A Simulation-Based Study of FAST TCP Compared to SCTP: Towards Multihoming Implementation Using FAST TCP

Mohammad Junaid Arshad and Mohammad Saleem

Abstract: **The current multihome-aware protocols (like stream control transmission protocol (SCTP) or parallel TCP for concurrent multipath data transfer (CMT) are not designed for high-capacity and large-latency networks; they often have performance problems transferring large data files over shared long-distance wide area networks. It has been shown that SCTP-CMT is more sensitive to receive buffer (rbuf) constraints, and this rbuf-blocking problem causes considerable throughput loss when multiple paths are used simultaneously. 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 for solving a number of throughput issues. This work proposes an end-to-end transport layer protocol (i.e., FAST TCP multihoming as a reliable, delay-based, multihome-aware, and selective ACK-based transport protocol), which can transfer data between a multihomed source and destination hosts through multiple paths simultaneously. Through extensive ns-2 simulations, we show that FAST TCP multihoming achieves the desired goals under a variety of network conditions. The experimental results and survey presented in this research also provide an insight on design decisions for the future high-speed multihomed transport layer protocols.**

Index Terms: **Concurrent multipath transfer (CMT), FAST TCP, multihoming, receiver buffer (rbuf), stream control transmission protocol (SCTP).**

I. INTRODUCTION

The Internet is a large group of millions of computers around the world that are all connected to one another and transmit data by packet switching based on the TCP/IP protocol suite. Although the current stability of the Internet largely relies on TCP's end-to-end congestion control mechanism, this classical protocol (which still constitutes the basis of today's Internet) has reached its limits of scalability and functionality. The Internet, due to its huge growth, complexity, and popularity, now continuously experiences modifications and changes in almost all aspects. There has been very fast progress in the area of grid [1], peer-to-peer, and distributed applications, although not much attention has not been focused to guarantee that the huge amount of data produced by such applications can be transferred and shared effectively over the wide area networks. In particular, a significant issue in the approach of high-speed grid applications [2] is network transport. It means an efficient and proper transport protocol is required to facilitate the capacity offered by such high-bandwidth long-distance network infrastructures.

In this context, it is important to address the performance problems of the current loss-based congestion control protocols (like TCP [3] and stream control transmission protocol (SCTP) [4], [5]) over large bandwidth-delay product paths when doing huge data transfers. The researchers realize that these protocols's somewhat naive congestion control approaches will quickly happen to be incompetent of successfully utilizing the network resources under extremely high-capacity and congested network environments, which motivates the design of new distributed algorithms for large delay-bandwidth product networks (i.e., FAST TCP [6]). The additive increase/multiplicative decrease (AIMD) [7] principal of TCP and SCTP protocols leads to potentially violent oscillations in window sizes, round-trip delay, and queue length of the bottleneck node, which could clearly impact the overall efficiency of the network connection.

While FAST TCP's delay-based approach [6] avoids such limitations; it aims to provide an overhaul to the loss-based AIMD algorithms by using not only queuing-delay but also packet loss as a metric to sense network congestion, because queuing-delay gives a better estimation of congestion and balances more efficiently with the available link-capacity of the network than packet-loss probability does [6] (which results in efficient link utilization).

Concurrent multipath transfer (CMT) is another promising approach for improving the consistency of the Internet connections, building healthier use of end-to-end multihoming, and utilizing the additional network capacity for improved application throughput. The SCTP supports CMT [7] between end-to-end multihomed hosts, while the standard TCP [3] and its variant (i.e., FAST TCP) do not support multihoming. In the early days of the Internet, extensive use of multihoming was not practicable for the reason of cost constraints, but nowadays, network interfaces have become ordinary and affordable items. We deem that in the near future, most Internet hosts will be multihomed, and so multihoming should be supported all across the Internet.

Thus, we believe that a transport protocol right for bulk data transfers/shares efficiently over the future high-speed optical networks (such as in highly dynamic environments like peer-to-peer and wide-area grid computing [2]) should take full advantage of the above approaches (i.e., end-to-end multihoming and high-speed algorithm like FAST TCP). It means to improve the

Manuscript received July 28, 2008; approved for publication by Saewoong Bahk, Division III Editor, September 18, 2009.

This work was supported by the Directorate of Research Extension and Advisory Services, UET, Lahore.

The authors are with the Department of Computer Science and Engineering, University of Engineering and Technology, Lahore-Pakistan, email: {junaidarshad, drmsaleem}@uet.edu.pk.

performance and reliability of increasingly bandwidth-greedy applications, and new design schemes for the protocols for the identified future networks problems will be proposed (or one way of achieving this goal is to embed multihoming capability within an existing transport protocol such as FAST TCP).

The remainder of this paper is organized as follows. Section II motivates FAST TCP used for multihoming by demonstrating the advantages of replacing loss-based approach with delay-based approach as the end-to-end transport-layer congestion control mechanism. In this section, we also present our simulation setup with network topologies simulated using ns-2 simulator [8], analyze the behavior of FAST TCP compared to SCTP, delineate the receive buffer (rbuf) blocking problems in SCTP-CMT [9], and identify the dilemma degrading its performance in the presence of a bounded rbuf. In Section III, we first introduce the key insight into the design and layout architecture, including an outline of our proposed algorithms for multihoming implementation using FAST TCP. Next, we present a performance analysis of FAST TCP multihoming by using a particular network scenario in ns-2 to prove its efficiency in producing high end-to-end throughput in multiple path environments. Finally, in Section IV, we present the conclusions.

II. MOTIVATION FOR MULTIHOMING IMPLEMENTATION USING FAST TCP

A. SCTP versus FAST TCP in High-Speed Networks (Loss-Based versus Delay-Based Approach)

In this section, we present a performance comparison of SCTP and FAST TCP protocols through simulation results and discuss the protocols behavior. However, there have been many works on comparisons between standard TCP and FAST TCP, but there seems no work that directly compares FAST TCP and SCTP (since congestion control algorithm of SCTP is almost the same as that of the standard TCP). Since protocols performance evaluations are addressed in several papers (e.g., [6], [10]–[14]) proposing revisions to the standard TCP's AIMD algorithm but with the vast deployment of high-performance grid and optical networks, there are still continuing examinations to assess the existing set of high-speed transport protocols to find out the best overall protocol.

FAST TCP's delay-based approach is essentially different from AIMD; its aim is to utilize not just queuing delay but also packet loss as an indication to sense congestion, and its benefit over loss-based congestion control algorithm is small at low-speed, but dominant at high-speed. The congestion window (cwnd) update algorithm of FAST TCP calculates the precise window size on the basis of the current measurement of queuing delay whenever reliable round-trip time (RTT) estimations are available, i.e., qdelay = avgRTT − baseRTT.

The jth RTT sample $T(j)$ recalculates the average RTT $T^*(j)$ according to the following equation:

$$T^*(j+1) = (1-\eta)T^*(j) + \eta T(j). \qquad (1)$$

FAST TCP periodically recalculates the cwnd on the basis of the average queuing-delay and average RTT, according to (2), as described in [6].

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

FAST TCP includes four separate components, as depicted in Fig. 1.

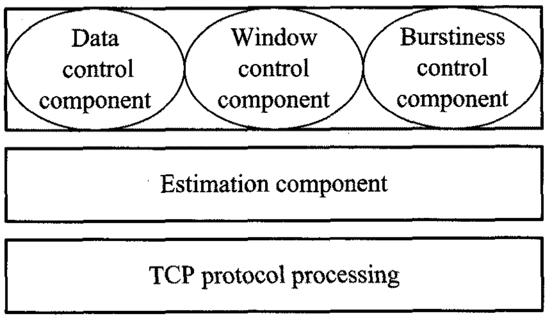


Fig. 1. FAST TCP architecture.

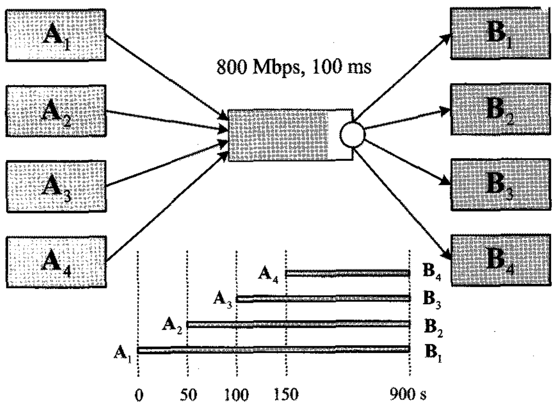


Fig. 2. Simulation network topology with a single bottleneck shared by four flows, with bottleneck link capacity of 800 Mbps, a common propagation delay of 100 ms, drop-tail queuing and the buffer size of 3000 packets with a fixed packet length of 1500 bytes ($\alpha = 200$ packets for each FAST TCP flow).

A.1 Simulation Results and Discussions

We used FAST TCP simulator module for ns-2 [15], ver-1.1 (selective ACK (SACK) introduced) and for SCTP, we used University of Delaware's module [16]. Fig. 2 shows the network topology simulated to compare the protocols performance in terms of queue occupancy, aggregate throughput, congestion window, and total number of packets lost at the bottleneck link.

SCTP's curves in Fig. 3 show that during the slow-start phase, there is no in-advance information of the available bandwidth that can be exercised to stop the exponential increase of the windows. Therefore, we notice that the sources increase their cwnds until the available bandwidth is exceeded, and they use progressively more buffers in the router until they lose packets by overflowing the bottleneck queue. We also see that as more number of SCTP competing sources join the network, stability becomes worse for this loss-based protocol, produces more oscillations in its cwnds and queue size, and increases packet losses (Fig. 3) in the network; since all these losses are only caused by congestion

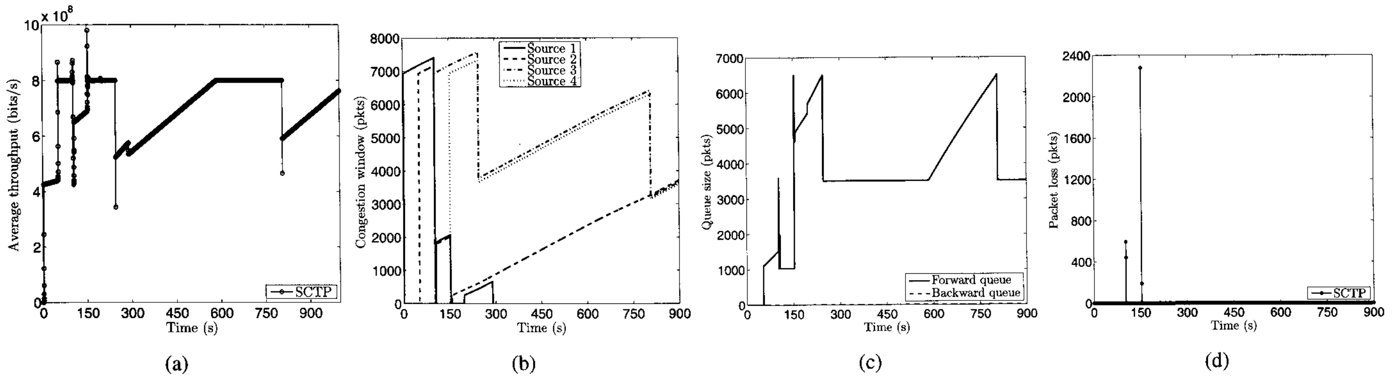


Fig. 3. SCTP's curves: (a) Aggregate throughput, (b) congestion window, (c) queue size, and (d) packet loss.

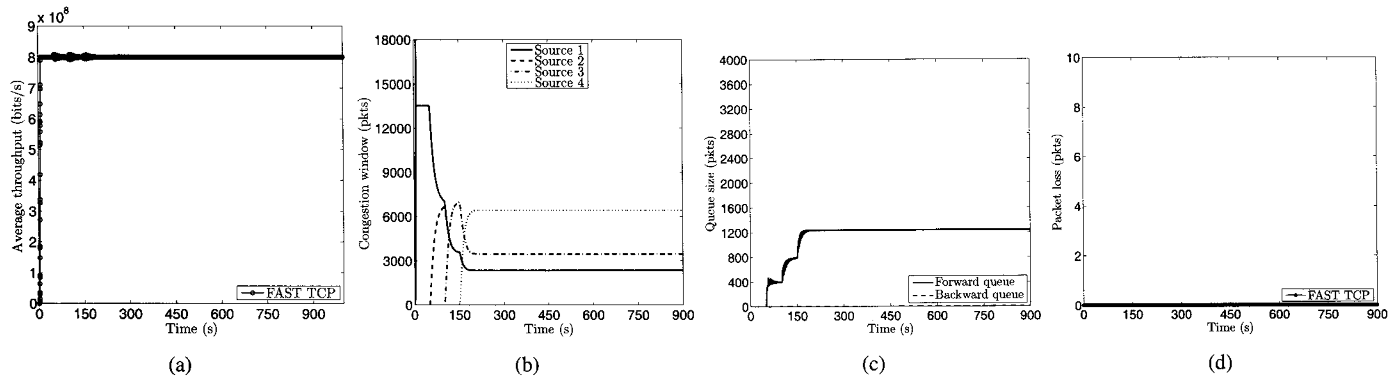


Fig. 4. FAST TCP's curves: (a) Aggregate throughput, (b) congestion window, (c) queue size, and (d) packet loss.

at the routers, we do not set the loss rate in our simulations.

While under same conditions, FAST TCP consistently does better than SCTP in terms of throughput and stability with zero packet loss at the bottleneck because each source attempts to keep the same number of packets in the queue in equilibrium so that each competing source equally shares the bottleneck link bandwidth, as shown in Fig. 4, which clearly shows that FAST achieves a better aggregate throughput (seeing as they can maintain the network link around full utilization).

B. Rbuf Blocking in CMT Using SCTP Multihoming

B.1 Problem Description

SCTP is a relatively new transport layer protocol that natively supports multihoming. It is an Internet engineering task force (IETF) standards-track protocol, which has not yet been largely deployed in the Internet regardless of its several advantages over standard Internet Protocols (UDP and TCP); however, research on extending SCTP to CMT using multihoming is currently in progress [5].

In SCTP-CMT, the multihomed receiver keeps a single rbuf, which is shared across all the paths (subflows), and it consumes data purely in order, regardless of the destination addresses they are directed to. The transmission rate of an SCTP sender is bounded by the peer-receiver window together with the relevant destination's congestion window. It has been shown in [9] that SCTP-CMT is more sensitive to rbuf constraints, which causes considerable throughput deficiency if data is transferred through multiple paths simultaneously.

B.2 Rbuf Blocking in Concurrent Multipath Transport due to Network Congestion-Based Losses

In this section, we study the impact of network congestion-based losses on rbuf blocking in CMT. During the concurrent multipath transfer of data when a path undergoes failure (due to congestive losses or non-congestive losses), its outstanding data has to be recovered by means of a retransmission timeout (RTO), which in turn causes rbuf blocking for the period of the timeout; thus, the possibilities of rbuf blocking are greater during periods of missing packet's recovery through retransmissions. Since each timeout results in the reduction of congestion window at the sender and causes idle time (that is, sender not sending data) that ultimately results in throughput degradation.

Although, several retransmissions policies [9], [17] are suggested to reduce the rbuf blocking problem in SCTP-CMT at the transport layer, rbuf blocking problem cannot be eliminated. We also demonstrate this problem in this section through simulations and analysis of the performance of SCTP-CMT during the occurrence of a bounded 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 these losses (mostly, due to congestion) due to its loss-based congestion detection mechanisms.* We then will identify and come to the conclusion that the rbuf blocking problem in SCTP-CMT multihoming is mostly due to its loss-based nature of detecting congestion.

B.2.a Simulation Setup. We use Univ. of Delaware's SCTP module [16] for our simulations; this module is currently a part

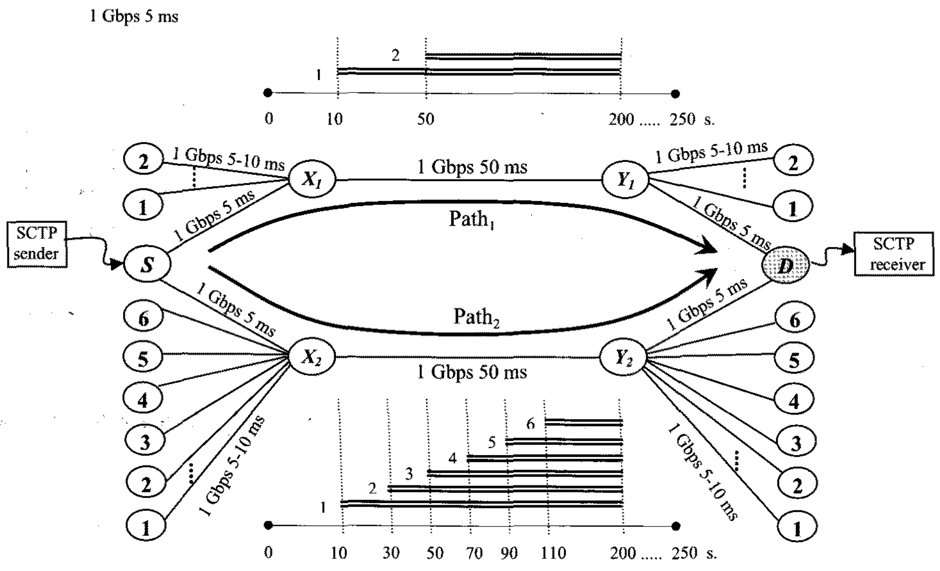


Fig. 5. Dual-homed simulation network topology with background-traffic, active periods of the flows, and congestion-based losses over 1 Gbps/60 ms delay link.

of the latest network simulator (*ns*) distribution [8]. We simulated a dual-dumbbell network topology with its core links having a capacity of 1 Gbps along with a one-way propagation delay of 50 ms, as illustrated in Fig. 5. The router pairs $(X1, Y1)$ and $(X2, Y2)$ are attached to three and seven edge nodes, respectively. For an SCTP endpoint, one of these edge nodes is a dual-homed node, whereas the remaining two/six nodes are single-homed and used for introducing background-traffic over the forward paths so as to create packet losses for the SCTP traffic.

The SCTP dual-homed nodes have a propagation delay of 5 ms, while propagation delays of the single-homed nodes are picked at random way from the same distribution between 5–10 ms to simulate end-to-end propagation delays in one direction ranging between 60 ms and 70 ms. The end-to-end one-way propagation delay between the two dual-homed nodes is 60 ms, and in this experiment, the router's buffer size is set to 400 Mb for each link (both edge and core) to support all of the active flows because in reality, the standard router's buffer size depends upon the number of flows and is typically in the range of couple of 100 Mb to Gb (this is quite common now).

This simulation setup includes two SCTP-CMT endpoints (a sender *S* and a receiver *D*); *S* has two paths (i.e., path_1 and path_2) to *D*, and each SCTP single-homed edging node has a single traffic generator, which is only used for background-traffic over the forward paths having different active flows, as presented in Fig. 5. The SCTP-CMT dual-homed sender (*S*) starts at time zero, and the simulation duration is 250 sec. We set every packet size to 1500 bytes, and the rbuf is sized at 1.2 Mb for the dual-homed receiver (*D*) with RTX-SAME retransmission policy.

B.2.b Simulation Results and Discussions. To demonstrate the rbuf blocking problem, an extract is presented from the SCTP-CMT association's simulation using the topology depicted in Fig. 5. Figs. 6(a) and 6(b) present the results for transmission sequence number progression over path_1 and path_2, respectively. Fig. 6(c) presents the peer-receiver window evolution during the time interval from 135 to 165 s at the sender (*S*) endpoint. Fig. 7 shows the SCTP-CMT sender's (*S*) observed cwnds evolution for the whole association (path_1 and path_2) during the period of the 250 s simulation run, which shows a number of cwnds reductions for both the paths, e.g., cwnd for path_1 is halved at times 21.66, 24.96, 30.38, etc. Congestion window reductions are noticed as a sender finds loss; however, for path_1 (from Fig. 6(a)), no packet loss is observed (not even a single packet loss is noticed throughout the 250 s simulation run), and it is also observed that data transmission over the path_1 (less congested path) discontinues at different times during the 250 s simulation run; for example, it stops unexpectedly around 145.75 s and starts again around 148.95 s. Fig. 6(c) presents the results to explain this 3.2 s pause, which reveals that at time 145.75 s, the peer-receiver window unexpectedly decreases to 644 bytes at the sender *S* (as an SCTP-CMT sender shares a single finite rbuf across all paths), thus constraining it from sending any new data through any path.

The reason for this surprising rbuf reduction is that path_2 (i.e., the highly congested path) experiences a severe congestion during the same time interval from 145.75 to 148.95 s, producing, as a result, consecutive packets losses (that only occur because of congestion under a simple level of forward-path traffic), as shown in Fig. 6(b). The sender starts to recover path_2 from these losses via repeated RTOs; meanwhile, the receiver awaits retransmissions to arrive and holds some of the subsequent transmission sequence numbers in the rbuf, which were transmitted over the path_1 and are not delivered to the application until the retransmissions are obtained, resulting in the blocking of the rbuf as well as the peer-receiver window.

path_2 (the highly congested path) is the main source of rbuf blocking in SCTP-CMT, and this is due to the lack of correlation between the level of background traffic and the SCTP-CMT window size over the path_2. Moreover, it is observed that at time around 202.8 s, the peer-receiver window reduces to zero bytes at the sender *S* and never recovers from losses, not even

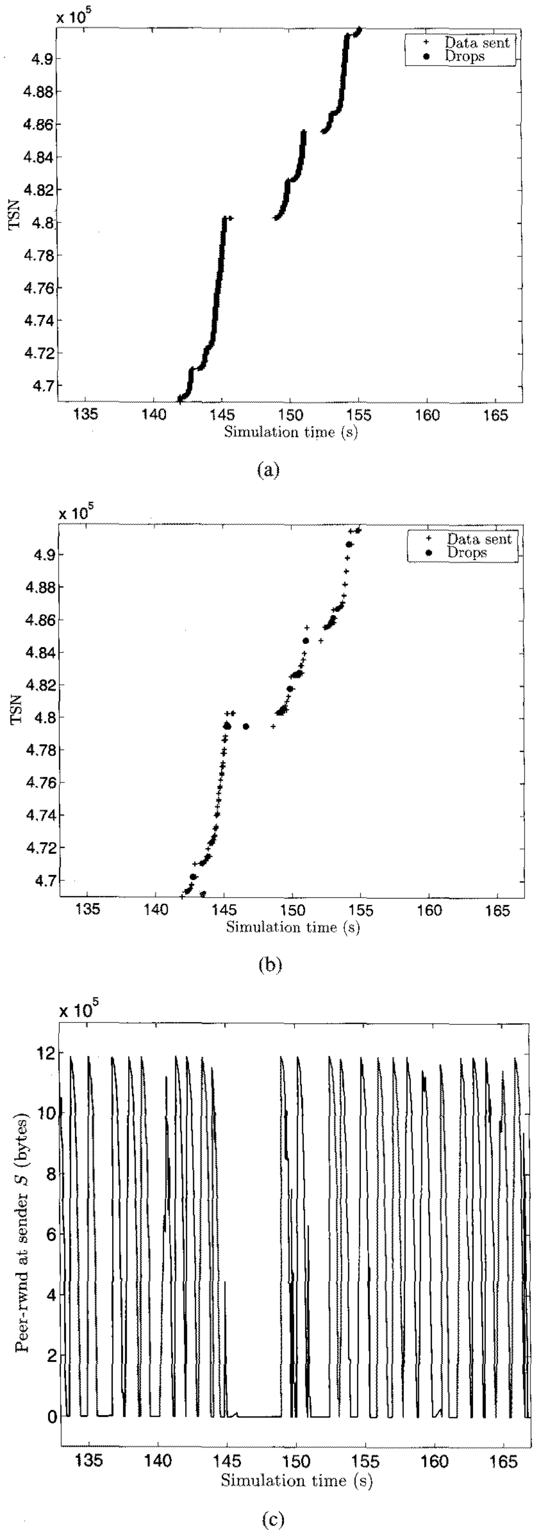


Fig. 6. SCTP-CMT's receive-buffer blocking over a select interval: (a) Transmission sequence number progression over path_1, (b) transmission sequence number progression over path_2, and (c) peer-receiver window value structured at sender S.

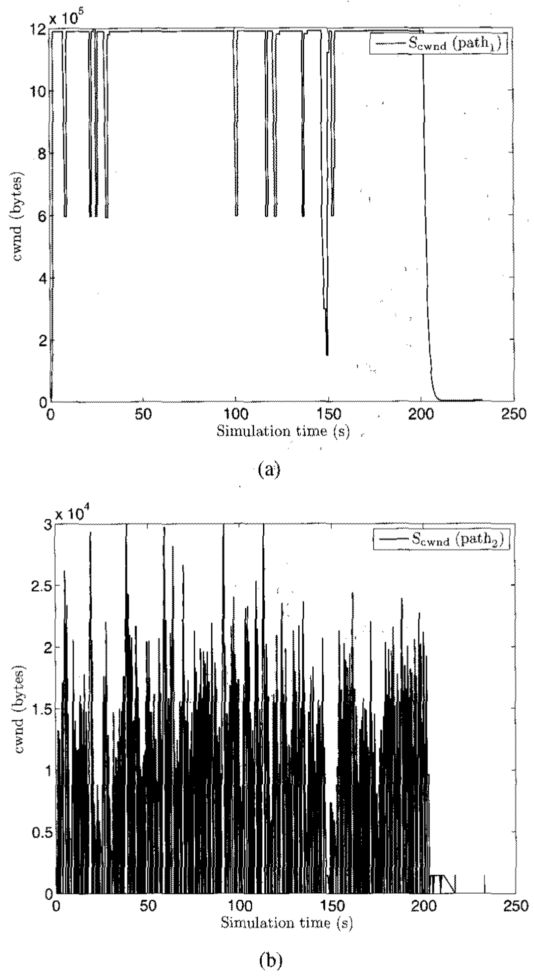


Fig. 7. SCTP-CMT's congestion windows evaluation over the 250 s simulation run: (a) Congestion window evolution over path_1 and (b) congestion window evolution over path_2.

through repeated RTOs; this is due to back-to-back timeouts with an exponential back off resulting in permanent blocking of the rbuf.

In contrast, Figs. 3 and 4 (in Section II-A) clearly show that FAST TCP's congestion avoidance mechanisms at work has the ability to anticipate congestion and adjusts its transmission rate accordingly in such a way that there are *little or no losses*. 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 rbuf size due to its delay-based congestion control mechanisms. To end with, we motivate the delay-based approach (i.e., FAST TCP) as a congestion control mechanism used for implementing end-to-end transport-layer multihoming for parallel data transfer (in high-speed long-distance networks) rather than other loss-based congestion control protocols.

III. FAST TCP MULTIHOMING PROPOSED DESIGN AND ALGORITHMS

In this section, we first go over the main points of research on existing techniques of transport-layer multihoming, which have been discussed in [5], [18]–[20], etc.

Table 1, thus, summarizes some existing mechanisms of the SCTP-CMT multihoming approach along with the proposed key mechanisms for FAST TCP multihoming.

Now, it is important to propose the key design elements of FAST TCP multihoming in such a way that it will provide the

Table 1. Overview of SCTP-CMT/FAST TCP multihoming approach.

Mechanism	Issue	Review of SCTP-CMT multihoming approach	Proposed FAST TCP multihoming approach
Congestion control for multiple paths	Separate or shared?	Each path has its own loss-based congestion control mechanism	Each path has its own delay-based congestion control mechanism
Receiver's advertised window (rwnd) at the sender for multiple paths	Shared buffer or individual buffer for each path?	Receiver maintains a single rbuf, which is shared across all subflows (paths) in an association	Same as SCTP-CMT
Sequence space among flows on different paths that occur within an association	Separate or shared?	A single sequence space is used across an association's multiple paths	Same as SCTP-CMT
Loss detection and recovery	Separate or shared fast retransmit algorithm?	Fast retransmit algorithm (on a per destination basis)	Split estimation component (SEC) algorithms for multiple paths
Reverse path for acknowledgements	Where to send a SACK?	Normally SACKs return over the same path to which they were originally sent	SACKs return over the same path to which they were originally sent
Retransmission path	Where to send a retransmission?	Using CMT retransmissions policies	Over the same path to which it was originally sent

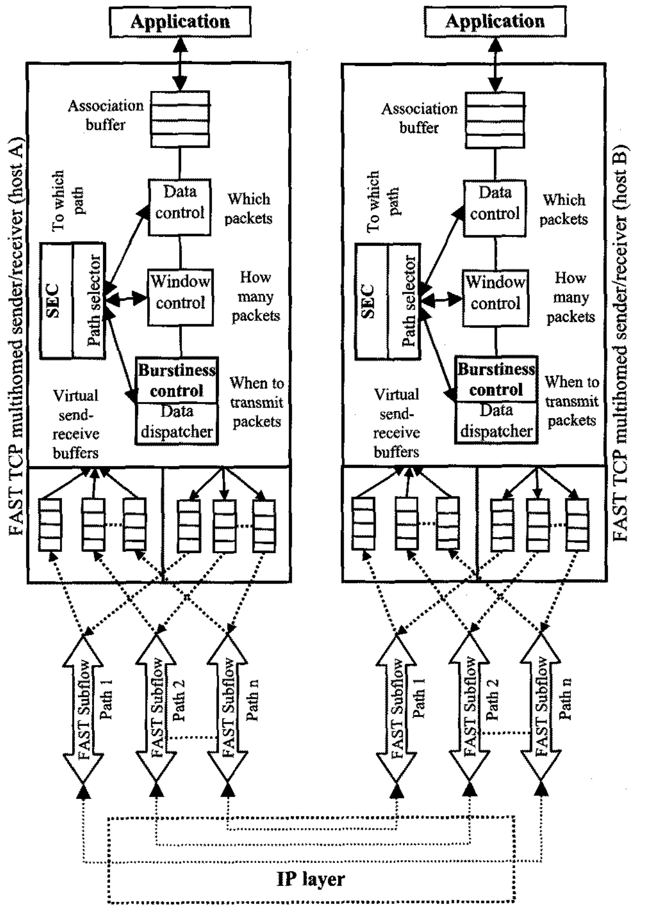


Fig. 8. Architectural view of FAST TCP multihoming.

same semantics to applications as simple FAST TCP—it will preserve the properties such as stability, fairness, reliability, and responsiveness. Thus, some key design elements are as follows:

- Since FAST TCP multihomed sender host sends data through multiple paths concurrently and different paths have different delay and/or bandwidth characteristics, each path needs to have its own window control mechanism, like a simple FAST TCP [6]. This means each path needs its own estimation component module (like FAST TCP) to estimate RTT, remember which packets it has sent, and match each received ACK with one of these packets.
- Each path maintains a cwnd as in simple FAST TCP. The cwnd changes independently as the path adapts to the network state (i.e., FAST TCP periodically recalculates the cwnd on the basis of the average queuing-delay and avgRTT provided by the estimation component, according to (3)).
- In our current design, we use a single sequence space across the multiple paths of an association that is used for congestion control as well as loss detection and recovery. The multihomed sender maintains a set of per path virtual queues and spreads the packets across all available paths; immediately, the congestion window allows it. Retransmissions are prompted only when a number of SACKs (generally three duplicate acknowledgements) report the missing data packets from the same virtual queue.
- In our current prototype, the FAST TCP multihomed receiver maintains and shares a single rbuf across all the paths within an association. In this way, the FAST TCP multihomed sender divides the global advertised window among all its subflows (paths) proportional to their congestion window sizes.
- As on the reception of an acknowledgement, FAST TCP updates its cwnd on the basis of the queuing delay, i.e., qdelay = avgRTT − baseRTT according to (2); hence, we have to keep a steady stream of SACKs from new transmissions with the aim of obtaining unambiguous-RTT measurements. This is generally possible only when all ACKs return over the same path to which they were originally sent. Thus, in this proposed design, although the data is transferred through multiple paths concurrently, all the SACKs return over the same path to which they were originally sent.

In the next section, we describe the overall architecture of FAST TCP multihoming with its different components and algorithms that will make simultaneous data transfer through multiple paths possible for achieving the maximum end-to-end throughput.

A. Component Design and Algorithms

We reviewed and discussed some issues in our effort to propose such a transport-layer multihome-aware protocol that is based on FAST; hence, first, the simple FAST's estimation, data control, burstiness control, and window control mechanisms need to be modified to completely take advantage of the benefits that FAST TCP multihoming has to offer.

```
On receiving a positive acknowledgement (a SACK carries a new ACK) [Processing at sender side]:
    1) Let $p_i$ to be the complete set of paths from source to destination.
           $\forall$ paths $p_i$, initialize $p_i$.SACK_received_path = FALSE;
    2) For each data packet $s_a$ that is acknowledged in SACK thus far do
           let $p_a$ be the path on which SACK is received;  /* i.e., a path on which $s_a$ was originally sent. */
           set $p_a$.SACK_received_path = TRUE;
 [Parameter estimation for the path on which SACK is received]:
    3) For each path $p_i$
        if ($p_i$.SACK_received_path == TRUE) then
           (i)   get average cwnd size (one average RTT ago) which will be used as a history record;
           (ii)  calculate the new average RTT as per (2) which will be used to estimate queueing delay for the path $p_a$;
           (iii) estimate the average queueing delay as per (1) for the path $p_a$ which will be used
                 to calculate the new cwnd size as per (3) in every update;
           (iv)  provide this information to the other three decision-making components of the relevant path $p_a$;
 /* estimate/update the average RTT, average queueing delay and minimum RTT (as in simple FAST TCP [6]) for one of the multiple paths. */
```

(a)

```
On receiving a negative acknowledgement (a SACK containing gap reports) [Processing at sender side]:
    1) Let $p_i$ to be the complete set of paths from source to destination.
           $\forall$ paths $p_i$, initialize $p_i$.lookingfor_newack = FALSE;
    2) For each receipt of duplicate ACK (i.e. SACK containing gap reports) for a data packet $s_a$, thus far do
           let $p_a$ be the path on which a duplicate ACK for $s_a$ is received;    /* i.e. a path on which $s_a$ was originally sent. */
           set $p_a$.lookingfor_newack = TRUE;
    3) $\forall$ paths $p_i$, set $p_i$.largest_in_SACK_for_path to the largest sequence number being recently acknowledged on $p_i$;
    4) To check whether duplicate acknowledgement (dupACK) total for the data packet $s_a$ should be increased and
           then to determine whether missing data packet $s_a$ should be retransmitted:
       if ( $p_a$.lookingfor_newack == TRUE ) and ( $p_a$.largest_in_SACK_for_path > $s_a$ ) then
           (i)   increment dupACK count for $s_a$;
           (ii)  if ( dupACK count for $s_a$ == 3 (dupthresh) ) then
                     generate a loss signal for the data packet $s_a$ to remaining components of the
                     relevant path $p_a$ (i.e., to retransmit each unacknowledged data packet right away
                     or hold off until a more appropriate time);
           (iii) else if ( dupACK count for $s_a$ < 3 ) then
                     (a) set $p_a$.SACK_received_path = TRUE;
                     (b) estimate/update the average RTT, average queueing delay and minimum RTT (through SEC:
                         Step (3) in Fig. 9 (a)) for the path $p_a$ and provide this information to the other three
                         decision-making components;
       else dupACK count for $s_a$ is not incremented;
```

(b)

Fig. 9. SEC algorithm: (a) When a positive acknowledgement is received and (b) when a negative acknowledgement is received.

FAST's essential congestion control mechanisms are compactly summarized in Fig. 1, showing that FAST TCP includes four independent components, allowing each component to be modified individually and updated separately. We initially proposed the split estimation component (SEC) algorithms to implement multihoming options in FAST TCP.

Fig. 8 presents an architectural overview of the FAST TCP multihoming approach. Some modifications made on simple FAST TCP are as follows:

i) A path selector at the sender,

ii) a data dispatcher at the sender, and

iii) the virtual send/receive buffers at each sender/receiver end.

For each packet transmitted to a specific path selected by the path selector, the SEC records a time-stamp, and the burstiness control component updates corresponding data structures for accounting. At a constant time interval, that is checked by the SEC on the arrival of each acknowledgment over a specific path. At the sender end, it is the responsibility of data-dispatcher to assign the striped data, which are to be sent out over the paths selected by the path selector in a FAST TCP multihoming association.

A.1 Estimation Component for Multiple Paths—(SEC)

Algorithms: Estimation is one of the core components of the FAST TCP's congestion control mechanism, is capable of providing different input parameters estimations and passing them to the remaining three components.

In our proposed system, each path has its own four independent components (i.e., estimation, window control, burstiness control, and data control), as shown in Fig. 8. For example, consider that some data packets are sent through one of the multiple paths (i.e., *pa*); its SEC calculates two bits of feedback information for each data packet sent. On receiving a +ACK over the path (*pa*), it determines the RTT for the pertinent data packet and recalculates the minimum RTT and average queuing delay (including the virtual send buffer) for the path (*pa*). When a–ACK (on the receipt of dupACK $\leq$ 3 or timeout) is received, it checks and if the dupACK is less than three, it behaves normally (as on receiving a positive ACK); if dupACK is equal to three, it creates a loss signal for this data packet to the further components of the relevant path (*pa*) as given in proposed algorithm (Fig. 9(b)).

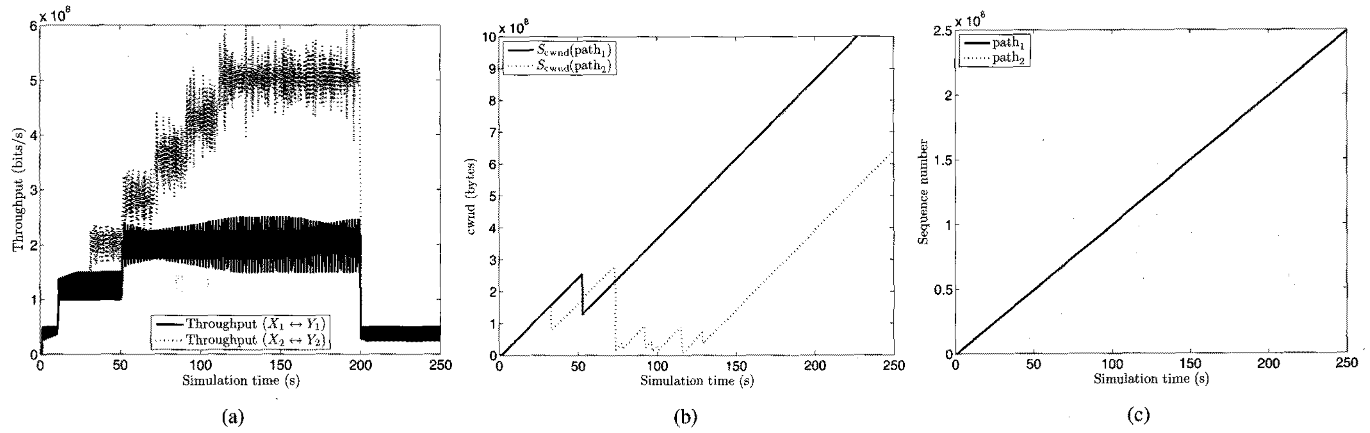


Fig. 10. FAST TCP multihoming evaluation (using the same topology shown in Fig. 5) under a bounded receive buffer (i.e., 1.2 Mb), with a fixed packet length of 1000 bytes: (a) Throughput achieved during the whole association, (b) CMT's sender (S) cwnds evolution over both the paths (path_1 and path_2), and (c) transmission sequence number progression over both the paths (path_1 and path_2).

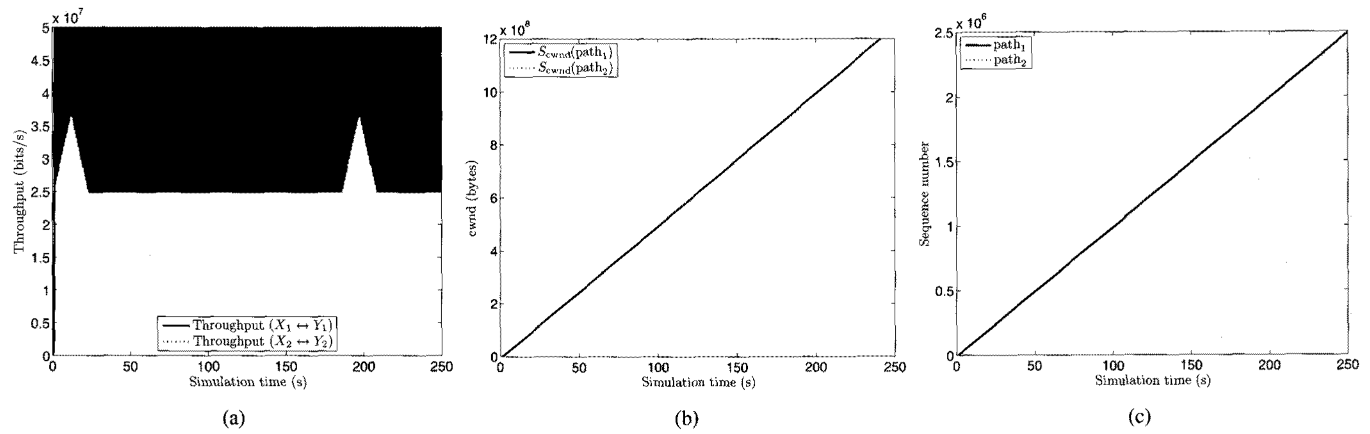


Fig. 11. FAST TCP multihoming evaluation (when no background traffic (Fig. 5)) under a bounded receive buffer (i.e., 1.2 Mb): (a) CMT's throughputs achieved over both the links, (b) CMT's sender (S) cwnds evolution over both the paths (path_1 and path_2), and (c) transmission sequence number progression over both the paths (path_1 and path_2).

We split the Estimation Component into two parts, one part is used to handle positive acknowledgement (on arrival of the SACK carries a new ACK) and other part is used to handle negative acknowledgment (on arrival of a SACK containing gap reports) through one of the multiple paths.

Algorithm details: Figs. 9(a) and 9(b) show the proposed SEC algorithms for multiple paths, which use SACKs, and determine the path on which each acknowledgement (positive or negative) is received. SEC computes the RTT for the pertinent data packet and recalculates the minimum RTT and average queueing delay for the path on which the SACK is received. SEC algorithms use three more variables for each path at a multihomed sender.

SACK_received_path: A variable employed for the period of a SACK-handling to deduce the contributory data packet(s)'s path(s) (i.e., used to determine on which path the SACK is received).

largest_in_SACK_for_path: This variable is used to store the largest sequence number for a data packer acknowledged per path by the SACK being processed.

lookingfor_newack: A variable employed for the period of a SACK-handling to deduce the contributory data packet(s)'s path(s) (i.e., used to determine on which path the SACK (containing gap reports) is received).

In Fig. 9(a), step (1) sets *SACK_received_path* to TRUE for the paths on which the SACKs (carries a new ACK) are received. Step (2) uses information collected in step (1) and estimates the different parameters values, that will be used to calculate the new congestion window size by the window control component for the path whose flag (i.e., *SACK_received_path)* value is TRUE.

In Fig. 9(b), step (1) assigns TRUE value to the *lookingfor_newacks* variable for the paths on which SACKs (containing gap reports) are received. For each path, step (2) determines the largest sequence number data packet being acked. In step (4), the details collected through steps (1) and (2) are exercised to deduce the absent data packet for which a dupACK is received.

B. Performance Evaluation and Assumptions

In this section, the performance of the FAST TCP multihoming is evaluated using the same topology shown in Fig. 5. We did not implement the initial negotiation phase of a FAST TCP multihoming connection since we believe this is not the limiting factor for end-to-end performance, and we are more addicted to the performance aspect of FAST TCP multihoming.

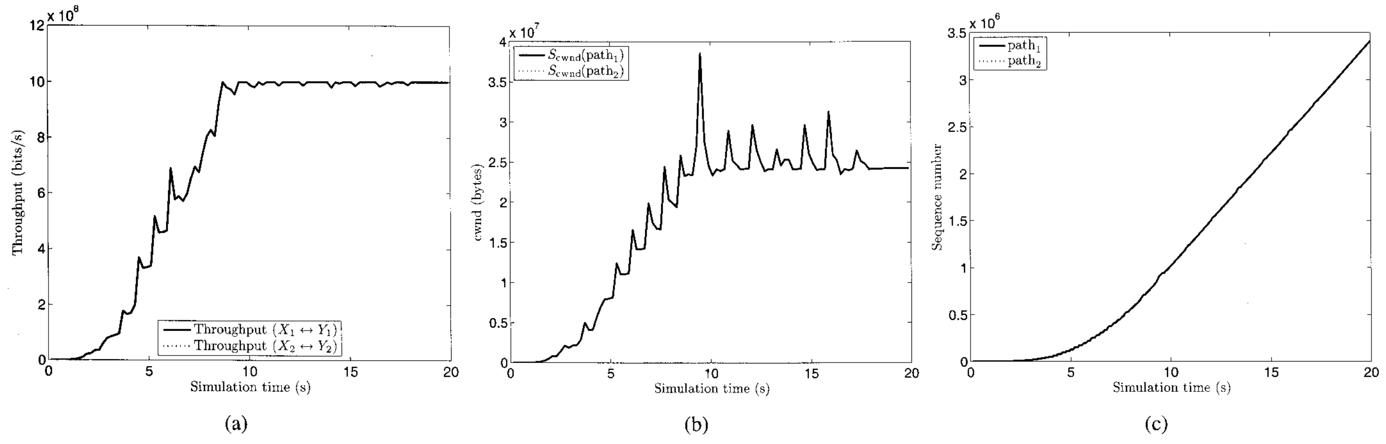


Fig. 12. FAST TCP multihoming evaluation (when no background traffic (Fig. 5)) with infinite receive buffer space, one-way propagation delay for each path = 100 ms and simulation time = 20 s: (a) CMT's throughputs achieved over both the links, (b) CMT's sender (S) cwnds evolution over both the paths ($path_1$ and $path_2$), and (c) transmission sequence number progression over both the paths ($path_1$ and $path_2$).

Figs. 10–12 show the trajectories (throughput, cwnd, and sequence number progression) of FAST TCP multihoming under varying network conditions. Referring to Figs. 7 and 10 (Section II-B), it can be observed that the two paths having different traffic-load distribution do not impact the relative performance of FAST TCP multihoming due to its delay-based congestion control mechanisms, while the SCTP-CMT multihomed sender, due to insufficient information, is unable to make informed assessments about the network congestion to adjust its sending rate so as to avoid losses, and hence, from rbuf blocking (i.e., it stops sending data (Fig. 7) at time around 202.8 s when 1.2 Mb rbuf is used). Referring to topology shown in Fig. 5, we also observed that SCTP-CMT multihomed sender stops sending data too early when using rbuf sizes of 1 Mb, 2 Mb, and 3 Mb or even with a reasonably large value (i.e., 10 Mb); rbuf blocking is not eliminated. However, in the case of FST TCP, when *packets are sent through multiple paths (having different traffic-load distribution) simultaneously to the destination using end-to-end multihoming, the packets are highly likely to arrive in the order in which they were initially sent (preventing rbuf from blocking).*

Referring to Figs. 11 and 12, it is possible to see that FAST TCP multihomed sender sends data up to the congestion window size value (as corresponding cwnds for each path allow) and then waits for ACKs to put more data onto the network by updating its congestion window on the basis of queuing delay according to (1). These results also verify that FAST TCP multihoming for end-to-end data transfer through multiple paths concurrently can effectively aggregate the bandwidth available on all the paths (i.e., two paths).

IV. CONCLUSIONS

In this research paper, we have proposed an end-to-end transport layer protocol (i.e., *FAST TCP multihoming* as a reliable, delay-based, SACK-based, and multihome-aware transport protocol) which can transfer data between a multihomed source and destination hosts through multiple paths simultaneously. We have redesigned the FAST TCP ns-2 module and proposed the algorithms to implement the end-to-end multihoming features into FAST TCP, with the goal of improving end-to-end throughput. We have described its architecture from design to implementation.

In this study, we have addressed two important topics: one is transport layer multihoming, which is essential for the aggregation of global routing table, and another is the long-fat-pipe problem, which more and more transport connections are suffering from. We demonstrated the weakness of SCTP-CMT rbuf constraints, and we then exposed that the rbuf blocking problem in SCTP-CMT multihoming was mostly due to its loss-based nature for detecting network congestion.

In this work, we have conducted simulations to illustrate the concept and to evaluate the performance of FAST TCP multihoming compared to SCTP-CMT in high-speed networks (ns-2), and via these simulations, we have shown that FAST TCP multihoming outperforms SCTP-CMT under similar network conditions. Therefore, we conclude that 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 that are not present in TCP and SCTP. The results of our initial efforts are encouraging, but there are several avenues for future work.

REFERENCES

[1] IBM. IBM grid computing home page. [Online]. Available: http://www.ibm.com/grid

[2] T. DeFanti, C. D. Laat, J. Mambretti, K. Neggers, and B. Arnaud, "TransLight: A global-scale lambdagrid for e-science," *Commun. ACM*, vol. 47, no. 11, Nov. 2003.

[3] M. Allman, V. Paxson, and W. Stevens. (1999, Apr.). TCP congestion control. IETF RFC2581. [Online]. Available: http://www.ietf.org/rfc/rfc2581.txt

[4] R. Stewart *et al.* (2000, Oct.). Stream control transmission protocol. IETF RFC 2960. [Online]. Available: http://www.ietf.org/ rfc/rfc2960.txt

[5] J. R. Iyengar, K. C. Shah, P. D. Amer, and R. Stewart, "Concurrent multipath transfer using SCTP multihoming," in *Proc. SPECTS, San Jose*, July 2004.

[6] D. X. Wei, C. Jin, S. H. Low, and S. Hegde, "FAST TCP: Motivation, architecture, algorithms, performance," to appear in *IEEE/ACM Trans. Netw.*, 2007.

[7] V. Jacobson, "Congestion avoidance and control," *ACM Comput. Commun. Rev.*, vol. 18, pp. 314–329, Aug. 1988.

[8] VINT Project. Network Simulator ns-2. [Online]. Available: http://www.isi.edu/nsnam/ns

[9] J. Iyengar, P. Amer, and R. Stewart, "Receive buffer blocking in concurrent

multipath transport," in *Proc. IEEE GLOBECOM, St. Louis, MO*, Nov. 2005.

[10] S. Floyd. (2000, Dec.). High speed TCP for large congestion windows. IETF RFC 3649. [Online]. Available: http://www. ietf.org/rfc/rfc3649.txt

[11] T. Kelly, "Scalable TCP: Improving performance in high-speed wide area networks," *Comput. Commun. Rev.*, vol. 33, no. 2, pp. 83–91, Apr. 2003.

[12] H. Bullot and L. Cottrell. TCP stacks testbed. [Online]. Available: http://www-iepm.slac.stanford.edu/bw/tcp-eval

[13] S. Hegde, D. Lapsley, B. Wydrowski, J. Lindheim, D. Wei, C. Jin, S. Low, and H. Newman, "FAST TCP in high-speed networks: An experimental study," in *Proc. GridNets, San Jose*, 2004.

[14] Y. Li, D. Leith, and R. N. Shorten, "Experimental evaluation of TCP protocols for high-speed networks," *IEEE/ACM Trans. Netw.*, vol. 15, no. 5, Oct. 2007.

[15] T. Cui and L. Andrew. FAST TCP simulator module for ns-2. version 1.1. [Online]. Available: http://www.cubinlab.ee.mu.oz.au/ns2fasttcp

[16] A. Caro and J. Iyengar. ns-2 SCTP module. [Online]. Available: http://pel.cis.udel.edu

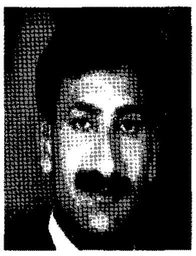
[17] J. Iyengar, P. Amer, and R. Stewart, "Retransmission policies for concurrent multipath transfer using SCTP multihoming," in *Proc. ICON*, Singapore, Nov. 2004.

[18] H. Hsieh and R. Sivakumar, "pTCP: An end-to-end transport layer protocol for striped connections," in *Proc. IEEE ICNP*, 2002.

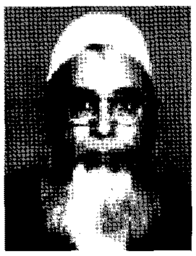
[19] A. Ishtiaq, Y. Okabe, and M. Kanazawa, "Issues of multihoming implementation using SCTP," in *Proc. The IASTED Int. Conf. Commun. Syst. Appl.*, July 2004.

[20] M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson, and R. Wang, "A transport layer approach for improving end-to-end performance and robustness using redundant paths," in *Proc. USENIX Annual Technical Conf., Boston, MA*, June 2004, pp. 99–112.

[21] M. Ohta, "The Architecture of End to End Multihoming," Internet-draft, IETF (Nov 2002), draft-ohta-e2e-multihoming-03.txt.



M. Junaid Arshad received his M.S. degree in 2000 and Ph.D. degree in 2009 in Computer Science from University of Engineering and Technology (U. E. T.), Lahore, Pakistan. Since 2001, he has been an Assistant Professor at the Department of Computer Science and Engineering, U. E. T., Lahore, Pakistan. His research interests include Internet protocols; multihomed networks focusing on performance, security, and mobility issues; congestion control, and wireless networks.



Mohammad Saleem received his B.E. degree in 1972 and M.E. degree in 1979 in Electrical Engineering from University of Engineering and Technology, Lahore, Pakistan and his Ph.D. degree in Electrical Engineering from the University of Manchester, U. K. in 1998. Since 1992, he has been a Professor at the Department of Electrical Engineering, U. E. T., Lahore, Pakistan, and is currently working as the Chairman of this department. His research interests include video streaming; multihomed networks focusing on performance, security, and mobility issues; digital image processing, and wireless networks. Publications of M. Saleem can be found in http://www.uet.edu.pk.