Issues of Multihoming Implementation Using FAST TCP: A Simulation Based Analysis

M. Junaid Arshad[†] and M. Saleem Mian[††]

[†]Department of Computer Science and Engineering, U.E.T., Lahore-Pakistan
[††]Department of Electrical Engineering, U.E.T., Lahore-Pakistan

Summary
Multihoming is currently widely used to provide end-to-end fault-tolerance and improved application performance. A node having different IP addresses or could be reached under several IP addresses is said to be multihomed. Transport layer multihoming is a feature that binds a single transport layer connection to multiple network addresses at each endpoint. Although transport layer multihoming is an old concept, neither of the Internet's current transport protocol workhorses, TCP or UDP, support multihoming. However, some transport protocols (like SCTP [10] or pTCP [17]) support multihoming, but unfortunately they are not designed for high-capacities and large-latencies networks, they often have performance problems transferring large data files over shared long-distance wide area networks. In this paper, we address a number of issues/challenges in attempting to develop such an end-to-end transport layer protocol based on FAST TCP, which can transfer data over the high-speed networks through multiple paths concurrently using multihoming (i.e., motivate to use FAST TCP as a reliable, multihome-aware, SACK-based and delay-based transport layer protocol). In our initial efforts, we propose some key design scheme for FAST TCP to enable the concurrent utilization of all the available paths over the multiple interfaces of a multihomed end host, with the goal of improving end-to-end throughput. After sketching this design scheme into 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 insight on design decisions for the future high-speed multihomed transport protocols.

Key words:
FAST TCP, Bandwidth Aggregation, FAST TCP multihoming, Multiple Paths, Transport Protocols

1. Introduction

Multihoming [1] is the ability of a host or site to access remote destination via more than one upstream connection, usually from different providers. The Internet may face a problem of physical failure at some specific location for a certain time due to any reasons. The user may switch to different ISP's to protect himself against such failure (link failures or overloaded links) because servers on the Internet today are recognized as being much less reliable. This has given rise to the idea of multihoming and a host is multihomed if it can be addressed by multiple IP addresses [18] as is the case when the host has multiple network interfaces.

Multihoming is a common requirement of many medium sized networks, including many businesses and ISP's, and can occur for two main reasons. One reason is for link redundancy, allowing a site to retain connectivity when one of the links fails. The other main reason is for optimal use of links, for example increasing bandwidth, or for quality of service (QoS) factors, or with a goal of improving end-to-end throughput.

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, and more home users are installing wired and wireless connections for added flexibility and fault tolerance. Multiple active interfaces also suggest the simultaneous existence of multiple paths between the multihomed hosts.

The existing TCP [11] and its different variants (such as HSTCP [4], Scalable TCP [2] and FAST TCP [5]) are not designed to manipulate multiple addresses in one TCP session. When a network outage occurs and the access-line associated with the local and remote addresses is down, the TCP session itself gets lost. However, some proposals have been suggested for TCP multihome options [3] [6] but they use multihome options for redundancy purpose only.

In this paper, we focus on FAST TCP for our investigation and our goal is to discuss the different issues involved for the implementation of FAST TCP multihoming for end-to-end data transfer through multiple paths concurrently and to suggest some practical guidelines for improving the end-to-end throughput. The general idea of multihoming implementation using FAST TCP in a network to obtain better performance and throughput has been explored in a number of different research efforts.

The remainder of this paper is organized as follows:

Manuscript received September 5, 2008.
Manuscript revised September 20, 2008.

In section 2, we first introduce FAST TCP along with its key features. In this section we also motivate FAST TCP used for multihoming by illustrating the advantages of replacing loss-based approach (i.e., like TCP [11] and SCTP [14]) with delay-based approach [5] as the end-to-end transport layer congestion control mechanism between two communicating hosts. In section 3, we discuss the various design issues and problems encountered in the basic implementation of end-to-end multihoming using FAST TCP. Section 4 delineates the proposed design and provides an overview of our approach, which is sketched in section 5 through simulations. In this section, we analyze the behavior of FAST TCP multihoming by using a particular network scenario in ns-2 (Network Simulator [9]), to prove its efficiency in producing high end-to-end throughput in multiple path environments. Finally in section 6, we present the conclusions and future work of this research.

2. FAST TCP Overview

FAST TCP [5] is a modification to the standard TCP [11] congestion control algorithm for high-speed long-distance connections. FAST TCP is a delay-based congestion control algorithm that aims to improve the performance of standard TCP. SCTP and TCP both use the packet loss as the measure of congestion (i.e., both rely only on packet loss to adjust congestion widow). But the FAST TCP uses both queuing delay and packet loss as signals of congestion, and departs from the loss based congestion control schemes [11][14]. FAST TCP uses queueing delay for congestion control and its advantage over loss-based approach is small at low speed, but decisive at high speed.

2.1 FAST TCP Congestion Control Algorithms

FAST TCP window calculate mechanism is divided into three sections: slow start (SS), multiplicative increase (MI), and exponential convergence (EC). SS is essentially identical to the standard slow start in TCP Reno; the only difference being that FAST exits SS when the number of packets queued in the network exceeds a threshold gamma rather than using packet loss. MI is used to rapidly move a FAST connection close to equilibrium whenever it falls below equilibrium. FAST TCP implements a safeguard mechanism in both MI and EC, where a window is increased or decreased on alternative RTTs. In EC, the window moves half-way between the current value and the target in each update interval so it is exponentially increasing, with a negative exponent so the window converges to the target, with time measured in multiples of 10 ms.

FAST TCP updates the window based on the following algorithm:

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

Where $\gamma \in (0,1]$, baseRTT is the minimum RTT observed so far, and qdelay is the end-to-end (average) queuing delay. The constant α is the number of packets each flow attempts to maintain in the network buffer(s) at equilibrium.

2.2 Related Work and Motivation for FAST TCP multihoming implementation

This section describes some prior work that exploit transport layer multihoming (i.e., implements a transport layer solution to aggregate bandwidth across multiple end-to-end paths) and differentiates our work from this earlier research by motivating the FAST TCP used for end-to-end multihoming.

SCTP is a transport protocol that introduces support for simultaneous data transfer over multiple paths in multihomed hosts. 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 [10]. We used FAST TCP for our reference multihoming implementation and investigation, because FAST TCP has a great advantage over other transport layer protocols (e.g., SCTP [14] and TCP [11]), it uses queueing delay, rather than loss probability as the main measure of congestion, and it is intended to solve the transport layer protocols limitation in high-bandwidth large-delay environments.

The loss-based transport protocols (i.e., SCTP and TCP) detect congestion only after a packet has been dropped at the gateway (i.e., use packet loss as the measure of congestion), which means the source will not know the situation in the gateway until congestion occurs. Another problem with concurrent multipath transfer using SCTP [10] is that, it is more sensitive to receiver buffer (rbuf) constraints [16], and this rbuf-blocking problem causes significant throughput degradation when multiple paths are used concurrently. In [13], 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 have also shown that FAST TCP consistently outperforms SCTP in terms of throughput, stability with zero packet loss at the bottleneck under a similar network conditions, because FAST TCP anticipates the onset of congestion by monitoring the difference between the rates it is expecting to see and the rate it is actually realizing. 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. Therefore, 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.

3. FAST TCP Multihoming Implementation Issues

In this section, we address several open questions and issues concerning the end-to-end multihoming implementation using FAST TCP. We will also suggest the solutions of some issues that have been explored in a number of different research efforts.

Some issues for the implementation of FAST TCP multihoming are as follows:

- How the use of FAST TCP multihoming for multiple paths increases aggregate throughput?
- How to determine the number of paths used that is necessary to maximize throughput while avoiding network congestion?
- Scheduling of traffic on multiple paths.
- Congestion Control: separate or shared?
- How to use receiver's advertised window (rwnd) at the sender for multiple paths: shared buffer or individual buffer for each path?
- How the sequence space shares among flows on different paths that occur within an association?
- Loss detection and recovery.
- Packet reordering introduced by the sender over multiple paths.
- Reverse path for acknowledgements.

Now, we address some of these issues in details.

3.1 Congestion Control and Flow Control Issues

The important issue that we want to address for end-to-end transfer of data through multiple paths using FAST TCP multihoming is its congestion control and flow control mechanisms. Congestion control is a critical issue for FAST TCP multihoming implementation as it tries to utilize the network resources more aggressively. Since the key obstacle for achieving the aggregate bandwidth for FAST TCP multihoming through multiple paths concurrently is that each of the individual paths can have vastly differing characteristics in terms of bandwidth and delay (round-trip time). If we ignore the factors for implementing the propose system, the bandwidth achieved through multiple paths can be significantly lower than the maximum possible. As different sub-flows (paths) take different network paths, so each path needs to have its own congestion control.

FAST TCP multihoming for multiple paths strives to keep all paths independent from each other. Suppose we had used only one global congestion window for the entire flow. The packet losses on any one of the paths will cause the global congestion window to be halved, thus affecting the all sub-flows (paths). If one sub-flow happens to go across a heavily congested path, it can keep the global congestion window small, and the other sub-flows will not be able to utilize the available bandwidth on other good paths. In certain situations, this can cause the throughput of the whole flow to be even lower than that of a single-path FAST TCP flow on a single good path. This fact was studied in [12]. Thus all the paths should share the same send/receive buffer as well as in this proposed system. Packets are assigned sequence numbers in the same way as in FAST TCP. But how the sequence space shares among these multiple paths? We will discuss this issue in section 4. Each path does its congestion control and maintains a congestion window independently as in simple FAST TCP [5]; i.e., congestion window changes independently as the sub-flow adapts to the network state and each sub-flow uses equation-based control with queueing delay (according to Eq. (1)), and multiplicative decrease with packet loss (when a packet loss is detected, FAST TCP halves its window and enters loss recovery).

3.2 Receiver's Advertised Window (rwnd) at the Sender for Multiple Paths

A transport layer receiver maintains receiver buffer space for containing data for two reasons: (i) to handle out-of-sequence data, and (ii) to receive data at a rate higher than that of the receiving application's consumption. In TCP, a receiver advertises currently available rbuf space through window advertisements (normally by ACKs) to a data sender. This value is the advertised receive window. A sender computes a peer-receiver window (rwnd) to deduce how much more data can be buffered at the receiver.

Now the issue is that how to use rwnd at the sender for multiple paths concurrently? It should be clear that each sub-flow (path) should have its own retransmission timer since each has its own round trip time (RTT) and of course has its own congestion window per destination. They represent the state of different network paths from a sender to each destination address. A sender has no reason to maintain separate rbufs or peer-rwnds per path since a receiver can consume data only in-sequence, irrespective of the destination address they are sent to. Thus a FAST TCP multihoming for multiple paths receiver should

maintain a single rbuf which is shared across all sub-flows in an association. In this way a sender divides the global advertised window among all its sub-flows (paths) proportional to their congestion window sizes and this will always yield better throughput. On the other hand if the advertised window is divided among sub-flows (paths) on the multihomed receiver side, this buffer sharing degrades overall throughput.

When multiple paths being used for simultaneous transfer of data have different delay and/or bandwidth characteristics between source and destination, additional packet reordering is observed at the receiver. In other words significant packet reordering can be introduced in the flow by a multiple path data transfer sender, which in turn can cause rbuf-blocking problems [13].

3.3 Sequence Space among the Multiple Paths and Reverse Path for ACKs

As we have discussed in the previous section that all sub-flows share the same send/receive buffer and each path has its own congestion control. Also, each sub-flow estimates its own round trip time, it needs to remember which packets it has sent and match each received ACK with one of the these packets. Thus each sub-flow also maintains the sequence numbers of the packets sent from this path but not acknowledged yet. This sequence space may or may not be shared among all the paths.

Since ACKs are cumulative, so sharing of sequence space across paths can help a multihomed sender for receiving ACK information on either of the return paths. Thus a shared sequence space for end-to-end transfer of data through multiple paths can effectively use both (if two paths are used) return paths for communicating ACK information to the sender. But at the same time the disadvantage of using different reverse paths for ACKs can make our system more complex, because the multihomed receiver has to maintain additional states about which ACK going through which paths. Moreover using several reverse paths will introduce ACK reordering, which in turn will affect the sender's behavior (e.g., increasing the burstiness of the sender).

4. FAST TCP Multihoming Proposed Design

In this section, we present some key design elements for FAST TCP multihoming implementation through multipath data transfer concurrently. It is important to propose the design scheme in such a way that it will provide the same semantics to applications as simple FAST TCP– it will preserve the properties such as reliability, fairness, stability and responsiveness. FAST TCP multihomed sender host sends data through multiple paths concurrently, so it has to decide how to schedule packets across the multiple paths and how to manage congestion control for each path (sub-flow).

In the previous sections, we have discussed that the key obstruction for achieving the aggregate bandwidth through multiple paths concurrently is that each of the individual paths can have immensely different characteristics in terms of bandwidth and delay (round-trip time). If we ignore these factors for implementing the propose system, the bandwidth achieved through multiple paths can be considerably lower than maximum possible. As different paths have different delay and/or bandwidth characteristics, so each path needs to have its own *window control* mechanism, like a simple FAST TCP. This means each path needs its own *estimation* module [5] (like FAST TCP) to estimate round-trip time (RTT) and remember which packets it has sent and match each received ACK with one of there packets.

Each path maintains a congestion window as in simple FAST TCP and this congestion window changes independently as the path adapts to the network state (i.e., under normal network conditions, FAST TCP periodically updates the congestion window based on the average RTT and average queuing delay provided by the estimation component, according to Eq. (1)).

In our current design, we use a single sequence space (used for congestion control and loss detection, and recovery) across an association's multiple paths. 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 3-duplicate acknowledgements) report the missing data packets from the same virtual queue.

In our current prototype, the FAST TCP multihomed receiver maintains and controls a single rbuf, which is shared across all sub-flows (paths) in an association. In this way the FAST TCP multihomed sender divides the global advertised window among all its sub-flows (paths) proportional to their congestion window sizes. We identify through extensive simulations that for N (sub-flows) paths the FAST TCP multihomed receiver roughly needs the maximum buffer size to be $3(\alpha 1 + \alpha 2 + \alpha 3 + ... + \alpha N)$ packets (not a strong conclusion), with each path injecting α packets (corresponding to their links capacities) into the network towards its destination, where alpha (α) is the minimum number of packets, each path aims to maintain in the network queue. Ideally, if the link capacity of a path is C packets/ms then we set alpha to equal to $2C$ so that the source generates 2ms of queueing delay for that path. In practice C is not known, so we set alpha to a value that

works well for a range of capacities expected. With alpha tuning enabled (i.e., *tcp_fast_at_sec* [8] not 0), this alpha value is ignored and an automatic alpha estimation is used. With alpha tuning, the maximum throughput achieved is detected, and the alpha value is set according to whether it is a low, medium or high speed environment. For example, if α_1 =100 packets and α_2 = 150 packets are fixed for the two paths ($path_1$ and $path_2$) respectively, then FAST TCP multihoming would not effectively aggregate the bandwidth available on two independent paths and reach equilibrium if the receiver buffer size is much less than $3(\alpha_1+\alpha_2)$ or 3(100+150) = 750 packets or (750KB).

FAST TCP is a delay-based congestion control algorithm. It departs from the loss-based congestion control schemes [4], [15] and [14], but it reacts primarily to queuing delay with the ability to anticipate congestion, and adjust its transmission rate accordingly to inform the source that the network will be congested. Since on the reception of an acknowledgement, FAST TCP updates its congestion window based on the queuing delay i.e., `qdelay = avgRTT - baseRTT` according to Eq. (1). It means, we must maintain a steady stream of ACKs (with selective acknowledgement) from new transmissions in order to obtain unambiguous RTT measurements. This is generally possible only when all ACKs return over the same path to which they were originally sent. Thus in our design, although the data packets are sent through multiple paths concurrently, all SACKs return over the same path to which they were originally sent. But despite its many desirable properties, FAST TCP multihoming would have throughput problems if network congestion builds up in the reverse direction on the destination host to source host. This occurs because FAST TCP uses the round-trip time to sense congestion, which includes reverse path congestion. Thus in our future work, we will effort for its solution and will include one-way congestion measurement, so that FAST TCP multihoming only reacts to forward path congestion.

5. Applicability of FAST TCP for Multihoming Implementation

In this section, we investigate the applicability of FAST TCP for end-to-end multihoming implementation. For this, we used ns-2 [9] network simulator as the basis for our investigation. We used FAST TCP simulator module for ns-2 [8], *version 1.1 (SACK introduced).* This fast-tcp-ns2-v1_1c patch was developed at the University of Melbourne's Centre for Ultra-Broadband Information Networks (CUBIN) and was written by Tony Cui with advice from Lachlan Andrew and others. For our multihoming implementation using FAST TCP, we

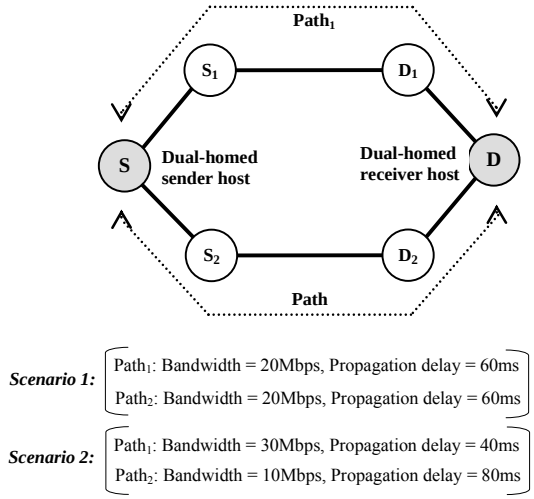


Fig. 1. Simulation topology used for FAST TCP multihoming evaluation

modified this ns-2 module [8] by sketching all the key design elements that we have discussed in the previous sections. We describe the setup of our experimental evaluation of FAST TCP multihoming performance in terms of application throughput during file transfers through multiple paths concurrently.

5.1 Experimental Setup and Network Topology

In this section, we discuss the simulation parameters and assumptions, network topology and evaluation metrics used. We did not implement the initial negotiation phase of a FAST TCP multihoming connection, as we believe this is not the limiting factor of end to end performance and we are more interested in the performance aspect of FAST TCP multihoming. Fig. 1 illustrates the network topology used in our experiments.

Two multihomed hosts, a sender host (S) having local addresses S_1, S_2 and a destination host (D) having local addresses D_1, D_2 are connected by two independent paths. We used two types of links in our simulations to show the FAST TCP multihoming behavior: (i) two paths $S_1 \leftrightarrow D_1$ and $S_2 \leftrightarrow D_2$ are connected as duplex links of 20Mbps with a one-way propagation delay of 60ms. Thus the roundtrip propagation delay on both paths is 120ms and both paths have exactly the same characteristics (i.e., delay, bandwidth), and this scenario is described in (section 5.2.1), (ii) two paths $S_1 \leftrightarrow D_1$ and $S_2 \leftrightarrow D_2$ are connected as duplex links having end-to-end available bandwidths 30Mbps and 10Mbps with a one-way propagation delay of 40ms and 80ms, respectively. Thus the roundtrip propagation delays on both paths are 80ms and 160ms, respectively and this scenario is described in (section 5.2.2).

We simulated the end-to-end paths from the multihomed host (S) to the destination (D) through our custom bandwidth, queuing delay, aggregate congestion window evolution and loss modules added to the link object in *ns-2*. Multihomed sender host (S) sends data to destinations D_1 and D_2 concurrently, as the bandwidth becomes available on corresponding paths, i.e., as corresponding congestion windows allow. In all of our experiments, alpha (α) was set to 100 for each path and packet size was set to 1000 bytes. We transferred an infinitely large file using FTP from S to D for 30 seconds with the maximum receiver buffer size of 600 packets (or 600KB) and drop-tail queuing.

5.2 Simulation Results and Analysis

We now present our experimental results and following experimental scenarios were investigated:

- *FAST TCP multihomed hosts with same path characteristics (i.e., bandwidth and delay): Scenario 1*
- *FAST TCP multihomed hosts with different path bandwidth and delay: Scenario 2*

All the two scenarios are based on the simulation topology shown in Fig. 1.

Before we look at the FAST TCP multihoming scenarios (1, 2) in detail, we first wanted to see how simple FAST TCP (FAST TCP without multihoming options) congestion control mechanisms handle and update the (single congestion window for two paths) window when applied to multiple path environments. For this purpose, we conducted a simple simulation also based on the topology shown in Fig. 1, in which both paths ($S \leftrightarrow S_1 \leftrightarrow D_1 \leftrightarrow D$ and $S \leftrightarrow S_2 \leftrightarrow D_2 \leftrightarrow D$) from the source host (S) to the destination host (D) are used at the same time and each host has a single local address. Both data packets and acknowledgements of the same FAST TCP connection are scattered over both paths (i.e., $path_1$ and $path_2$) having different path characteristics. Both paths ($path_1$ and $path_2$) are connected as duplex links having end-to-end available bandwidths 30Mbps and 10Mbps respectively.

Fig. 2 shows the throughput achieved by the FAST TCP in the first few seconds of the connection life over the two paths having 60ms propagation delay (i.e., 120ms round-trip time) for each path. Similarly, Fig. 3 shows the throughput achieved by the FAST TCP over the two paths, when $path_1$ and $path_2$ have 40ms and 80ms propagation delays (similar to scenario 2) respectively.

The results in Fig. 2 show that the achieved aggregate bandwidth of FAST TCP over two equal propagation delays paths is 20Mbps, which is theoretically twice the bandwidth of $path_2$ (smaller bandwidth path of the two paths). This achieved aggregate bandwidth is much

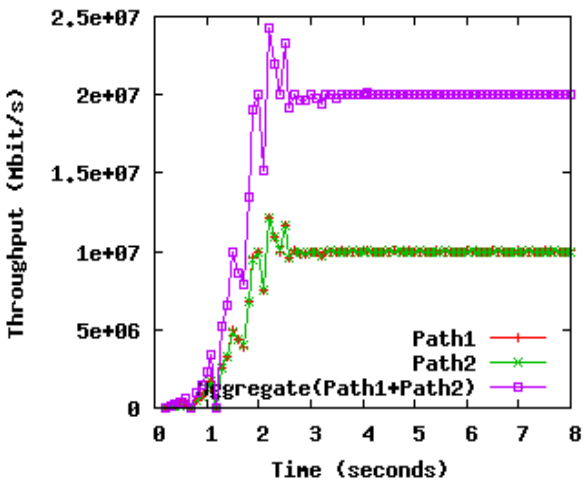


Fig. 2. Throughput achieved by simple FAST TCP over the two paths when both paths ($path_1$ and $path_2$) have the same 60ms propagation delay.

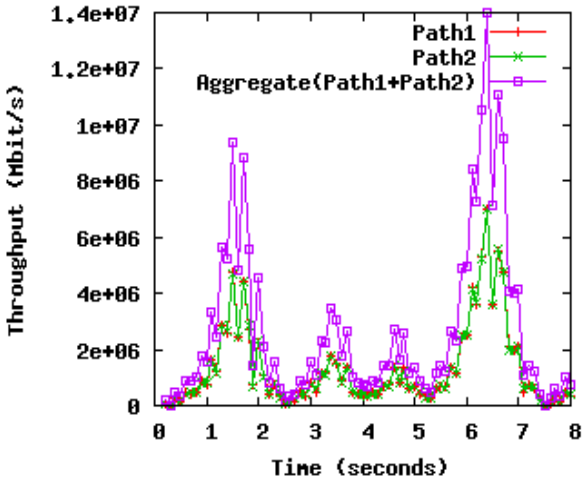
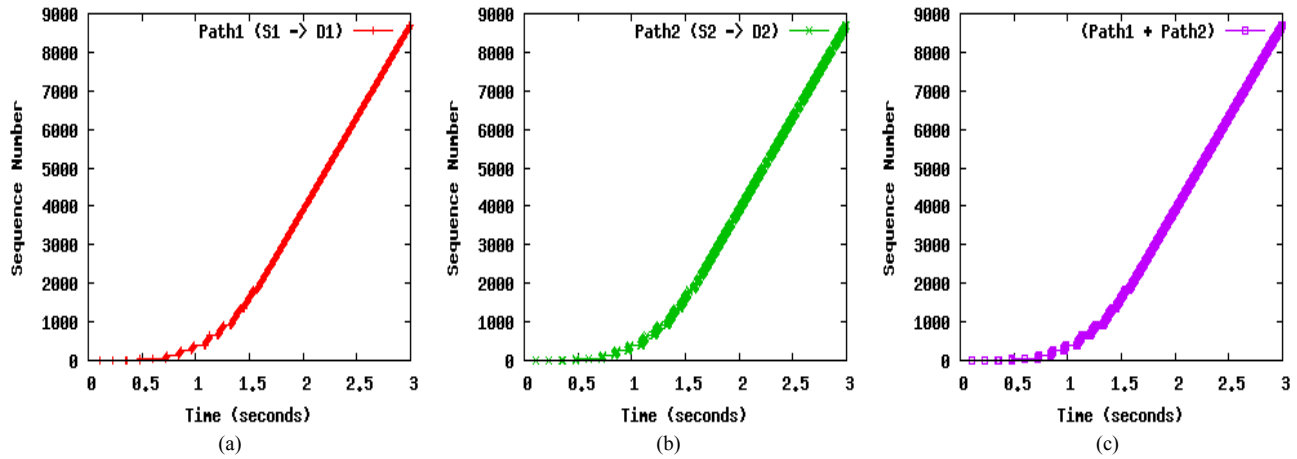


Fig. 3. Throughput achieved by simple FAST TCP over the two paths when $path_1$ and $path_2$ have 40ms and 80ms propagation delays, respectively.

smaller than the expected aggregate bandwidth of two paths (i.e., $path_1$ (30Mbps) + $path_2$ (10Mbps)). Similarly, the results in Fig. 3 show that the achieved aggregate bandwidth of FAST TCP over two different propagation delays paths is smaller than that of a single-path FAST TCP flow on a single good path.

To explain these surprising results (Fig. 2 and Fig. 3), we remind the reader that FAST TCP updates its congestion window based on the queueing delay and the average queueing delay is estimated as $\mathtt{qdelay} = \mathtt{avgRTT} - \mathtt{baseRTT}$, where baseRTT is the propagation delay and avgRTT comes from exponentially



(a) (b) (c)

Fig. 4 Simulation scenario 1: FAST TCP multihoming Sequence Number Plot (a close-up look at the first 3 seconds of the simulation) (a) for path1 ($S_1 \leftrightarrow D_1$), (b) for path2 ($S_2 \leftrightarrow D_2$) and (c) for both destinations (D_1, D_2).

averaging instantaneous RTT samples, as (for Fig. 2) both paths have the same 60ms propagation delay, resulting in qdelay to zero and FAST TCP performs multiplicative increase and grows exponentially at a rate α (100) until it reaches to (2 * bandwidth of $path_2$). Also at the same time, FAST TCP uses a single global congestion window for the entire flow (two paths) that basically causes the $path_1$ not be able to utilize its available bandwidth. Same case is with the Fig. 3 result, as both paths have different propagation delays, the acknowledgements coming from $path_1$ (having 80ms RTT) and $path_2$ (having 160ms RTT) causing a large queuing delays for updating the FAST TCP's single global congestion window. As queueing delay is the dominant congestion signal for FAST TCP to adjust its window, thus it forces the FAST TCP to decrease its window to unnecessarily small values.

The problems that arise due to bandwidth differences and/or due to delay differences over the multiple paths data transfer can be solved by performing the independent congestion control mechanism for each path separately (i.e., each path maintains a congestion window as in simple FAST TCP).

Now we move on to the above two cases and look at these two scenarios (1, 2) in detail, comparing not only the throughput behavior by implementing the proposed multihoming design scheme into FAST TCP, but also the queue behavior inside the network, by examining trajectories of throughputs, windows, instantaneous queue and link utilization. In particular, we wanted to see how well FAST TCP multihoming handles congestion control mechanisms and packet reordering in presence of multiple paths.

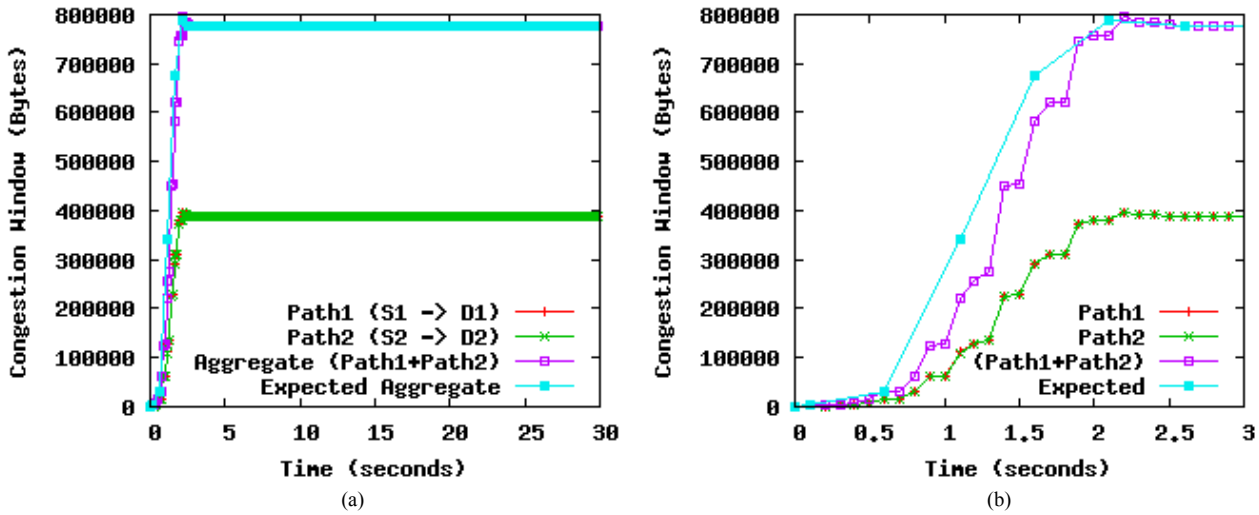
5.2.1 Scenario 1

In the first scenario, we conducted simple simulation to investigate how much FAST TCP multihoming improves throughput using multiple independent paths under certain conditions. Although this simulation scenario does not justify some effects seen in the Internet and other real networks such as network induced reordering and delay. However, for an idealize case, we selected a simple topology to avoid influence of these effects, and to present the best possible performance expected by an application that strips data over multiple paths concurrently. The bandwidths were chosen to be high enough because FAST TCP can sustain high throughput for extended periods of time in high-speed long-distance networks using standard packet size.

Fig. 4 shows the sequence number plots of the sent packets verses the time at which they were sent. Y-axis represents the sequence number plot and X-Axis shows the time when it was transmitted. Fig. 4 has a single curve, which shows the FAST TCP multihomed sender host's sequence number plots of the sent packets through both paths.

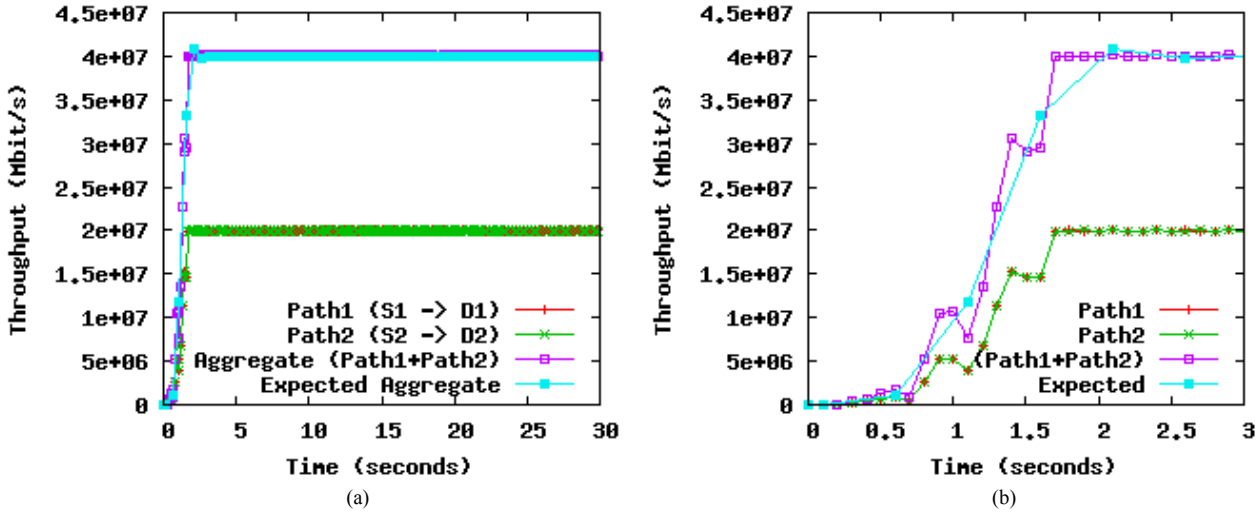
The plots show the way multihomed sender normally sends packets, when no loss is involved. It is possible to see that FAST TCP multihomed sender sends data up to the congestion window size value (as corresponding cwnds for each path allow) and then waits for ACKs to put more data onto the network by updating its congestion window based on the queuing delay according to Eq. (1).

Similarly, Fig. 5 shows congestion window evolution over time for the whole association. The plots in Fig. 5 show the FAST TCP multihomed sender's observed cwnd evolution for each destination for this association, and compare the calculated aggregate cwnd evolution (sum of



(a) (b)

Fig. 5 Simulation scenario 1: (a) Evolution of the different Congestion Windows (FAST TCP multihoming) (b) A close-up look at the first 3 seconds of the simulation.



(a) (b)

Fig. 6 Simulation scenario 1: (a) End-to-end throughput achieved by FAST TCP multihoming through multiple paths (two paths) data transfer concurrently under equal path delays (path1 = 60ms, path2 = 60ms) (b) A close-up look at the first 3 seconds of the simulation.

both paths ($path_1$+$path_2$)) with the expected aggregate cwnd evolution (the expected aggregate cwnd is the sum of the cwnd growth of two independent FAST TCP runs). We observed that FAST TCP multihomed sender's aggregate cwnd growth came close to the expected aggregate cwnd growth. This observable fact shows that the performance gains of parallel transfer through multiple paths can be fully achieved by using a sharing sequence space among flows on different paths that occurs within a single association.

We presented the throughput results in Fig. 6. The curves in Fig. 6 compare expected aggregate throughput (the expected aggregate throughput is the sum of the throughputs of two independent FAST TCP runs) with the aggregate throughput actually achieved by the FAST TCP multihomed sender when using both paths ($path_1$ and $path_2$) for end-to-end data transfer concurrently. We observed that FAST TCP multihoming achieved the aggregate throughput near the ideal performance (i.e., expected aggregate bandwidth). So this is exactly what we expected to see from the simulation: the throughput climbed nearly to 40Mbps by implementing the proposed multihoming design scheme into FAST TCP.

In fact this is because, a FAST TCP multihomed sender

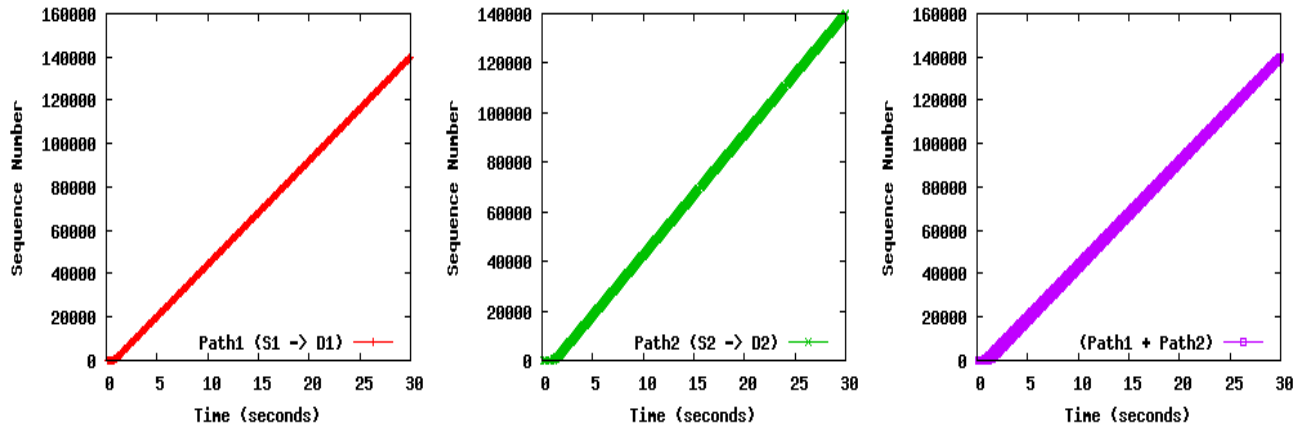


Fig. 7 Simulation scenario 2: FAST TCP multihoming Sequence Number Plot (a close-up look at the first 30 seconds of the simulation) (a) for path1 ($S_1 \leftrightarrow D_1$), (b) for path2 ($S_2 \leftrightarrow D_2$) and (c) for both destinations (D_1, D_2).

during the association, simultaneously opens up a separate congestion window for the each path, so that it is nearly close to the expected aggregate throughput.

These results (Fig. 4 – Fig. 6) 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). From this experiment, some other information was also gathered that higher throughput can be achieved when congestion control is performed for each path separately.

5.2.2 Scenario 2

The exact same setup (Fig. 1) was used for this scenario except that the both paths have the different characteristics (i.e., bandwidth and delay) as previously described in (section 5.1). In this second test, we tried to evaluate how FAST TCP multihomed hosts determined congestion window based on congestion information (queueing delay and packet loss) and reacted to packet reordering[1] due to different path characteristics for end-to-end data transfer through multiple paths concurrently. Thus for this scenario, we increased the delay of link $S_2 \leftrightarrow D_2$ from 60ms to 80ms and bandwidth of link $S_1 \leftrightarrow D_1$ from 20Mbps to 30Mbps, and we decreased the delay of link $S_1 \leftrightarrow D_1$ from 60ms to 40ms and bandwidth of link $S_2 \leftrightarrow D_2$ from 20Mbps to 10Mbps; as a result one path from S to D now has a total delay 40ms greater and a total bandwidth 20Mbps smaller than the other (Fig. 1). The packet loss was set to 0%.

[1] The packets 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 [13]).

Fig. 7 corresponds to the sequence number progressions of an experiment performed following this scenario. Through extensive simulations, we observed that unequal delay on the two paths do not impact the relative performance of FAST TCP multihoming. Fig. 7 – Fig. 9, demonstrate this consistent behavior with different end-to-end link capacities and unequal path delays of 40ms on $path_1$, and 80ms on $path_2$.

We also examined that these outcomes are consistent with the results obtained from scenario 1 (Fig. 4 – Fig. 6), which have equal delays of 60ms on both paths. Specifically, as the bandwidth ratio between multiple paths increases in the network, stability becomes worse for the loss-based protocols, produces more out-of-sequence packets at the receiver, causing lots of retransmissions and blocking the rbuf, which impair the whole association throughput. But we observed that the performance of FAST TCP multihoming under a single sequence space within a transport layer association did not degrade in any significant way for any evaluation criterion and its sketched design scheme for parallel data transfer through multiple paths using end-to-end multihoming remain quite stable and insensitive to the path characteristics changes.

6. Conclusion and Future Work

In this research work, we have addressed a number of issues 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. We have also proposed a design scheme for multihoming implementation using FAST TCP by taking advantage of its delay-based features with the goal of improving end-to-end throughput.

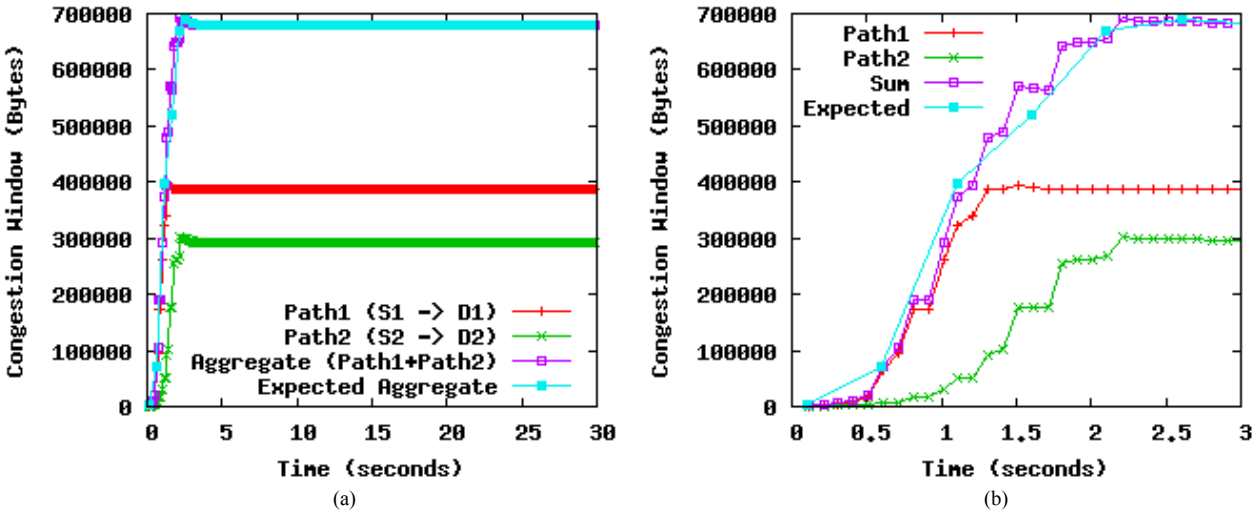


Fig. 8 Simulation scenario 2: (a) Evolution of the different Congestion Windows (FAST TCP multihoming) (b) A close-up look at the first 3 seconds of the simulation

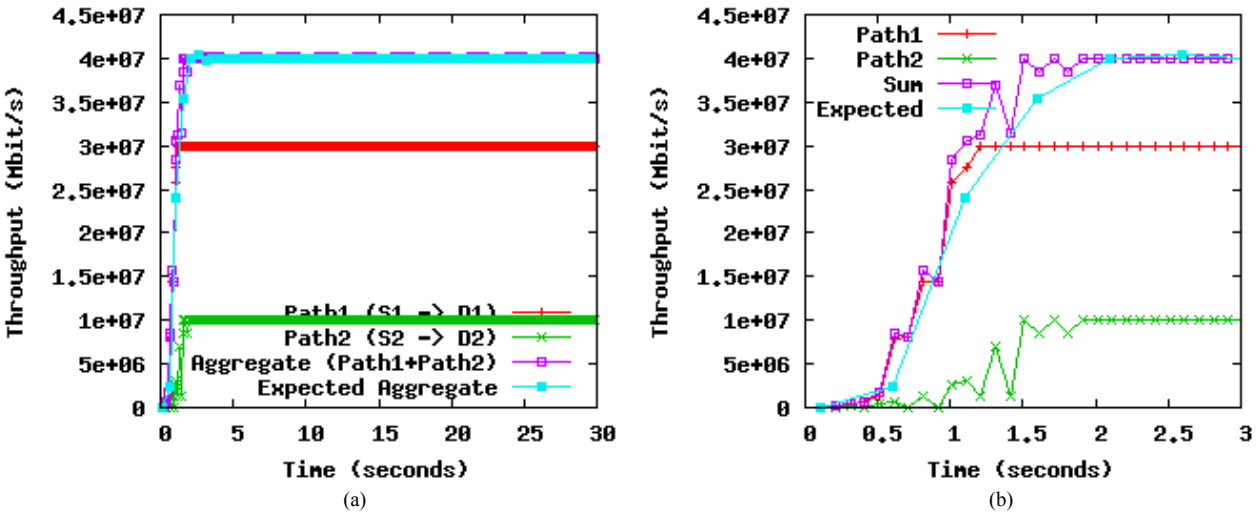


Fig. 9 Simulation scenario 2: (a) End-to-end throughput achieved by FAST TCP multihoming under different link capacities (path1 = 30Mbps, path2 = 10Mbps) and unequal path delays (path1 = 40ms, path2 = 80ms). (b) A close-up look at the first 3 seconds of the simulation.

In this study, two scenarios were sketched to illustrate the concept and to evaluate the proposed design scheme of FAST TCP multihoming on two high-speed networks (*ns-2*), and through these simulations we have shown that FAST TCP multihoming achieves bandwidth aggregation efficiently under a variety of network conditions.

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 not present in current TCP and SCTP. Since data transfer through multiple paths is becoming an increasingly popular topic of research and mechanism for using extra network links, so we also conclude that the issues considered in this research provide insight on design decisions for future multihomed transport protocols. The results of our initial efforts are encouraging but there are several avenues for future work.

In this paper, we focus on evaluation of proposed design scheme through analysis and simulation, however, much remains to be done to make it a practical scheme for implementation in the networks, thus we will first modify or re-design the FAST TCP algorithms [9] for the ns-2 to perform the simulation work by evaluating these design decisions.

Acknowledgments

The authors would like to express their deep gratitude to Ishtiaq A. Choudhry, whose continuous criticism helped to shape up the idea, and the special thanks go to Masanori Kanazawa for the ongoing moral support. This research was supported by the Directorate of Research Extension and Advisory Services U.E.T., Lahore-Pakistan.

References

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

[2] Tom Kelly, "Scalable TCP: Improving performance in highspeed wide area networks," Submitted for publication, http://www-lce.eng. cam.ac.uk/~ctk21/scalable/, December 2002.

[3] A. Matsumoto, M. Kozuka, K. Fujikawa, Y. Okabe, "TCP Multi-Home Options," Internet-draft, IETF (Oct 2003), draft-arifumi-tcp-mh-00.txt. (work in progress).

[4] 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.

[5] 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.

[6] M. J. Arshad and M. S. Mian, "Simulation and Visualization of Transmission Control Protocol's (TCP) Flow-Control and Multi-Home Options," in *Proceedings of IEEE IBCAST*, Islamabad, Pakistan, 8th - 11th January, 2007.

[7] Van Jacobson. Congestion Avoidance and Control. *InACM SIGCOMM,* 1988.

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

[9] VINT Project, Network Simulator *ns-2,* http://www.isi.edu/nsnam/ns/.

[10] 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.

[11] M. Allman, V. Paxson, and W.Stevens,"TCP Congestion Control," RFC2581, IETF, April 1999, http://www.ietf.org/rfc/rfc2581.txt.

[12] 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 Conference, Boston, MA, June 2004, pp. 99–112.

[13] M. J. Arshad and M. S. Mian, "Experimental Study of FAST TCP compared to SCTP:", Submitted for publication 2008.

[14] R. Stewart et al. Stream control transmission protocol. IETF RFC 2960, Oct. 2000.

[15] Kevin Fall and Sally Floyd, "Simulations–based comparisons of Tahoe, Reno and SACK TCP," *ACM Computer Communication Review*, vol.26, no. 3, pp. 5–21, July 1996.

[16] J. Iyengar, P. Amer, and R. Stewart. Receive Buffer Blocking In Concurrent Multipath Transport. In *IEEE GLOBECOM*, St. Louis, Missouri, November 2005.

[17] H. Hsieh and R. Sivakumar. ptcp: An end-to-end transport layer protocol for striped connections. In *Proceedings of IEEE ICNP*, 2002.

[18] R. Braden. Requirements for Internet Hosts – Communication Layers. RFC1122, IETF, October 1989.

M. Junaid Arshad received his Master degree in Computer Science in 2000. Since 2001, he has been an Assistant Professor in the Department of Computer Science and Engineering, U.E.T., Lahore-Pakistan. His master thesis received the outstanding thesis award of the faculty of Computer Science, in 2000. He is currently a PhD student in the Department of Computer Science and Engineering, at University of Engineering & Technology, Lahore-Pakistan. His research interests include Internet protocols, multihomed networks focusing on performance, security and mobility issues, congestion control and wireless networks.

M. Saleem Mian received the B.E. degree in 1972 and M.E. degree in 1979 in Electrical Engineering from University of Engineering & Technology Lahore, Pakistan and PhD degree in Electrical Engineering from University of Manchester U.K. in 1998. Since 1992, he has been a Professor in the Department of Electrical Engineering, U.E.T., Lahore-Pakistan and currently working as a Chairman in 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 Mian can be found in http://www.uet.edu.pk.