Unified Transport Layer Support for Data Striping and Host Mobility

Tom Goff, *Student Member, IEEE,* and Dhananjay S. Phatak, *Member, IEEE*

***Abstract*— Data striping across multiple network interfaces and its applications to mobile environments was recently investigated in [1]. Therein, a network-layer IP-in-IP encapsulation mechanism was proposed to aggregate the bandwidth of multiple network paths by striping a single transport layer (TCP) connection across them. The analysis and experimental results from that study demonstrated fundamental limitations to TCP's performance in such scenarios. In this paper we therefore propose a method of overcoming these limitations by striping data at the transport layer. For a proof-of-concept demonstration we use a substantially enhanced version of the Stream Control Transmission Protocol (SCTP). Our analytical results and experimental data illustrate that there are significant advantages to using a transport protocol with native support for the simultaneous use of multiple network interfaces, as opposed to stretching TCP to a point where it is no longer effective.**

This work naturally leads to the more fundamental issue of end-to-end support for host mobility at the transport layer. Our analysis and results demonstrate that transport layer support for multiple network (IP) interfaces, together with the capability to dynamically add or delete IP addresses can yield the following advantages: higher bandwidth, load balancing and increased fairness, enhanced reliability, and end-to-end support for host mobility. This is independent of the underlying network layer and hence is applicable to static/wired as well as wireless/ad hoc environments. The proposed protocol offers a unified solution to both data striping across multiple networks interfaces as well as end-to-end mobility support.

***Index Terms*— Network protocols, transport layer, TCP, SCTP, data striping, inverse multiplexing, mobility, wireless networking.**

I. Introduction

With the increasing availability and variety of networking technologies, hosts will inevitably have multiple means of network connectivity. For example, mobile hosts may currently support Internet access via wired Ethernet, 802.11, Bluetooth, or cellular phone modem. It is then a dynamic optimization problem to select which combination of available network services to use at any given time. The varying network conditions associated with each active network interface must be considered in order to best meet an application's requirements. Factors such as effective bandwidth, latency, jitter, power consumption, cost, and network load might be taken into account with relative priorities determined by an application or user. In particular, wireless bandwidth is relatively scarce when compared to wired networks. It is therefore likely that mobile users will want to take advantage of as much bandwidth as is available by simultaneously striping data through multiple network interfaces.

The issue of simultaneous data striping across multiple network paths was recently investigated in [1]. It proposed a solution at the IP layer which is similar in spirit to Mobile IP [2], [3], namely IP within IP encapsulation (i.e., tunneling) is used for simultaneous data striping across multiple network paths. The motivation for a network layer solution was to isolate the transport layer (TCP) and higher layers from any modifications. In [1] the problem of striping a TCP connection across multiple network paths was analyzed in depth, and the proposed solution worked well in many cases. In particular, when the round-trip-times (RTTs) of packets sent along multiple paths is similar, the performance of the proposed solution was quite good. However, a wide disparity in the RTTs of the underlying paths can lead to severe performance degradation. That is, using multiple paths can result in much worse performance then when a single path is used alone.

The main reason for this phenomenon is that packets sent on a higher latency path will take much longer to reach the destination (compared to packets sent on lower latency paths) resulting in a RTT above the timeout threshold. This causes TCP to invoke congestion avoidance procedures which in turn leads to the under utilization of all paths being used. TCP expects RTTs to be relatively stable during steady-state operation and significant fluctuations in RTT are regarded as transient events, not as a persistent condition. TCP therefore maintains a single RTT estimate per connection, irrespective of the actual number of underlying network paths used by a connection. Hence, as long as the RTTs from different paths are within the tolerated deviation, there is no performance degradation. However, since independent RTT estimates (per network path) are needed when the RTTs are dissimilar, any data striping solution that operates below TCP may negatively impact performance in the general case.

In this paper, we therefore identify the characteristics of a transport protocol that will best support data striping across multiple network paths. The most important of these characteristics are:

(1) Dissociation of network layer (IP) addresses from their conventional role as transport layer connection identifiers, and
(2) Stream management, including striping and re-merging issues, which in turn involves load balancing and other optimizations depending on overall performance goals.

The Stream Control Transmission Protocol (SCTP) [4], [5],

Manuscript received March 15, 2003; revised September 30, 2003.

This work was supported in part by Aether Systems Inc. and NSF grants ECS-9875705 and ECS-0196362.

The authors are with the Computer Science and Electrical Engineering Department,University of Maryland, Baltimore County, Baltimore, MD 21250. E-mail: {tgoff1, phatak}@umbc.edu.

[6] has many of the desired characteristics. For a proof-of-concept demonstration of principles, we therefore use a modified version of the SCTP reference implementation [5]. Note that the use of SCTP is incidental, it is a vehicle to illustrate the ideas being proposed. This protocol is used to conduct experiments and the resulting data demonstrates significant advantages (bandwidth aggregation, increased reliability, etc.) from using such a transport protocol, as opposed to stretching TCP to a point where it is no longer effective.

This work naturally leads to the more fundamental issue of end-to-end support for mobility at the transport layer. Transport layer support for multiple network interfaces, together with the capability to dynamically add or delete IP addresses can yield the following advantages:

(1) Higher bandwidth/throughput,
(2) Load balancing leading to increased fairness,
(3) Enhanced reliability, and
(4) True end-to-end transport layer support for host mobility.

This is independent of the underlying network layer and hence is applicable to static/wired as well as wireless/ad hoc environments.

The remainder of this paper is organized as follows. The next section defines the problem being considered and gives a brief survey of prior related work including the strengths and drawbacks of the proposed approaches. This naturally leads to as yet unaddressed problems which are identified. Characteristics of a transport layer protocol which will enable a clean solution to these problems are then discussed in the following section. The subsequent section presents our preliminary implementation of such a protocol, the experimental setup, and results. The last section presents a unified view, namely that the proposed mechanism not only solves the multiple interface problem, but it also has the advantage of supporting end-to-end transport-layer mobility which in turn has substantial benefits in its own right.

II. Background and Related Work

The simultaneous use of multiple network interfaces to achieve data striping has been investigated in several domains. Most recently the performance of application layer data striping was considered in [7], transport layer striping was studied in [8], [9], while network layer striping was analyzed in [1]. In its most general form, the common question is one of how to make efficient use of a possibly dynamically changing set of network interfaces. We therefore present a brief summary of the problem definition and include an overview and comparison of related work.

A. Problem Definition

Fig. 1 illustrates the case when two multihomed hosts, A and B, wish to communicate via the Internet. For instance, host A might be a laptop with network access via an 802.11 LAN (say interface A_1), a Bluetooth scatternet (A_2), and a cellular phone modem (A_3). In general, the IP addresses assigned to interfaces A_1, A_2, and A_3 will be controlled by separate Internet Service Providers (ISPs) who might in turn implement firewalling to various degrees.

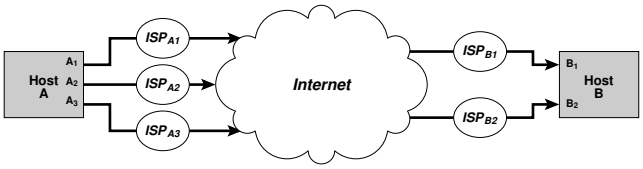


Fig. 1. Communicating Multihomed Hosts

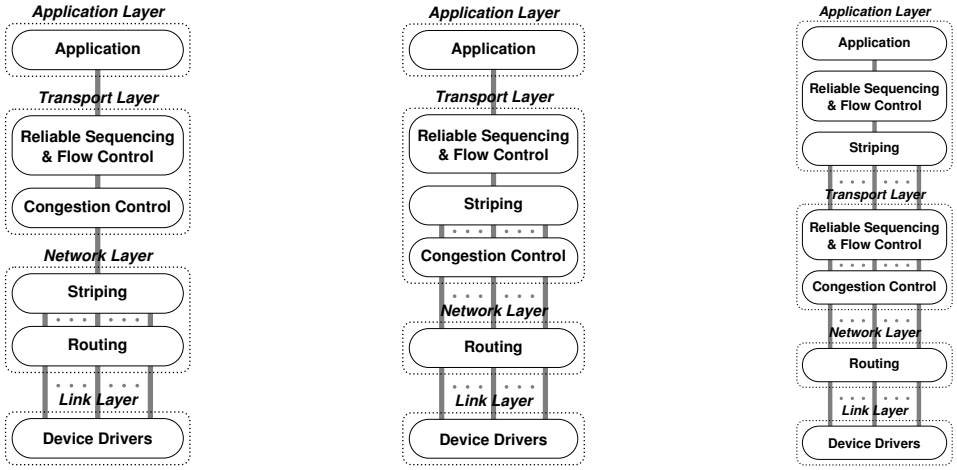
To take advantage of as much bandwidth as is available, suppose A wants to simultaneously stripe data sent to B over multiple network interfaces. This striping could be achieved in a variety of ways including striping at the link layer, network level, transport level, application level, or even striping at multiple layers. Data striping at each of the layers is considered in the following sections, with a focus on bandwidth aggregation as well as enabling support for host mobility.

B. Approaches to Data Striping

Since fundamentally data striping can be implemented at any layer of the network protocol stack, the applicability and tradeoffs of several approaches are discussed in the following sections. Fig. 2 illustrates, from a single application's point of view, some of the relevant features of each layer and how striping might be incorporated into the protocol stack. Here the assumption is that the application requires reliable in-order delivery. Each vertical line represents a distinguishable stream of packets which a layer may have to differentiate between, all of which originated from or are destined to the application. For example, multiple streams through the congestion control layer in Fig. 2(b) and Fig. 2(c) indicate that each stream is processed independently where congestion in one stream has no impact on other streams. Link layer striping is not included in Fig. 2 because our focus is on inter-network striping applications. Note that the functionality included in Fig. 2 is not meant to be comprehensive, it is only meant to draw attention to the components which might be affected by the introduction of striping. It should also be noted that complexity and memory requirements of each layer will likely be a function of the number of streams passing through it.

1) Link Layer Data Striping: Although most link layer striping schemes can only be used within a local network, a discussion of several approaches is included for completeness. Bandwidth aggregation across multiple channels has been considered in the literature in varying contexts and for scenarios and applications distinct from those considered in this paper. For instance, multi-link PPP [10], [11] is designed to aggregate multiple *logical* data channels into one. Since PPP was intended to operate at the data link layer, below the IP level [12], [13], [14], the multi-link PPP extension bundles multiple data-link level channels into a single logical link.

So-called layer two tunneling using L2TP [15] does allow PPP to operate over packet switched networks. IP and TCP can then operate over the bundled logical PPP channel, which accomplishes the same result as striping at the network layer. In this case the transport level uses one logical channel where packets are sent on several distinct network paths. This results



(a) Network Layer Striping

(b) Transport Layer Striping

(c) Application Layer Striping

Fig. 2. Data Striping Alternatives

in the problems discussed in detail below, primarily because congestion control is not performed independently for each network path.

Bandwidth aggregation has also been considered in the context of storage systems and high volume data or file servers, for instance [16], [17], [18], [19]. Typical storage-server architectures are confined to be within a LAN having a single controlling authority. In this case bandwidth can be aggregated at the MAC layer. For instance, Cisco's EtherChannel [20] product provides bandwidth aggregation across multiple Ethernet links. Likewise, a recent IETF draft from the HP storage systems group [19] deals with exploiting ultra wide-band SCSI connections for distributed storage. The use of these solutions is usually restricted to local networks and cannot be used for general communication through the Internet.

2) Network Layer Data Striping: The existing body of work on the Mobile IP standard suggests that any proposed solution should target the network layer (IP) since restricting modifications to this layer can maximize compatibility with the existing infrastructure. Therefore all existing applications can transparently benefit from any enhancements. The network layer is also, in essence, the first device-independent layer making data striping at this layer applicable to a wide variety of networking technologies.

With this in mind, network layer striping was analyzed in detail in [1]. As shown in Fig. 2(a), network layer striping was implemented prior to routing individual packets. Although striping could be achieved through routing alone, additional packet processing is usually necessary to ensure a packet is routed as desired and accepted properly by the destination.

For network level striping to work in the general case, tunneling or encapsulation of IP packets is required. For the purpose of illustration, consider Fig. 1 and assume that host A is the source and host B is the destination. Any IP packets sent from A to B must contain the source address of whichever network interface was used to send the initial SYN packet to establish the TCP connection. If this were not the case, B would not recognize incoming packets as belonging to an active connection, and the packets would be dropped. On the other hand, packets sent from A whose source address differs from that of the outgoing network interface appear as spoofed to intermediate routers. For legitimate security reasons, spoofed packets may be dropped by ISPs who employ ingress filtering. Thus, packets sent from A through alternate network interfaces must be tunneled, that is encapsulated in a packet that has a proper source address.

Further performance limitations of any data striping approach at the network level are described in Section II-C.

3) Transport Layer Data Striping: Transport layer solutions to data striping with support for host mobility have been recently proposed in both [8] and [9]. These works are similar in spirit to the approach this paper presents, but there are several substantial differences as indicated below.

(1) The most important distinction is that we propose a unified protocol to efficiently stripe data across multiple network paths as well as provide end-to-end support for host mobility. In sharp contrast, the mechanisms presented in [8] and [9] are solutions to the problem of data striping alone, albeit in mobile scenarios. In other words, neither pTCP presented in [9] nor RMTP demonstrated in [8] is proposed as a transport layer solution to the problem of host mobility itself. Our framework, on the other hand, presents a true end-to-end transport level solution to handle host mobility.

(2) Moreover, a comparison to network level striping is not considered in either [8] or [9]. We present analytical and simulation results comparing network and transport

level striping.

(3) In [9] a somewhat complex mix of pTCP and TCP-v is proposed. The TCP-v is essentially standard TCP running on each individual network path to be aggregated, while pTCP is a wrapper responsible for the overall operation of the aggregate of the underlying paths. The use of TCP-v on each individual network path has its own advantages and disadvantages. For instance, a TCP-like connection on each individual path implies that the ACKs will come back on the same path. Our framework is more flexible, it would allow ACKs to be sent on a different path.

(4) Another important distinction is that handoffs are not explicitly considered in [9] (this is mentioned by the authors in the discussion section of [9]). In this paper we consider handoffs and have presented simulation results for such scenarios (i.e., the dynamic addition and deletion of IP interfaces, see Section IV and Section V).

(5) The Reliable Multiplexing Transport Protocol (RMTP) [8] also targets bandwidth aggregation on multihomed mobile hosts. However, it is a fundamentally different rate-based approach. It performs explicit bandwidth based striping wherein the available bandwidth is actively probed via packet-pair probes. As mentioned in [9], if bandwidth fluctuations occur on a time scale smaller than the bandwidth probing intervals, then RMTP would have difficulty in effectively adapting to the fluctuations. Our scheme does not perform active bandwidth probing.

SCTP (an existing transport level protocol) supports multi-streaming and multihoming, that is a host having multiple IP addresses. SCTP implements congestion control on a per network path basis (although currently SCTP defines a network path by its destination address alone). Also, a recent extension [21] proposes a mechanism for dynamically adding or deleting IP addresses at either the source and destination. However, in its current form SCTP does not do load-sharing, that is multihoming is used only for redundancy. A single address is chosen as the primary address for a connection, meaning the destination to which all normally transmitted data chunks are sent. Retransmitted data chunks may use an alternate address to improve the probability of reaching the remote endpoint. Persistent send failures from a primary address will ultimately result in choosing a new primary address for the connection. We have substantially modified SCTP to overcome all these limitations, our implementation is discussed in Section IV.

Note that for any transport level striping mechanism to be effective, an essential characteristic is that congestion control must be performed independently for each network path. This is pictorially illustrated in Fig. 2(b).

4) Application Layer Data Striping: Fig. 2(c) depicts an approach to application layer data striping. The obvious duplication of functionality is necessary because flow control and reliable in-order sequencing are typically performed in conjunction with congestion control by the transport layer. Ideally an application would be able to disable transport layer reliable sequencing in this situation, since requiring in-order delivery introduces the possibility of head-of-line blocking. Implementing this as close to the application as possible allows packets to be processed lower in the protocol stack with the maximum degree of parallelism.

Striping data at the application layer places the burden of opening and managing multiple transport layer connections on the application or library developer. In this case the application would open multiple connections from different network interfaces and be responsible for striping the data stream at the sender and merging it properly at the receiver. This solution was used in [22] and [23], where the use of multiple transport level connections was motivated by the limitation of TCP window sizes when communicating over "long-fat pipes".

Aside from possibly being tedious, this approach could be used effectively in scenarios where paths are not too lossy and disconnections due to mobility are infrequent. The latter two cases lead to the following problems when opening multiple TCP connections.

(1) An application level approach would have to implement its own buffering and acknowledgment scheme to support the scenario where connections may be permanently lost. This is necessary since any data buffered to be sent on a TCP connection that was lost due to mobility, for example, would never be received at the destination unless it is resent at the application level. This is not addressed in either [22] or [23].

(2) Striping at the application level can also be less efficient when packet retransmissions are considered. Once the application level entity chooses a transport layer connection to send data on, that block of data will be retransmitted as many times as needed over the same network path. If data striping were instead implemented by the transport layer, packets needing retransmission could be sent on an alternate network path which might be more reliable or less congested.

(3) An application or library would have to perform a substantial amount of cross-layer data gathering. For instance if a host moves and acquires a new IP address, the library must learn the new address. Likewise, it should implement its own heartbeat-type scheme to determine whether inactive paths are alive or dead.

Furthermore, as demonstrated in [9], if the application level buffer size at the receiver is not sufficiently large, the overall throughput of multiple TCP connections could be slower than that of the slowest network path. Even if the buffering requirements can be met, stalls due to losses in the slower path can lead to a poor overall throughput as illustrated in [9].

C. A Comparison of Network and Transport Layer Striping

From the issues described in the previous sections, striping data at the transport layer seems appropriate. This, however, comes at the cost of sacrificing interoperability with the large existing infrastructure which uses TCP. The limitations of network layer striping detailed in [1] fall into two main categories: first, disparate RTTs can cause spontaneous timeouts

and secondly, persistent out-of-order packets can lead to unnecessary fast retransmissions. In both cases TCP's congestion control and avoidance mechanisms are invoked needlessly which degrades performance.

This drag-down effect, due to TCP timeouts and fast retransmissions, is illustrated in Fig. 3. In this scenario two network paths having a bandwidth ratio of 4:1 exist between a pair of communicating hosts and the striping solution from [1] was used. Realized Application bandwidth is plotted as a function of raw bandwidth, where raw bandwidth is the combined raw bandwidth available from both paths between the hosts. The experimental setup used to produce these results is described more thoroughly in Section IV, but in this case data splitting was done at the network layer. The application bandwidth provided by TCP was measured using the netperf benchmarking package [24].

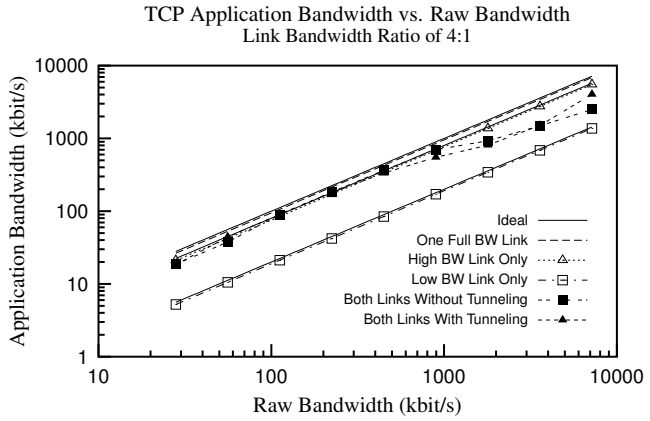


Fig. 3. Application Throughput with Network Layer Data Striping

The 8 curves in Fig. 3 can be interpreted as follows. The 3 solid lines labeled "Ideal" are simply the $y = x/5$, $y = 4x/5$, and $y = x$ straight lines corresponding to the ideal bandwidths of each individual path as well as the combined bandwidth. However, this ideal is never achievable because of various protocol headers, processing overhead, Ethernet collisions, etc. The second curve labeled "One Full BW Link" shows the TCP application bandwidth when there was a single path with the same raw bandwidth as the abscissa value (which is the sum of bandwidths of both paths). This is slightly less than the corresponding ideal case, as expected, due mainly to headers and processing overhead since the physical network was a private subnet where collisions and congestion were not a factor. The curve labeled "High BW Link Only" corresponds to using only one of the two paths, the one with higher bandwidth. The plot labeled "Low BW Link Only" is analogous.

The most relevant plot is the one labeled "Both Links With Tunneling", this curve is below the plot labeled "High BW Link Only" thereby illustrating that the lower bandwidth path has degraded the overall performance to a level below that achievable by using the higher bandwidth path alone. This clearly illustrates the limits of striping data at the network layer.

Striping at the transport layer can completely avoid these problems by applying congestion control independently for each network path. In addition, transport layer data striping is more robust than network layer striping when packet loss is considered. Results in the literature [7], [25], [26], [27] indicate that for moderate packet loss rates (less than 1/100) the bandwidth TCP will attain is bounded by

$$\text{BW} \leq \frac{\text{MSS}}{\text{RTT}} \frac{C}{\sqrt{L}}, \tag{1}$$

where BW is the realized bandwidth, MSS is the Maximum Segment Size, C is a constant, and L is the overall loss probability of the path. To isolate the impact that packet loss has on TCP throughput, this can be approximated as

$$\text{BW} \propto \frac{1}{\sqrt{L}}. \tag{2}$$

A performance comparison between network and transport layer striping can then be made with respect to packet loss by considering the loss factor, L, that the congestion control algorithm(s) would experience in each case.

Assume there are n network paths over which data striping is performed, and let the loss probabilities for each path be denoted $l_1, l_2, \cdots, l_n$. In a network layer striping approach, when data striping is performed below congestion control in the protocol stack, the overall loss probability becomes

$$L_{\text{ns}} = \sum_{i=1}^{n} l_i, \tag{3}$$

assuming the individual path loss probabilities are independent. This, in turn, makes the network layer striping bandwidth, BW_{ns},

$$\text{BW}_{\text{ns}} \propto \frac{1}{\sqrt{\sum_{i=1}^{n} l_i}}. \tag{4}$$

For transport layer striping, the congestion control layer experiences each path loss directly and independently. Since each path is handled by a separate TCP-like control mechanism, the bandwidth realized by each network path is $\text{BW}_{\text{ts}_i} \propto 1/\sqrt{l_i}$. Consequently, the total transport layer striping bandwidth, BW_{ts}, is then

$$\text{BW}_{\text{ts}} \propto \sum_{i=1}^{n} \frac{1}{\sqrt{l_i}}. \tag{5}$$

From (4) and (5) it is easy to verify that transport layer striping is, in this regard, inherently better since a higher bandwidth is attained because

$$\sum_{i=1}^{n} \frac{1}{\sqrt{l_i}} > \frac{1}{\sqrt{\sum_{i=1}^{n} l_i}} \quad \text{or} \quad \text{BW}_{\text{ts}} > \text{BW}_{\text{ns}}. \tag{6}$$

III. General Protocol Characteristics Required for the Efficient Use of Multiple Interfaces

Fundamentally, TCP was not designed to simultaneously operate across multiple paths. Hence any solution that is completely transparent to TCP is bound to run into problems similar to those discussed above. For example, some of a host's multiple interfaces may go up or down dynamically. This may happen, for instance, when a user connected to the Internet via cellular phone modem drives into the vicinity of an Airport and can now access the Internet via the Airport LAN as well.

Likewise, someone using a wireless campus LAN might drive out of range and would then like to switch to the cellular phone modem alone. It is evident that binding a transport layer connection to a single source/destination address pair breaks down when a host moves in this fashion and acquires a new IP address.

This immediately suggests that a solution to the multiple interface problem at the IP layer cannot be applied to mobile environments. This could be overcome by using Mobile IP simultaneously for each network interface of a mobile host, but Mobile IP was not designed with such a complicated scenario in mind and is likely to be inefficient. Since the problem at hand basically deals with reliable in-order delivery, unified support for mobility is better handled at the transport layer. To simultaneously support multiple network paths, the most logical approach is to have the transport level protocol allow multiple network layer addresses for the source as well as the destination. Furthermore, to support mobility it must allow dynamic addition and deletion of IP addresses for both the source and destination. In essence, the transport layer connection identifiers must be distinct or dissociated from the network addresses.

The following is a list of desirable characteristics of a transport protocol that enables the efficient use of multiple IP interfaces.

(1) Dissociation of network (IP) addresses from their conventional role as transport layer connection identifiers. This includes at least the following two aspects:
   (i) Transport layer connection identifiers should be able to support multiple network layer addresses (this implies transport layer support for multihomed hosts), and
   (ii) The dynamic addition and deletion of network layer (IP) addresses to/from the transport later connection identifier must be allowed (this inherently provides support for host mobility).

(2) The ability to stripe a single data stream into multiple flows. This could involve load balancing and many other optimizations to achieve desired goals, such as optimizing delay, fairness, or power consumption.

(3) The resulting multiple data flows must then be sent along distinct physical network paths. Congestion control must be performed independently for each path where, in general, a path is defined by a source-destination address pair.

(4) Finally, packets from the multiple data flows need to be merged and properly sequenced by the receiving end.

When a connection is initiated, there needs to be some way of finding what current IP address(es) a possibly mobile host has. This must be performed through a process similar to DNS name resolution, which is necessarily outside the scope of a transport protocol itself. In other words, in addition to the characteristics listed above, a robust name resolution scheme must be available to initiate or bootstrap a connection. We would like to point out that such a name resolution process is universally required in any protocol (standard IP, Mobile IP, etc.).

Assuming mobile hosts, a DNS update mechanism like those proposed in [28], [29] is one way to provide the name resolution required to start a connection. In a similar manner, a so-called rendezvous server could be used. A rendezvous server is a static host known to both communicating mobile hosts that is notified of address updates by at least one mobile host. The mobile hosts can then rendezvous through the static host in the event of simultaneous moves. On the other hand, in a completely ad hoc scenario, a brute-force solution to name resolution is flooding. After a connection has been established, a move by either peer alone does not necessitate a name resolution. The host which moved still has a valid address for its stationary peer and can directly inform the static host of its new address. However, when both peers move simultaneously, that is before either can notify the other, the name resolution process must be repeated. An example of such a scheme can be found in [30].

While all items from the list above deserve thorough consideration, we focus here on illustrating the substantial advantages the first characteristic alone can yield. The emphasis is to demonstrate the ideas via a proof-of-concept implementation and experiments. Hence, for items (2), (3), and (4) we have used straightforward schemes. For example, we stripe packets among the streams (and hence network paths) according to the static ratio of path bandwidths. In fact, properties (2), (3), and (4) are not even necessary to use multiple IP interfaces under the SCTP protocol. As explained in the following section, SCTP can use multiple network interfaces to send a single SCTP data stream thereby obviating the need for any striping and merging of data by applications. It should therefore be stressed that our use of multiple SCTP streams below was for experimental convenience. SCTP does not impose any restriction on what network path the packets from a single stream take; packets from a SCTP stream may be sent on any network path from an association. SCTP congestion control is completely independent from stream designations and is applied to all packets that share a common network path (currently defined by destination IP address) regardless of their stream number.

IV. Implementation and Results

Having identified the desirable transport layer protocol characteristics above, SCTP is a natural starting point. SCTP supports multihomed hosts, and the dynamic addition and deletion of IP addresses. In its current form, however, SCTP lacks the multi-streaming capability. By default, multiple network interfaces are used only to increase reliability. When packets sent through the primary interface are lost, a different interface might then be used. We therefore use a modified version of the SCTP reference implementation [5] as a vehicle to demonstrate our ideas.

SCTP is a connection oriented transport layer protocol that provides the reliable in-order delivery of data packets. Within a single SCTP connection, known as an “association”, multiple independent communication streams can exist. Packets from the same stream are delivered to an application in-order, while the order of packets from different streams is arbitrary. This

avoids the head-of-line blocking problem which can affect TCP, where a single lost packet delays the reception of all packets that come after it. SCTP is also resilient to network disruptions because it supports multihomed hosts and performs congestion control separately for each destination address. This is further enhanced by an extension (IETF draft [21]) that allows dynamic addition and deletion of IP addresses at either end of an active association.

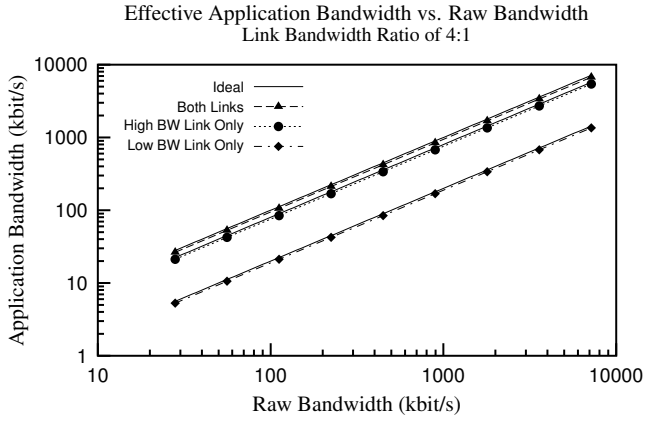
For the purpose of our experiments, the reference implementation of SCTP was enhanced to allow an application to associate a given communication stream with a particular network path by setting connection options, similar to how the Berkeley socket API provides `setsockopt()` to control TCP options. The network interface used to reach a destination is subject to the normal IP routing procedure. In other words, if a source node has multiple network interfaces that can be used to reach a single destination address, the interface that is actually used is determined by the IP layer, not by SCTP. Once a stream is bound to a network path, the corresponding network interface gets used for sending the stream's outgoing packets. If a failure is detected, by a lost packet for example, an alternate destination address (network path) may be used. Therefore to accomplish data striping, an application using our prototype implementation would initiate an association (connection) with a foreign host, open multiple communication streams, and bind each stream to a network path. The application was made explicitly responsible for controlling the number of streams used for communication only to allow greater experimental flexibility. In the future this could be made more transparent to applications by having SCTP manage streams according to more abstract communication categories, for example bulk data transfer or interactive session.

We would like to emphasize that multiple SCTP streams were used merely for the sake of convenience and flexibility in the initial experiments. The SCTP protocol itself does not in any way require that all packets from a particular stream use a common network path. In fact, this restriction is explicitly avoided by keeping reliable delivery functionally separate from sequenced stream delivery [4, Section 1.3.4]. SCTP in effect assigns two sequence numbers per packet; one used to implement congestion control and ensure reliable delivery, and the other used for sequencing packets within a stream.

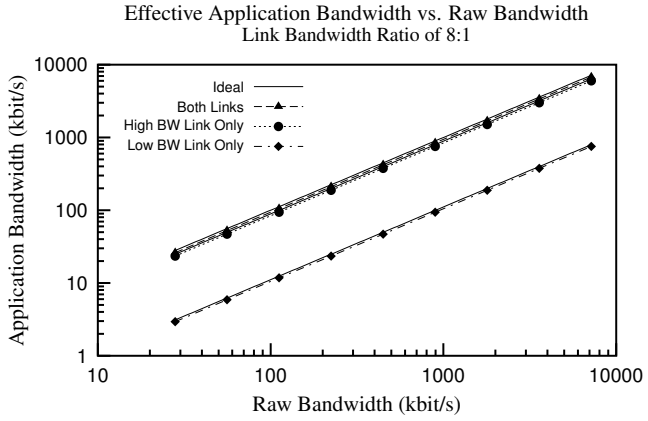
An application could therefore use a single SCTP stream for communication and realize the benefits illustrated below while maintaining in-order delivery of all packets, assuming the SCTP implementation striped packets from a single stream over all available network paths. In other words, it is not necessary for applications to stripe data into multiple SCTP streams, actively managing the different streams, and merge them at the receiving end. A single stream can use all available network interfaces, thereby completely avoiding stream management issues. In this case SCTP's native in-order delivery of packets within a stream ensures that packets sent across multiple interfaces are delivered in-order to the receiver.

All experiments were performed on private subnets using a pair of identical hosts running FreeBSD 4.5. There were two independent 100 Mbps full-duplex Ethernet connections between them. The bandwidth through each network interface of the client machine was controlled using the dummynet feature native to FreeBSD [31]. Among other things, this allows a system administrator to dynamically limit bandwidth according to firewalling rules. Note that the task of discovering multiple IP addresses for the source and the destination is subsumed by the SCTP protocol itself.

A client-server program was written to measure the effective application throughput under varying network conditions. The client would request some amount of data from the server, which was then delivered in 1024 byte blocks. The client would log the reception of every n-th packet, where n was chosen according to the total amount of requested data. The results are presented in Fig. 4(a) and Fig. 4(b). These results are the average of five runs, where each run lasted approximately ten minutes.



(a) Two Paths with Bandwidth Ratio 4:1



(b) Two Paths with Bandwidth Ratio 8:1

Fig. 4. Application Throughput with Transport Layer Data Striping

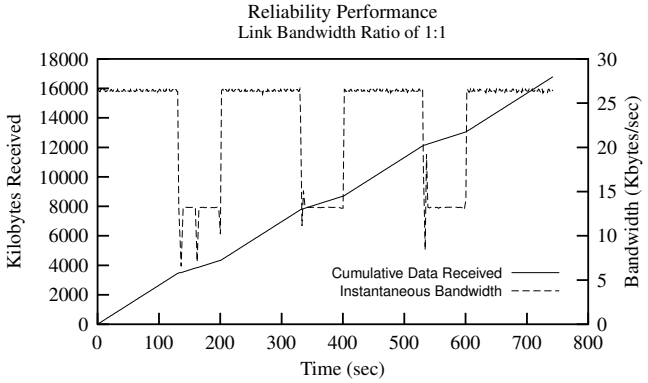
We first illustrate that the proposed transport level protocol avoids the drag-down problem which TCP suffered from. This can clearly be seen by contrasting Fig. 4(a) with Fig. 3. Here the two paths between the hosts were configured to have a bandwidth ratio of 4:1 over the range of bandwidths shown along the x axis. The second bandwidth ratio we considered

was 8:1, shown in Fig. 4(b). This was deliberately chosen above the bandwidth ratio threshold of 7:1 which was the analytically derived upper bound from [1], beyond which TCP retransmission timeouts would severely degrade performance. This demonstrates that the new protocol is completely free from the drag-down problem, irrespective of the relative bandwidths of the paths involved.

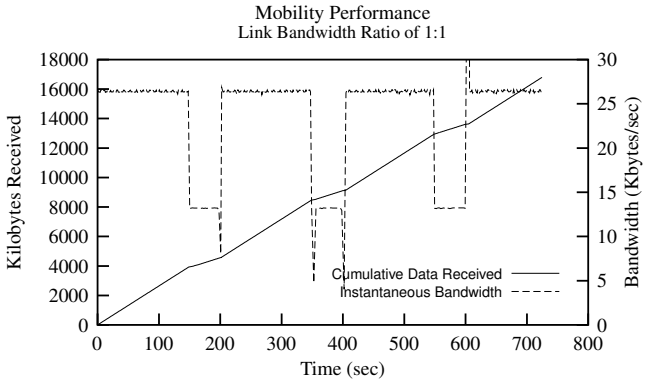
Both figures plot application throughput as a function of the available raw bandwidth in a manner analogous to Fig. 3. In both plots, the curves labeled "Both Links" are appropriately above the curves labeled "High BW Link Only", thereby illustrating that the new protocol gainfully exploits the available bandwidth of both paths simultaneously.

The next performance factor considered was reliability. Fig. 5(a) shows how the new protocol behaves as one network interface intermittently fails. This was emulated by periodically bringing one of the interfaces up and down. In this case, both paths were configured to have the same raw bandwidth of 112 Kbps. Three failure cycles are shown, where one network interface remained down for the last third of each cycle. The cumulative amount of data received, as well as the instantaneous bandwidth, is plotted as a function of time. Initially both paths are operational. The first failure occurs after roughly 130 seconds. During the failure the application continues to receive data, but at half the rate since only one of the two paths is available. After the recovery, at around 200 seconds, the data rate returns to its prior level indicating that both paths are again being fully used.

In a similar experiment, the protocol's reaction to mobility was tested. This scenario is identical to the reliability experiment, with the exception of how the network interface was brought up after each failure. In this case, the network interface was given a different IP address each time it was brought up. Fig. 5(b) illustrates that the change in IP address has a negligible effect on performance thereby demonstrating that the proposed protocol effectively handles host mobility as well.



(a) Reliability performance of two identical 112 Kbps paths.



(b) Mobility performance of two identical 112 Kbps paths.

Fig. 5. Reliability and Mobility Performance

V. Unified Support for Multiple Network Interfaces and Host Mobility

Having illustrated the effectiveness of the proposed protocol through experimental results, we now discuss three aspects of supporting mobility in more detail: namely the separation of network addresses from transport layer connection identifiers, security, and application to mobile ad hoc scenarios.

A. Separating Network Addresses and Transport Layer Identifiers

Note that a transport level protocol which dissociates IP addresses from transport layer identifiers can also provide effective end-to-end support for mobility, as illustrated by Fig. 5(b). We would like to emphasize that transport layer support for host mobility and the related mechanisms, such as multihoming support which have been expressly proposed for the purpose of mobility support alone is not our contribution in this paper. This issue has been widely investigated in the literature where a sampling can be found in [29], [30], [32], [33], [34], [35], [36]. Rather, our contribution here is a unified framework; we arrive at a clean, true end-to-end transport level solution to mobility as a byproduct of our proposed solution to the multiple-interface problem.

The Host Identity Payload (HIP) mechanism proposed in [37] also advocates decoupling network addresses used for routing from transport layer identifiers used to establish and maintain communication. It proposes replacing all references to IP addresses in application and transport layer entities with identifiers from a new namespace, the Host Identity (HI). The HI would "name" a specific host, where hosts may have multiple names. The HI as proposed is cryptographically based, with a public key identifying a host. Public host identities, such as for well known servers, could be stored in DNS records or LDAP directories, while the identities of anonymous clients can be exchanged between communicating peers as needed.

A new protocol layer, the HIP protocol, is needed between the network and transport layer to manage translations between the HI and network layer identifiers, as well as to authenticate hosts. This guards against man-in-the-middle connection hijacking, although additional security measures are needed to detect if a man-in-the-middle has modified any packets it forwards. HIP natively supports mobility since network addresses can be changed by the HIP layer in a manner that

is transparent to the transport and application layers. Again, an external dynamic DNS or rendezvous server is needed to initiate a connection, or to reestablish a connection when both hosts move.

A main objective of SCTP was to work within the existing host naming framework provided by DNS and IP addresses. Hence, it is less elaborate than more recent proposals such as HIP when it comes to cryptographic authentication and the like. This maintains interoperability and avoids the translation layer used by HIP. If stringent security is required, SCTP could be be used in conjunction with other security mechanisms (including HIP) as explained in the following section.

Once transport layer connection identifiers are distinct from network addresses, host mobility support is natural. For instance, if a node moves and acquires a new network address after establishing a transport level connection, it simply communicates the new address and both peers add this new address and delete the stale one from their transport level connection identifiers. This is demonstrated in Fig. 5(b), where the network interface that was brought down and back up was assigned a different IP address in each cycle.

Note that the mobility support scheme proposed here is somewhat different from the one proposed in [29] which requires an explicit connection migration mechanism. Here connection migration is handled transparently through the process of first adding a new IP address and then deleting an old IP address. This approach is also taken in [38] and [39], where SCTP is used in conjunction with Mobile IP to support mobile hosts. Specifically, Mobile IP is used for location management (i.e., to initially find a mobile host) and SCTP is used to handle handoffs (i.e., to deal with movement while a connection is active and to eliminate triangle routing). Provided at least one network path exists between communicating peers, an active data transfer is not interrupted. When a host moves suddenly, any packets its peer sent to the previous IP address would be lost, causing the peer to assume congestion and initiate the typical exponential back-off. In addition, proper slow-start congestion control is automatically performed for each new path between the peers. Note that the proposed scheme still requires a name resolution step to start or bootstrap a new connection, as well as when both peers of a connection move simultaneously.

The proposed scheme completely eliminates triangle routing, and reverse-tunneling used in the standard Mobile IP approach. Routing can therefore be simplified for regular mobile (as opposed to ad hoc mobile) scenarios. This can, in turn, potentially yield many other benefits such as load balancing, fairness, higher throughput and efficiency, etc. Furthermore, such a scheme does not require any intermediaries. Unlike the Foreign Agent, Home Agent, etc. that are required in the Mobile IP standard, a mobile node simply acquires a new address and tells its peer to add it to the new connection identifier and delete the old one. This constitutes a true end-to-end solution to mobility.

We would like to point out that SCTP was originally introduced to address a completely different set of issues. The primary motivating factor for the development of SCTP was transporting signaling traffic over IP networks, for telecommunication applications in particular. The features that efficiently support such signaling protocols are also beneficial to a broader set of applications that require partially ordered data and wish to avoid the unnecessary delay from the head-of-line blocking exhibited by TCP.

The experimental results in this paper indicate that, with some extensions, SCTP constitutes a clean unified solution to two more problems of particular importance in wireless environments: viz., (i) the efficient use of multiple IP interfaces, and (ii) host mobility support. It appears that radically different sets of design and performance goals have lead to a unified solution. We therefore conjecture that the proposed protocol is likely to represent a "good quality" solution in the otherwise vast space of all possible solutions.

B. Security

Using the SCTP extension for dynamic IP address addition and deletion poses little addition risk to security. The existence of this additional feature may provide a man-in-the-middle, who can intercept and alter packets, an alternate approach to connection hijacking in that a connection could be forwarded to a third party. However, this threat is small and is not unique to the dynamic IP address addition/deletion mechanism. Current SCTP and TCP implementations are also equally susceptible to connection hijacking and precautions are typically taken at the application level to detect such attempts. In general, for scenarios where security is critical, additional authentication methods may be used, for example [21] proposes using the IP authentication header [40].

The additional security provided by HIP can resist connection hijacking due to mobility spoofing, but the general integrity of individual packets is not guaranteed by HIP. The dynamic IP address addition/deletion feature of SCTP balances the tradeoff between security and efficiency by supporting the use of an additional comprehensive security plan at the network or application layer(s) when needed. For example, if security at the network layer verifies individual packets, dynamic addition or deletion of IP addresses at the transport layer is inherently secure.

C. Applications to Mobile Ad Hoc Scenarios

Routing is fundamentally related to location, that is bits must be delivered to a particular destination. The process of determining the location of the destination within the network is therefore implied. For this reason, routing is typically simplified when addresses correspond to some sort of actual (physical/geographical) or logical network structure. Examples for ad hoc scenarios include routing that exploits geographical location [41] and hierarchical routing [42], [43]. Other examples include the many investigations that have considered how to dynamically create and manage clusters, and elect cluster heads to impose some hierarchy that simplifies routing in ad hoc networks.

Once a transport level connection is dissociated from any specific network layer address, as a node moves in an ad hoc network it can assume any network address which best exploits the underlying network structure. In this way routing issues are

separated from transport layer issues and the proposed scheme can also benefit mobile ad hoc scenarios.

VI. Conclusion

We have identified the attributes a transport layer protocol should have in order to efficiently support simultaneous data striping across multiple network interfaces. The most important characteristic necessary was shown to be the dissociation of network layer (IP) addresses from their traditional role as transport level connection identifiers. This subsumes the ability to support multihomed hosts as well as the capability to dynamically add or delete IP addresses for either a source or destination. We implemented such a protocol and the experimental data demonstrates the significant advantages of using such a transport protocol as opposed to stretching TCP to a point where it is no longer effective.

Furthermore, this work naturally leads to the more fundamental issue of end-to-end support for host mobility at the transport layer. It shows that transport layer support for multiple interfaces with the capability of dynamic address addition and deletion can yield the following advantages: higher overall bandwidth, load balancing leading to increased fairness, enhanced reliability, and end-to-end support for host mobility. This is independent of the underlying network layer and hence is applicable to both static/wired and wireless/ad hoc environments. We arrived at the mechanism for mobility support as a byproduct of our solution to the multiple interface problem, thereby demonstrating a unified framework that can effectively solve both of these problems simultaneously.

References

[1] D. S. Phatak and T. Goff, "A Novel Mechanism for Data Streaming Across Multiple IP Links for Improving Throughput and Reliability in Mobile Environments," in *IEEE INFOCOM – Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, New York, NY, June 2002, pp. 773–781.

[2] E. C. Perkins, *IP Mobility Support*, October 1996, rFC 2002.

[3] C. Perkins, *IP Encapsulation within IP*, October 1996, rFC 2003.

[4] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson, *Stream Control Transmission Protocol*, October 2000, rFC 2960.

[5] "SCTP Reference Implementation," http://www.sctp.org/.

[6] "GPL SCTP Prototype Implementation," www.sctp.de/.

[7] T. J. Hacker, B. D. Athey, and B. Noble, "The end-to-end performance effects of parallel tcp sockets on a lossy wide-area network," in *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Fort Lauderdale, FL, April 2002, pp. 46–55.

[8] L. Magalhaes and R. Kravets, "Transport level mechanisms for bandwidth aggregation on mobile hosts," in *Ninth International Conference On Network Protocols (IEEE ICNP 2001)*, Riverside, CA, November 2001, pp. 165–171.

[9] H. Hsieh and R. Sivakumar, "A transport layer approach for achieving aggregate bandwidths on multihomed mobile hosts," in *MobiCom: The ACM Annual International Conference on Mobile Computing and Networking*, Atlanta, GA, September 2002, pp. 35–46.

[10] G. Malkin, *Nortel Networks Multi-link Multi-node PPP Bundle Discovery Protocol*, September 1999, rFC 2701.

[11] K. Sklower, B. Lloyd, G. McGregor, D. Carr, and T. Coradetti, *The PPP Multilink Protocol (MP)*, August 1996, rFC 1990.

[12] E. W. Simpson, *The Point-to-Point Protocol (PPP)*, July 1994, rFC 1661.

[13] ——, *PPP in HDLC-like Framing*, July 1994, rFC 1662.

[14] D. Rand, *PPP Reliable Transmission*, July 1994, rFC 1663.

[15] W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn, and B. Palter, *Layer Two Tunneling Protocol "L2TP"*, August 1999, rFC 2661.

[16] D. C. Anderson, J. S. Chase, and A. M. Vahdat, "Interposed request routing for scalable network storage," in *Proceedings of the Fourth Symposium on Operating System Design and Implementation (OSDI)*, October 2000.

[17] Sun Microsystems, "Sun StorEdge Network Data Replicator White Paper," http://www.sun.com/storage/white-papers/sndr.html.

[18] A. Watson, Network Appliance, Inc., "Filer Deployment Strategies for Evolving LAN Topologies," http://www.netapp.com/tech_library/3009.html.

[19] R. Haagens, "iSCSI requirements," http://www.ietf.org/proceedings/00jul/SLIDES/ips-iscsi-reqs.pdf.

[20] Cisco Systems, "Etherchannel technologies," http://www.cisco.com/warp/public/779/largeent/learn/technologies/fast_echannel.html.

[21] R. R. Stewart, M. A. Ramalho et. al., "SCTP Extensions for Dynamic Reconfiguration of IP Addresses and Enforcement of Flow and Message Limits," June 2001, http://www.ietf.org/internet-drafts/draft-ietf-tsvwg-addip-sctp-02.txt.

[22] H. Sivakumar, S. Bailey, and R. L. Grossman, "Psockets: The case for application-level network striping for data intensive applications using high speed wide area networks," in *Proceedings of Supercomputing 2000*. Dallas, TX: IEEE, November 2000, pp. 240–246.

[23] M. Allman, H. Kruse, and S. Ostermann, "An application-level solution to tcp's satellite inefficiencies," in *Proceedings of the First International Workshop on Satellite-based Information Services (WOSBIS)*, Rye, NY, November 1996.

[24] *Netperf: A Network Performance Benchmark*, Information Networks Division, Hewlett–Packard Company, February 15 1996, http://www.netperf.org/.

[25] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the tcp congestion avoidance algorithm," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 3, July 1997.

[26] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling tcp throughput: A simple model and its empirical validation," in *SIGCOMM: ACM Special Interest Group on Data Communication*, Vancouver, B.C., September 1998, pp. 303–314.

[27] J. Bolliger, T. Gross, and U. Hengartner, "Bandwidth modelling for network-aware applications," in *IEEE INFOCOM – Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, New York, NY, March 1999, pp. 1300–1309.

[28] D. Eastlake, *Secure Domain Name System Dynamic Update*, April 1997, rFC 2137.

[29] A. C. Snoeren and H. Balakrishnan, "An end-to-end approach to host mobility," in *Proceedings of the ACM MOBICOM 2000*, Boston, MA, August 2000.

[30] S. Tilak and N. Abu-Ghazaleh, "A Concurrent Migration Extension to an End-to-End Host Mobility Architecture," *Mobile Computing and Communications Review*, vol. 5, no. 3, pp. 26–31, July 2001.

[31] L. Rizzo, "Dummynet: a simple approach to the evaluation of network protocols," *ACM Computer Communication Review*, vol. 27, no. 1, pp. 31–41, Jan. 1997.

[32] C. Huitema, "Multi-homed tcp," Internet Engineering Task Force," Internet Draft, 1995, expired.

[33] J. Mysore and V. Bharghavan, "A new multicasting based architecture for internet host mobility," pp. 161–172, Sep. 1997.

[34] D. A. Maltz and P. Bhagwat, "MSOCKS: an architecture for transport layer mobility," in *IEEE INFOCOM – Annual Joint Conference of the IEEE Computer and Communications Societies*, San Francisco, California, March/April 1998, pp. 1037–1047.

[35] C. Huitema and R. Draves, "Host-centric IPv6 multihoming," Internet Engineering Task Force," Internet Draft, Oct. 2001, work in progress. [Online]. Available: http://www.ietf.org/internet-drafts/draft-huitema-multi6-hosts-00.txt

[36] M. Crawford and C. Huitema, *DNS Extensions to Support IPv6 Address Aggregation and Renumbering*, July 2000, rFC 2874.

[37] R. Moskowitz, "Host identity payload," February 2001, http://homebase.htt-consult.com/draft-moskowitz-hip-arch-02.txt.

[38] M. Riegel and M. Tuexen, Ed., "Mobile sctp," August 2003, http://www.ietf.org/internet-drafts/draft-riegel-tuexen-mobile-sctp-03.txt.

[39] S. J. Koh and Q. Xie, "msctp with mobile ip for transport layer mobility," August 2003, http://www.ietf.org/internet-drafts/draft-sjkoh-mobile-sctp-mobileip-02.txt.

[40] S. Kent and R. Atkinson, *IP Authentication Header*, November 1998, rFC 2402.

[41] Y. Ko and N. H. Vaidya, "Location-Aided Routing (LAR) Mobile Ad Hoc Networks," in *Proceedings of MOBICOM'98, Dallas*, Oct. 1998.

[42] D. Gu, G. Pei, M. Gerla, and X. Hong, "Integrated hierarchical routing for heterogeneous multi-hop networks," in *Proceedings of the IEEE MILCOM, Los Angeles, CA*, October 2000.
[43] G. Pei and M. Gerla, "Mobility management for hierarchical wireless networks," *ACM/Kluwer Mobile Networks and Applications*, 2000.