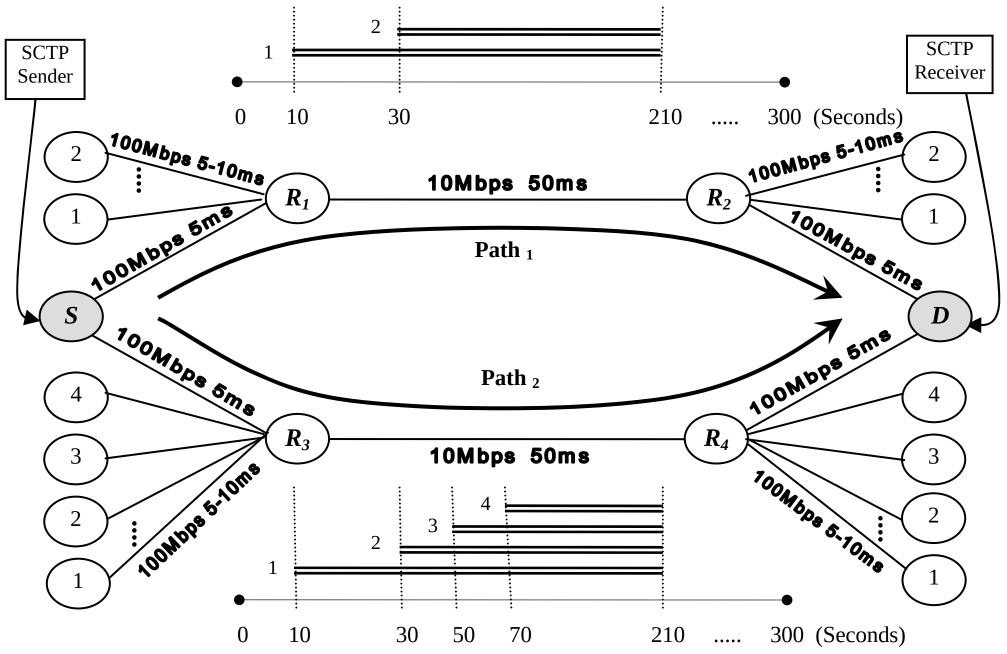


Figure 6: Simulation network topology with background-traffic, active periods of the flows, and congestion-based losses

The links to the dual-homed nodes have a bandwidth of 100Mbps and a one-way propagation delay of 5ms. The single-homed nodes also have 100Mbps links, but their propagation delays are randomly chosen from a uniform distribution between 5-10ms to simulate end-to-end one-way propagation delays between 60ms and
70ms. The end-to-end one-way propagation delay between the two dual-homed nodes is 60ms, and in this experiment loss is introduced by making the buffer size small for each link (both edge and core) to support all of the active flows.

This configuration has two SCTP endpoints with CMT (sender *S*, receiver *D*) on either side of the network, which are attached to the dual-homed edge nodes. *S* has two paths, labeled $Path_1$ and $Path_2$, to *D*. And each single-homed SCTP edge node has a single traffic generator, which only used for background-traffic over the forward paths with different active periods (flows) as shown in Figure 6. The SCTP-CMT dual-homed sender (*S*) starts at time zero and we run this simulation for the period of 300 seconds. We set every packet size to 1500 bytes and the rbuf is sized at 64KB for the dual-homed receiver (*D*) with RTX-SAME retransmission policy.

3.2.2 Results and Discussions

In this section, we analysis the impact of rbuf blocking on CMT due to network congestion based losses, for this we present an extract from a simulation of a CMT association using the topology shown in Figure 6. Figure 7(a) and Figure 7(b) show transmission sequence number (TSN) progression over $Path_1$ and $Path_2$, respectively, and Figure 7(c) shows rwnd evolution at the sender (endpoint *S*) during the time interval from 20 to 40 seconds. Figure 7(d) shows the SCTP-CMT sender's (*S*) observed cwnds evolution for the whole association ($Path_{1,}$ $Path_2$) during the period of 300 seconds simulation run, which shows a number of cwnd reductions for both the paths, for example, the cwnd for $Path_1$ is reduced to half at times 16.52, 26.65 and 28.51 etc. Since cwnd reductions are noticed when a sender detects loss, but for $Path_1$ (from Figure 7(a)) no packet loss is observed (even not a single packet loss is observed throughout 300 seconds simulation run), and Figure 7(a) also shows that data transmission over the $Path_1$ (i.e., less congested path) stops abruptly around 27.61 seconds and resumes around 34.98 seconds. This 7.37 seconds pause can be explained with the help of Figure 7(c), which shows that at time 27.61 seconds, the peer-rwnd at the sender *S* abruptly reduces to 944 bytes, thus constraining the sender (an SCTP-CMT sender shares a single finite rbuf across all paths) from transmitting any new data through any path.

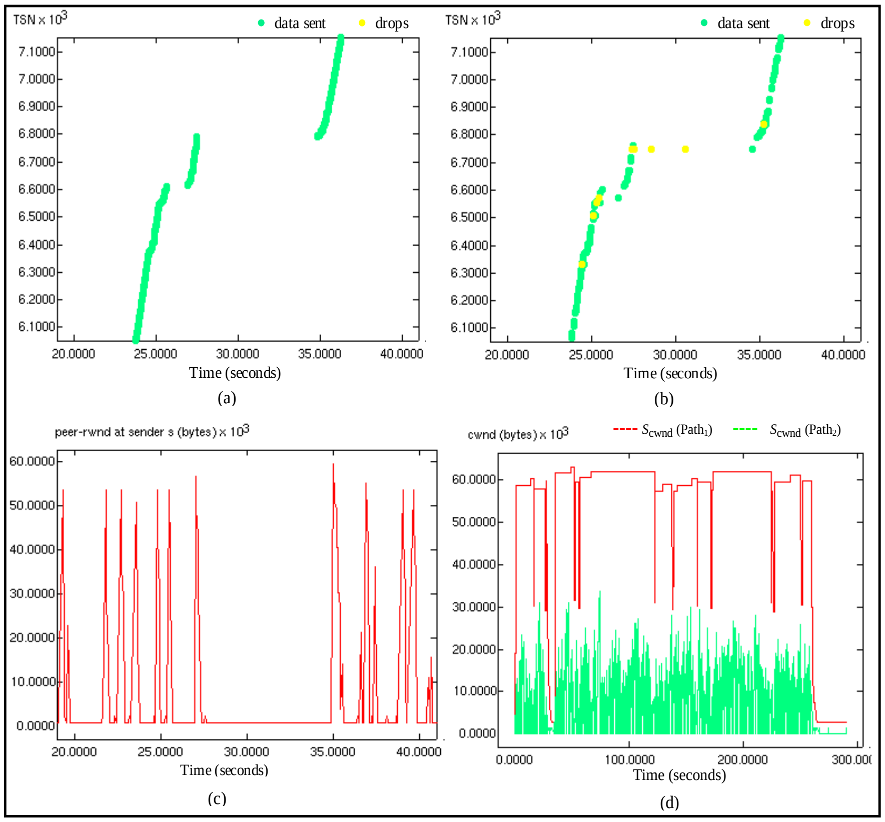


Figure 7: Instantiation of rbuf blocking: (a) Progression of data sent to destination *D* through $Path_1$ over a select interval; (b) Progression of data sent to destination *D* through $Path_2$ over same interval; (c) peer-rwnd value maintained at sender *S* over same interval; (d) CMT cwnds evaluation for both the paths over the 300 seconds simulation run

And the reason for this unexpected rbuf reduction is that the $Path_2$ (i.e., highly congested path) experiences a sever congestion during the same time interval from 27.61 to 34.98 seconds, as a result, produces a consecutive packets losses as shown in Figure 7(b). Since these losses in our simulation only occur due to congestion, we do not set the loss rate. Instead, we set a simple level of forward-path traffic to analyze the impact of rbuf blocking on CMT due to congestive losses. Figure 7(b) also illustrates that $Path_2$ recovers from these losses through repeated retransmission timeouts– the longest recovery time being 7 seconds for TSN 6744. During this entire period of 7.37 seconds while 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 that further forces the CMT's cwnds to stop transferring data on either path (as shown in Figure 7(d)), and hence reducing the overall throughput for SCTP-CMT association.

Furthermore, it is observed that at time around 260.2 seconds the peer-rwnd at the Sender *S* reduces to zero bytes and never recovers from these losses, not even through repeated retransmission timeouts, this is due to back-to-back timeouts with exponential back-off results in permanently blocking the rbuf.

3.2.3 Suggested Solution

As we just examined (in Subsection 3.2.2), the $Path_2$ (highly congested path) is a main source of this rbuf blocking in concurrent multipath transfer of data (CMT), because there is a very little correlation between the SCTP-CMT window size and the level of background traffic over the $Path_2$, for example, as the background traffic increases at time 10 and 30 seconds (when flow 1 and flow 2 join the $Path_2$), the SCTP-CMT sender *S* keeps increasing its window size too until there is a congestion over the $Path_2$. This results in losses, both to itself and to flows which are part of the background traffic over the $Path_2$, and hence an increase in the number of timeouts, also increases loss recovery time on $Path_2$ (especially for fast retransmit based recovery), results in blocking the CMT rbuf.

On the other hand, Figure 2 – Figure 5 clearly show FAST TCP's congestion avoidance mechanisms at work and how its throughput adapts to the changing conditions on the network. FAST TCP strategy is to adjust the source's sending rate in an attempt to keep a small number of packets buffered in the routers along the path such that it never exceeds the delay-bandwidth product of the connection plus the number of buffers at the bottleneck. This technique gives FAST TCP the ability to anticipate congestion, and adjust its transmission rate accordingly in such a way that there are *little or no losses*. We believe that *if packets are sent through multiple paths (having different traffic-load distribution) simultaneously to destination using end-to-end multihoming, the packets are highly likely to arrive in the order they were initially sent (preventing rbuf from blocking)*, and this belief is based on the experiments reported in this paper for comparing the protocols.

Therefore, we argue that– under CMT (which uses two congested paths) FAST TCP will perform much better than SCTP in high-speed multihomed networks under the same finite receive-buffer size due to its delay-based congestion control mechanisms.

To end with, we motivate delay-based approach (i.e., FAST TCP) as a congestion control mechanism used for implementing the end-to-end transport layer multihoming for parallel data transfer (in high-speed long-distance networks) rather than other loss-based congestion control protocols.

4. Conclusions and Future Work

In this research paper, we have studied the congestion control mechanisms of FAST TCP and SCTP protocols. We have conducted simple simulations to evaluate the performance of delay-based (FAST TCP) versus loss-based (SCTP) on high-speed networks (ns-2), and through these simulations we have shown that FAST TCP often has very good performance under a similar network conditions.

In this study, we demonstrated the weakness of SCTP-CMT rbuf constraints and, we then exposed that rbuf blocking problem in SCTP-CMT multihoming was mostly due to its loss-based nature for detecting network congestion.

Space restrictions naturally limit the number of results that we can show; however, the experimental results and survey presented in this research provide insight on design decisions for the future high-speed multihomed transport protocols. We conclude hat FAST TCP is better suited as a transport layer protocol for parallel data transfer through multiple paths using end-to-end multihoming because of its several distinct features not present in current TCP and SCTP. In this way, after deploying these options in FAST TCP, it will be able to meet the increasing demand of the future high-speed network infrastructures such as in Grid computing.

In our forthcoming article, a number of issues will be discussed in attempting to develop such a transport layer protocol based on FAST TCP, which can transfer data parallel through multiple paths using end-to-end multihoming. The problem areas in this design and a brief introduction to each of the problems that are discussed in this research along with different alternatives will also be addressed. Moreover, the complex network and more experiments will be simulated to prove the practicability of this policy.

Corresponding Author:
M. Junaid Arshad
Department of Computer Science
University of Engineering and Technology
Lahore, Pakistan
Email: junaidarshad@uet.edu.pk
junaid_qu@yahoo.com
Tel: 92-42-9029260

Received: 09/17/2008

References

1. Sally Floyd, “HighSpeed TCP for large congestion windows”. Internet draft draft-floyd-tcp-highspeed-02.txt, work in progress, http://www.icir.org/floyd/hstcp.html, February 2003.
2. Tom Kelly, “Scalable TCP: Improving performance in highspeed wide area networks,” Submitted for publication, http://www-lce.eng. cam.ac.uk/~ctk21/scalable/, Dec 2002.
3. IBM, “Ibm grid computing home page,” URL: http://www.ibm.com/grid.
4. M. Ohta, “The Architecture of End to End Multihoming,” Internet-draft, IETF (Nov 2002), draft-ohta-e2e-multihoming-03.txt.
5. V. Jacobson. Congestion avoidance and control. ACM Computer Communication Review, 18:314–329, August 1988.
6. C. Jin, D. Wei, and S. H. Low, FAST TCP: motivation, architecture, algorithms, performance, Tech. Rep. CaltechCSTR: 2003.010, Caltech, Pasadena CA, 2003, http://netlab.caltech.edu/FAST.
7. H. Bullot and L. Cottrell, “Tcp stacks testbed,” www-iepm.slac.stanford.edu/bw/tcp-eval/.
8. J. R. Iyengar, K. C. Shah, P. D. Amer, and R. Stewart, “Concurrent multipath transfer using SCTP multihoming,” in Proc. SPECTS, San Jose, CA, July 2004.
9. M. Allman, V. Paxson, and W.Stevens,”TCP Congestion Control,” RFC2581, IETF, April 1999, http://www.ietf.org/rfc/rfc2581.txt.
10. R. Stewart et al. Stream control transmission protocol. IETF RFC 2960, Oct. 2000.
11. VINT Project, Network Simulator ns-2, http://www.isi.edu/nsnam/ns/.
12. A. Caro and J. Iyengar, “ns-2 SCTP module,” http://pel.cis.udel.edu.
13. T. Cui and L. Andrew, "FAST TCP simulator module for ns-2, version 1.1", http://www.cubinlab.ee.mu.oz.au/ns2fasttcp.
14. J. Iyengar, P. Amer, and R. Stewart. Receive Buffer Blocking In Concurrent Multipath Transport. In IEEE GLOBECOM, St. Louis, Missouri, November 2005.
15. J. Iyengar, P. Amer, and R. Stewart. Retransmission Policies For Concurrent Multipath Transfer Using SCTP Multihoming. In ICON 2004, Singapore, November 2004.