pTCP: An End-to-End Transport Layer Protocol for Striped Connections [*]

Hung-Yun Hsieh and Raghupathy Sivakumar
School of Electrical and Computer Engineering
Georgia Institute of Technology
Email: {hyhsieh, siva}@ece.gatech.edu

Abstract

The TCP transport layer protocol is designed for connections that traverse a single path between the sender and receiver. However, there are several environments in which multiple paths can be used by a connection simultaneously. In this paper we consider the problem of supporting striped connections that operate over multiple paths. We propose an end-to-end transport layer protocol called pTCP that allows connections to enjoy the aggregate bandwidths offered by the multiple paths, irrespective of the individual characteristics of the paths. We show that pTCP can have a varied range of applications through instantiations in three different environments: (a) bandwidth aggregation on multi-homed mobile hosts, (b) service differentiation using purely end-to-end mechanisms, and (c) end-systems based network striping. In each of the applications we demonstrate the applicability of pTCP and how its efficacy compares with existing approaches through simulation results.

1. Introduction

The TCP transport layer protocol is designed for connections that traverse a single path between the source and the destination. If a TCP connection is striped over multiple network paths, the performance of the aggregate connection can potentially be even worse than the bandwidth available along any of the paths. This is because TCP expects a predominantly first-in-first-out delivery of packets through the network, and uses out-of-order arrival of packets as indication of packet losses due to congestion. A seemingly better solution of using *application layer striping* over multiple TCP sockets [2, 6, 12] (one each for every path available), does not fare well either due to a variety of reasons, the primary one being the inability to accurately profile bandwidths available along individual paths at the application layer. While we discuss the performance of the application layer striping approach later in the paper, we contend that TCP as-is is not an appropriate transport protocol for striped connections.

On the other hand, striping of connections can have considerable benefits in a variety of Internet settings. For example, a multi-homed mobile host can stripe connections on its multiple wireless interfaces, and thus achieve aggregate bandwidths in a typically bandwidth scarce environment. Striping of connections on the backbone Internet has potential advantages including better utilization of network resources and increased fairness to the connections. While there are several such gains to be attained through striping, the focus of this paper is not to motivate connection striping. Instead, we consider the question: *If connections are to be striped, how can a transport protocol support such connections effectively?* We measure effectiveness as the ability of the transport protocol to deliver the aggregate bandwidths available along the multiple paths to the application. A subset of the challenges such a transport protocol has to address include: (a) the different paths available to the striped connection can have diverse characteristics in terms of bandwidth, delay, and loss; (b) path characteristics can be prone to both short-term and long-term fluctuations; (c) in the worst case, a path can experience a complete shutdown; and (d) the buffer at the receiver is finite irrespective of whether it is partitioned evenly among the different paths or is shared more dynamically.

In this context, we propose an end-to-end transport layer protocol called *pTCP* (parallel TCP) that effectively supports striped connections. Although the pTCP design does not require any specific behavior from the component flows of the striped connection, in this paper we focus only on the case where a component flow exhibits the same behavior as that of a regular TCP connection. pTCP achieves effective aggregation of bandwidth for a striped connection through a combination of unique mechanisms including: (a) decoupling functionalities pertaining to the aggregate connection from those that pertain to an individual path; (b) effective striping across the multiple paths based on the in-

[*]This work was supported in part by funds from NSF (award #ANI-0117840) and Yamacraw (http://www.yamacraw.org).

stantaneous bandwidths; and (c) appropriate re-striping and redundant striping of packets during periods of fluctuation in path characteristics.

As pointed out earlier, a protocol such as pTCP can have applications in several different settings. We identify and focus on the following three application domains for pTCP: (a) bandwidth aggregation at mobile hosts with multiple wireless interfaces, (b) end-to-end weighted rate service differentiation, and (c) striping on an overlay network based on application layer switching. For each of the settings, we discuss the applicability of pTCP, why pTCP is superior to existing approaches in each of the domains, and present simulation results that substantiate our arguments.

The contributions of this work are thus twofold:

- *We propose an end-to-end transport protocol called pTCP for striped connections. pTCP uses a combination of mechanisms to provide a striped connection with the aggregate bandwidth available along the multiple paths.*
- *We also show the use of pTCP in three different Internet settings, and demonstrate its efficacy when compared to existing approaches in those settings.*

The rest of the paper is organized as follows: Section 2 lists the key design principles that underlie pTCP. Section 3 presents the software architecture and protocol overview of pTCP. Section 4 discusses the three application domains for pTCP. Finally, Section 5 discusses some related work and concludes the paper. In the rest of the section, we define some of the key terminology that we use in the paper.

1.1. Terminology

- *Striped Connection:* We refer to a connection that has multiple delivery paths between the source and the destination as a striped connection.
- *Micro-Flow:* We refer to the portion of a striped connection that traverses a single individual path as a component flow or a micro-flow. Hence, several micro-flows together comprise the striped connection. In this paper, we assume that a micro-flow will exhibit the same behavior as a normal TCP connection.
- *Aggregate Bandwidth:* We refer to the sum of the bandwidths available for each of the micro-flows as the aggregate bandwidth. Note that the challenge of providing the aggregate bandwidth to the striped connection is non-trivial.

2. pTCP Design

The key obstacle to achieving the aggregate bandwidth for striped connections is that each of the individual paths can have vastly differing characteristics in terms of bandwidth and delay (round-trip time). If data-striping is done without taking into account these differences, the bandwidth achieved by the striped connection can be significantly lower than the maximum possible. For example, consider a connection striped across two paths $p1$ and $p2$ with bandwidths of $b1$ and $b2$ respectively (where $b1 < b2$), and the same delay d. If data is striped across the two paths equally, the bandwidth achieved is theoretically $(2 * b1)$ which is smaller than the ideal bandwidth of $(b1 + b2)$.

However, in TCP the achieved bandwidth can be even smaller. Consider a fast micro-flow $f2$ co-existing with a slower micro-flow $f1$. If data is striped equally, $f2$ is likely to experience an overflow of its receive buffer (since packets will be buffered waiting for in-sequence packets to arrive through $f1$). In TCP, when a window advertisement of zero is made, the sender enters the *persist state* and can take a considerable amount of time to recover (either through an explicit window update sent by the receiver, or through the expiry of TCP's persist timer that has a starting value of 5 seconds, and a maximum of 60 seconds [17]). In fact, this can induce a domino effect of micro-flows alternately entering persist states if any of the explicit updates are lost. Not surprisingly, a simple scheme that stripes data down a path till the send buffer of the micro-flow blocks does not perform well either, as long as the size of the send buffer is different from the bandwidth-delay product of the micro-flow (which is typically true in TCP).

The problems that arise due to bandwidth differences can be solved by making sure that the data-striping ratio is the same as the ratio of the bandwidths of the different paths. From the standpoint of TCP, this translates into striping data based on the ratio of the $\frac{congestion\_window\_size}{round\_trip\_time}$ values for the different micro-flows. The problems that arise due to delay differences, on the other hand, can be solved by simply over-allocating the receive buffer of the striped connection. Specifically, for k paths where path p_i has bandwidth b_i and delay d_i, a total buffer allocation of $(\Sigma b_i) * max(d_i)$ (as opposed to $\Sigma(b_i * d_i)$) will absorb the problems caused by delay differences. Similar observations have also been made in [9]. In the rest of the paper, we assume the use of such buffer provisioning at the receiver to overcome delay differences.

The problems identified this far are further exacerbated when fluctuations of path characteristics occur. For example, when the bandwidth of a path p_i decreases, for a transient period the number of packets outstanding on the path is disproportional to its bandwidth. If bandwidth fluctuations are a norm (like in wireless environments), this can result in a failure to keep the striping ratio the same as the ratio of the bandwidths. In the rest of the section, we list the fundamental design elements in pTCP that are targeted toward addressing the problems discussed above.

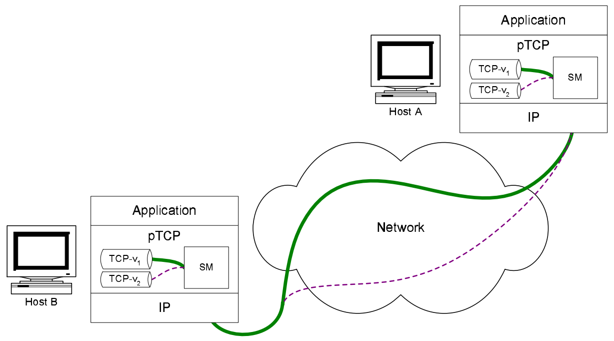


Figure 1. Use pTCP for Striped Connections

- *Decoupling of Functionalities:* pTCP decouples functionalities associated with per-path characteristics from those that pertain to the aggregate connection. We refer to the component in pTCP that handles aggregate connection functionalities as *SM* (Striped connection Manager), and the component in pTCP that handles per-path functionalities as *TCP-v* (TCP-virtual). TCP-v is a modified version of TCP that deals only with virtual packets and virtual buffers.[1] Each micro-flow of a striped connection is controlled by an independent TCP-v.

  Congestion control, the mechanism that estimates the available bandwidth along a path, is obviously a per-path functionality and is handled by TCP-v. Since SM is responsible for striping data across the different TCP-vs, it handles *buffer management* (including sequencing at the receiver), and consequently *flow control*. *Reliability* in pTCP is primarily handled by TCP-v just as in TCP. However, under certain circumstances that we describe later, SM can intervene and cause loss recovery to occur along a path different from the one on which the loss occurred. Finally, *connection management* is handled by SM, although it recursively uses the connection management performed by individual TCP-vs. Figure 1 illustrates the roles of the two components in pTCP.

- *Delayed Binding:* Although striping based on the $\frac{cwnd}{rtt}$ ratios is mostly sufficient to achieve aggregate bandwidths, such a scheme will remain vulnerable to instantaneous fluctuations in the bandwidth and delay of a path. pTCP uses a delayed binding strategy that satisfies the requirement of striping based on ratios of $\frac{cwnd}{rtt}$, and at the same time dynamically adapts to instantaneous fluctuations in bandwidth and delay. Specifically, since the buffer management is performed by SM, TCP-v operates only on virtual buffers. Thus, when a packet is transmitted by TCP-v, the virtual packet should be bound to the real data in the buffer at SM. pTCP performs this binding only just before the virtual packet is transmitted. This coupled with TCP's self-clocked property results in pTCP adapting the data-striping to instantaneous changes in bandwidth and delay. Looking at it from another perspective, any packet bound to any TCP-v will have already been transmitted or will be in the process of being transmitted, precluding any chances of packets being held up in buffers at the source due to bandwidth fluctuations. An exception to this rule is when there is a packet loss and the lost packet falls outside the reduced congestion window of the concerned TCP-v. In this case, the lost packet is bound to a particular TCP-v, but is not in transit through the network. We describe how pTCP handles such a situation in the next design element.

[1]We refer to a packet with the data portion stripped off as a virtual packet, and a buffer of virtual packets as a virtual buffer.

- *Packet Re-striping:* In steady state, pTCP will ensure that the number of outstanding packets in a micro-flow is proportional to the bandwidth along the corresponding path. Moreover, the delayed binding strategy further ensures that all bound packets are already in transit in the network. However, during congestion when packets are lost in the network, the reduction of the congestion window by a TCP-v can result in bound packets falling outside the congestion window. If such packets are lost during the congestion, then they will remain un-transmitted till the congestion window of that TCP-v expands beyond their sequence numbers. This can potentially result in an overflow of the receive buffer if the other micro-flows are active in the meantime, finally resulting in a connection stall. pTCP handles the problem by unbinding packets that fall outside of the congestion window of TCP-v they were assigned to. Such unbinding results in those packets being available to be re-assigned to the next TCP-v that can send more data.

  We thus define re-striping as the process of sending a packet that was earlier transmitted through a micro-flow i, through another micro-flow j, where $j \neq i$. Note that such re-striping can presumably re-transmit data that has already been successfully delivered by the old TCP-v. This can very well be construed as an undesirable overhead. However, recall that such re-striping will be performed only during bandwidth fluctuations. If such fluctuations are rare (say the only losses are due to probe losses), then the re-striping overhead is insignificant. On the other hand, if such fluctuations are frequent, the benefits of the re-striping far outweigh the overheads incurred due to any unnecessary re-transmissions. *The results that we present in Sec-*

tion 4 show considerable performance improvement for pTCP even after accounting for the re-striping overheads.

- *Redundant Striping:* The re-striping mechanism described earlier is equivalent to a *move* operation that shifts a packet from one TCP-v to another. However, note that a packet that still remains inside the congestion window of a TCP-v will not by re-striped by SM. Consider a scenario in which a path has shutdown completely (e.g. blackout of a wireless channel). In this case, the first packet in the congestion window of the concerned TCP-v will never be re-assigned, thus potentially resulting in a connection stall. pTCP handles such cases by *redundantly striping the only packet inside the congestion window through another micro-flow* every time a timeout is experienced. A *copy* operation is performed (the packet is bound to the new micro-flow) as opposed to a *move* operation because the old TCP-v needs at least one packet to probe for recovery from the shutdown.

- *Selective Acknowledgements:* Although pTCP does not stipulate the use of any specific congestion control or reliability mechanism at TCP-v, the use of SACK information by TCP-v can significantly improve the striped connection's performance. While re-striping helps in recovering faster from loss of packets that have fallen outside the reduced congestion window of a TCP-v, SM does not intervene in the case of a lost packet that lies inside the congestion window. If there are multiple losses within the congestion window, it can take TCP-v multiple round-trip times to recover from the losses (TCP-Reno or TCP-NewReno will recover from one loss every round-trip time). This can result in the receive buffer overflow problem identified earlier. Using SACK information will enable recovering from the losses in a much shorter time span. Moreover, the availability of SACK information can also be used for more intelligent re-striping decisions and reducing the associated overheads. We do not delve into a SACK based optimization due to lack of space.

3. pTCP Architecture

3.1. Software Architecture

Figure 2 illustrates an overview of the pTCP architecture. Conceptually, SM functions as a wrapper around TCP-v. SM maintains the socket buffers and handles communication with both the higher and the lower layers. Any packet from the higher layer (say application) is queued onto the `send buffer` awaiting further treatment. Similarly, any packet from the lower layer (say IP) is queued onto the `recv buffer` based on its sequence number. Since the packets are handled wholly within SM, TCP-v merely operates on virtual packets and hence virtual buffers.

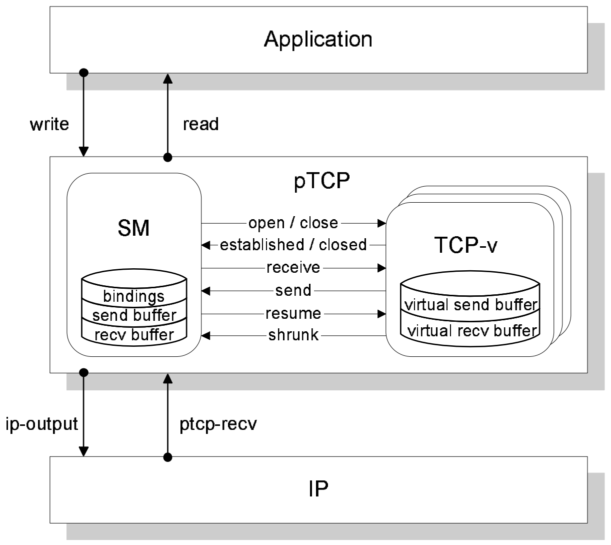


Figure 2. pTCP Architecture Overview

The interactions between SM and TCP-v that are initiated by the former are: (a) $open$, (b) $close$, (c) $receive$, and (d) $resume$. The interactions initiated by TCP-v are: (a) $established$, (b) $closed$, (c) $send$, and (d) $shrunk$. When SM receives a packet, it processes the packet, strips the data, enqueues the data onto the receive buffer, and sends the skeletal packet (sans the data) to the appropriate TCP-v through the $receive$ interface. When TCP-v receives the packet, it enqueues the packet onto its virtual buffer, updates its local state, and sends any required acknowledgement packets. If a TCP-v has space in its congestion window to send new data, it invokes the $send$ function with the appropriate virtual packet. SM on receiving the call binds the virtual packet to the next in-sequence unsent data packet, maintains the binding in `bindings`, and sends it out. If SM does not have new data to send, it responds with a `FREEZE` message to the concerned TCP-v. TCP-v continues to invoke the $send$ function till either there is no more space left in its congestion window, or it receives a `FREEZE` from SM. In either case, the concerned TCP-v goes to sleep. SM maintains a list of $active$ micro-flows consisting of those to which it has issued a `FREEZE`. When new data is received from the application, SM uses the $resume$ message to notify all active micro-flows. The micro-flows then initiate the send requests as before. If a TCP-v had gone to sleep due to running out of free congestion window space, it would be awoken through a $receive$ call that can potentially open up more space.

The $open$ and $close$ functions are used by SM to recursively invoke the $open$ and $close$ functions of TCP-v during connection setup and teardown. The *established* and *closed* calls are used by TCP-v to inform SM of successfully accomplished *open* and *close* requests. The *shrunk* call is used by TCP-v to inform SM about any reduction in the congestion window size and the consequent dropout of packets from within the window.

3.2. Protocol Overview

- *Connection Management:* We assume that the number of TCP-v micro-flows to open is conveyed to pTCP when the application opens the pTCP socket. The connection setup involves recursively invoking the setup of the TCP-v micro-flows. pTCP allows data transfer to begin as soon as one of the micro-flows reaches the established state in the TCP state diagram. Similarly, for connection teardown, the close command is issued to all TCP-vs. Unlike for open, pTCP waits for all the micro-flows to be closed before returning the closed status to the application.

- *Congestion Control:* Congestion control is the sole responsibility of TCP-v. Although pTCP does not require a specific congestion control protocol to be used for a micro-flow, the discussions in this paper assume the use of TCP's congestion control scheme. One of the salient features of pTCP is the decoupling of the SM functionality from that of TCP-v. This in turn allows independent and tailored congestion control schemes to be used along the different paths.

- *Reliability:* pTCP primarily relies on the individual TCP-vs to support reliability. A packet once bound to a TCP-v is expected to be delivered reliably. The one exception to this rule is when SM unbinds a packet and re-stripes it on a different path. In this scenario, when (and if) the old micro-flow attempts a re-transmission of the packet, a new packet might be bound to the transmission. Therefore, a re-transmission at TCP-v does not need to be a re-transmission for the striped connection. This is possible because of the decoupling of functionalities between SM and TCP-v.

- *Flow Control:* Since TCP-v operates only with virtual buffers, it does not need to perform flow control. Hence, our current implementation of pTCP uses the window advertisement field already available in the TCP header for performing flow control at SM.[2] Because the overall available buffer size is an upper bound on the number of transmissions a single TCP-v can perform, the TCP flow control mechanisms do not interfere with the TCP-v operations even if such overloading of the window field is done. Note that all TCP-vs see the same advertisement and might individually attempt to transmit (subject to congestion window availability) enough data to fill in the advertised buffer space. SM, being a single point of departure for all transmissions, tackles this problem by sending a FREEZE message to the concerned TCP-vs once it knows that the total number of outstanding packets is equal to the advertised window space.

[2]Note that window scaling may be necessary since the buffer size is multiple times that of a single regular TCP connection's buffer size.

Due to lack of space we do not present the pTCP state diagram, handshakes during connection setup and teardown, and packet header formats, but instead refer interested readers to [7] for more information on these subjects.

4. pTCP Instantiations

In this section, we investigate the applicability of pTCP in the following three settings: (a) bandwidth aggregation at a mobile host with multiple wireless interfaces, (b) end-to-end approach to weighted rate differentiation, and (c) striping on an overlay network constructed through application layer switching. For each of the application domains, we discuss the key characteristics of the problem, the applicability of pTCP, and simulation results that compare pTCP to existing solutions.

4.1. Multi-homed Striping

A mobile user today has a variety of options in terms of the wireless access technologies that can be used for Internet connectivity. It is very likely that a mobile device has multiple interfaces connecting to two or more of such options. For example, a scenario in which the mobile device has a wireless LAN interface along with a wireless WAN interface is very conceivable. When such multi-homed hosts exist, an interesting question is: how are the different interfaces used in tandem?

Vertical handoffs schemes have been proposed wherein the mobile device is handed off from one access network to another depending upon its location [14]. For instance, the wireless LAN interface can be used as long as the device is inside the building. When the device moves outside the building, a vertical handoff to the wireless WAN network can be performed. Note that in the above scenario, it is possible for the mobile device to be connected to *both interfaces* when the device remains inside the building. However, an important issue is whether a connection that is striped across the two interfaces can indeed provide the aggregate bandwidth. If effective bandwidth aggregation can be achieved, connections to or from the mobile device can enjoy significantly better bandwidth performance, especially when the bandwidths across the different interfaces are of the same order.

Link layer striping approaches such as [13] do not perform well in this context because the multiple interfaces usually belong to different access networks, and an ideal striping algorithm will need to address fluctuations in capacity caused by the end-to-end multi-hop nature of the paths. On the other hand, bandwidth-based transport layer striping approaches [9] that rely on explicit bandwidth probing of individual paths to obtain accurate estimation and thus to stripe accordingly may also suffer when aggressive probing cannot be performed (e.g. due to bandwidth constraints) or when fluctuations of link bandwidths occur at a finer time-scale than the probing period.

4.1.1. Applicability

Several properties of pTCP make it applicable to the above problem setting. pTCP uses a delayed binding strategy that handles short-term bandwidth variations common in wireless environments. The re-striping mechanism used by pTCP overcomes frequent but more permanent bandwidth variations due to reasons such as congestion. The decoupling of congestion control performed by the TCP-v from the other mechanisms at SM through a well-defined interface enables use of tailored congestion control protocols for each wireless link [7]. Finally, pTCP explicitly addresses potentially blackouts for micro-flows through its redundant striping technique.

4.1.2. Performance

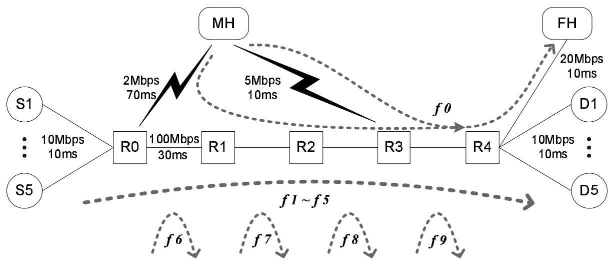


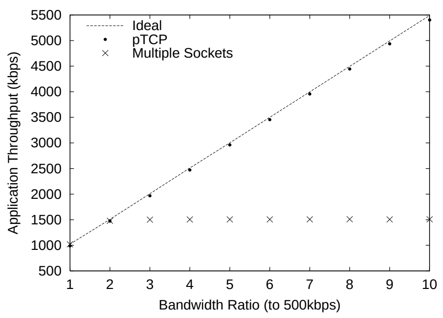
Figure 3. Network Topology for Multi-homed Striping

In this section, we present the simulation results showing the performance of pTCP when used in a multi-homed mobile host for bandwidth aggregation. We use the *ns-2* network simulator [15] for the network topology shown in Figure 3. For simplicity the backbone routers also double up as wireless access points (or base stations) that the mobile host (MH) can use to communicate with a remote fixed host (FH). The mobile host primarily uses two different types of wireless connections: (a) link MH-R0 with bandwidth ranging from 500Kbps to 2Mbps (depending upon the scenario) and a 70ms access delay, and (b) link MH-R3 with bandwidth ranging from 500Kbps to 5Mbps (depending upon the scenario) and a 10ms access delay. Together with the propagation delay experienced in the backbone network, these values simulate a WWAN connection with 400ms round-trip time and a WLAN connection with 100ms round-trip time respectively. MH utilizes both links using either application layer striping over multiple TCP sockets (data is striped through each socket till the socket buffer fills up) or pTCP. Besides the striped connection between MH and FH, we use 9 other flows to simulate the background traffic: 5 long TCP flows with a round-trip time of 280ms, and 4 short UDP flows with 10Mbps data rate and 100ms round-trip time as shown in the figure. We use TCP-SACK for all TCP flows including pTCP micro-flows. The simulation is run for 600 seconds and the results are obtained by averaging 10 runs. We measure the throughput delivered to the application at the destination (FH) as the metric to evaluate pTCP's performance. We also present an "ideal" aggregate throughput that is obtained by summing the throughputs of two independent TCP flows (used by two different applications) for the same scenario.

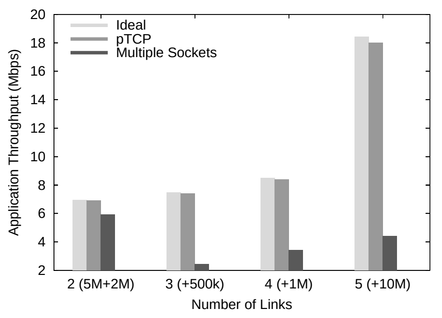
Figure 4(a) shows the simulation results when the bandwidth of the WWAN link is 500Kbps and the bandwidth of the WLAN link varies from 500Kbps to 5Mbps. pTCP is able to achieve the aggregate throughput even when the bandwidth ratio between two links goes to 10. However, by using multiple sockets the MH cannot enjoy the aggregate throughput when the ratio is greater than 2. *This result reinforces our argument that using multiple sockets cannot achieve effective bandwidth aggregation as discussed in Section 2.*

Figure 4(b) compares the performance of pTCP and multiple sockets when the number of wireless links increases from 2 to 5. Besides the 2Mbps link with 400ms round-trip time (MH-R0)and 5Mbps link with 100ms round-trip time (MH-R3), in sequence we add a 500Kbps link with 300ms round-trip time (MH-R1), a 1Mbps link with 200ms round-trip time (MH-R2) and a 10Mbps link with 50ms round-trip time (MH-R4). The bandwidth of the access link to FH is increased to 20Mbps to absorb the traffic from MH. As shown in the figure, pTCP is able to achieve effective bandwidth aggregation for multiple links. On the other hand, the aggregated bandwidth by using multiple sockets is throttled by the slowest 500Kbps link (the aggregate throughput is approximately equal to $500\text{Kbps}*(2*4+1) = 4500\text{Kbps}$ using 5 links).

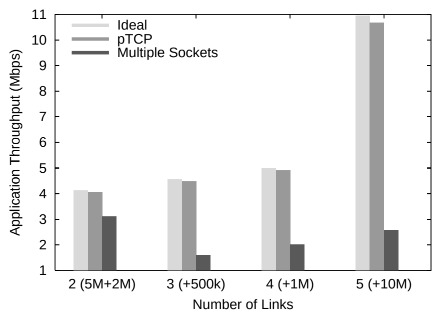
Figure 4(c) shows the simulation results when all wireless links used in Figure 4(b) experience bandwidth fluctuations (say, due to channel dynamics). We simulate the scenario when the bandwidth of individual link randomly



(a) Rate Differential



(b) Multiple Links (Stable)



(c) Multiple Links (Fluctuating)

Figure 4. pTCP Performance for Multi-homed Striping

fluctuates between 20% to 100% of the normal values. The fluctuating period of each wireless link is set to 10, 3, 6, 20, and 5 seconds respectively. Under such fluctuating conditions, the performance of pTCP still closely follows that of the “ideal” bandwidth aggregation.

4.2. Weighted Rate Differentiation

Relative end-to-end service (rate) differentiation involves providing connections with rates relative to pre-assigned weights. For example, if two connections (with potentially different source-destination pairs) with weights $w1$ and $w2$ share the same bottleneck, their respective achieved rates $r1$ and $r2$ should be in the ratio $w1$:$w2$. Although several QoS architectures and service provisioning schemes have been proposed to address this problem, the challenge lies in achieving the rate differentiation using *purely end-to-end techniques* without relying on underlying network infrastructure.

There are two key lines of approaches for achieving end-to-end service differentiation. First, a connection with weight w is actually composed of w micro-flows of the default weight of one (note that the w micro-flows share the same path). We refer to this approach as the *striped approach*. The connection, in this case, would theoretically receive w times the bandwidth enjoyed by a default connection *provided the aggregate bandwidth can be achieved*. The second approach involves using just one micro-flow, but adapting the congestion control to achieve a behavior that is equivalent to the aggregate behavior of w default micro-flows. The former approach has thus far been considered to be a non-viable solution because of the problems in effectively aggregating the bandwidths offered by the micro-flows [10]. On the other hand, related work that adopts the latter approach either does not scale beyond nominal weights (e.g. 10 in the case of mulTCP [4]), or relies on unrealistic assumptions (e.g. TCP-LASD [10] assumes accurate loss profiling), thus limiting the applicability.

4.2.1. Applicability

A clear application of pTCP is the striped approach where w micro-flows are used. Because of pTCP’s unique ability to aggregate bandwidths across micro-flows that have widely differing and fluctuating bandwidths, the effective bandwidth perceived by the connection will indeed be w times that of a default connection. Two key issues of the striped approach to achieving rate differentiation are: (a) *Widely varying instantaneous bandwidths across the micro-flows.* Although the average throughput enjoyed by each individual micro-flow will approximately be the same, the instantaneous throughputs can be very different. (b) *Highly fluctuating bandwidths.* The bandwidth enjoyed by a micro-flow can by itself be fluctuating, especially when the number of micro-flows increases (for larger weights). pTCP handles the first issue through its delayed binding mechanism, and addresses the second issue by its packet re-striping strategy.

4.2.2. Performance

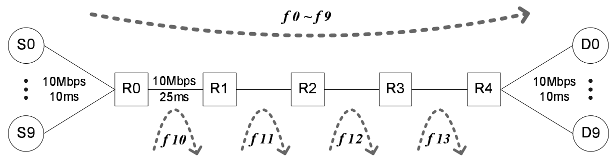
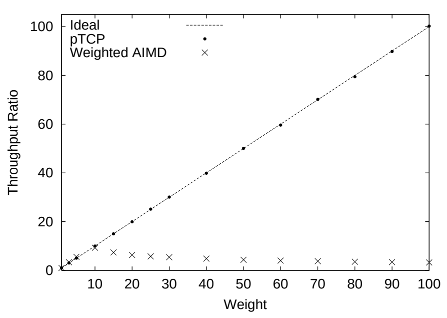
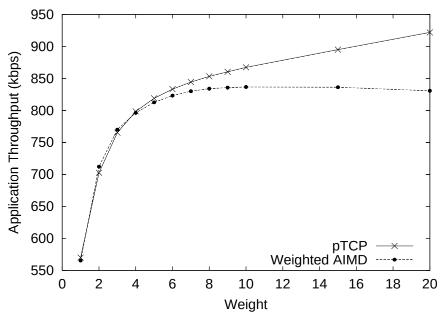


Figure 5. Network Topology for Weighted Rate Differentiation

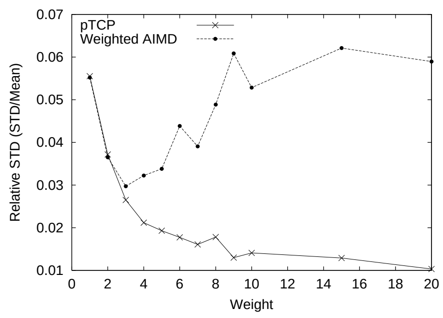
Figure 5 shows the network topology used for evaluating the weighted rate differentiation instantiation of pTCP. It consists of 10 TCP flows with each having a 240ms round-trip time in a multi-link bottleneck environment. All backbone routers use RED queues with buffer size approximately equal to the bandwidth-delay product of the bottleneck link. The configuration roughly emulates the multi-



(a) Throughput Ratio (1 pTCP)



(b) Throughput Mean (10 pTCPs)



(c) Throughput Unfairness (10 pTCPs)

Figure 6. pTCP Performance for Weighted Rate Differentiation

plexing of a large number of flows at the backbone routers where each packet experiences approximately equal drop probability. We use two scenarios to compare the performance of pTCP and WAIMD (Weighted AIMD) [4] in providing service differentiation to end-to-end flows: (a) 1 pTCP/WAIMD flow with weight w and the other 9 regular TCP flows with weight 1, and (b) all 10 flows are pTCP/WAIMD flows with the same weight w. The first scenario is targeted toward evaluating the throughput scalability of pTCP with increasing weights, and the second scenario evaluates pTCP's fairness properties in this context. We also introduce 4 short TCP flows with 90ms round-trip time as the background traffic. We use TCP-SACK for all TCP flows and measure the throughput delivered to the application.

Figure 6(a) shows the simulation results for the first scenario where the weight of the pTCP/WAIMD flow increases from 1 to 100, while the other 9 flows are regular TCP flows with a weight of 1. It is clear that when using WAIMD the throughput of the weighted flow does not scale beyond 10. On the other hand, the performance of pTCP closely follows the ideal curve even when the weight is 100. The simulation result thus shows the effectiveness of pTCP in achieving scalable weighted service differentiation without relying on accurate loss measurement as in [10].

In Figure 6(b) and Figure 6(c) we show the performance of pTCP in the second scenario where all of the 10 flows are weighted flows with the same weight. The background TCP flows still have a weight of 1. As the weight of the 10 flows increases, the background TCP flows yield more bandwidth causing the throughput of the weighted flow to increase. As shown in the figure, however when WAIMD is used the throughput of the weighted flows saturates at $w = 8$ and starts decreasing afterward. This is a fundamental problem with the WAIMD scheme where each weighted TCP flow becomes sensitive to packet losses and its performance deteriorates as w increases. Moreover, although ideally all 10 flows should enjoy the same throughput we show in Figure 6(c) the throughput unfairness (standard deviation normalized to the mean) observed among the 10 flows. Because pTCP is able to effectively aggregate the throughput of the component micro-flows, as weight increases any unfairness introduced in the router (e.g. due to unequal packet drop probability) on individual micro-flows is reduced. On the other hand, since WAIMD flows become sensitive to packet losses at large weights, the unfairness in fact increases as the weight increases. Therefore, we show that using pTCP for end-to-end service differentiation results in better scalability and fairness than that demonstrated in related work [4, 10].

4.3. Application Layer Switching

Several Internet performance studies have identified two characteristics of the Internet: (a) Predominantly, there exists a path different from the default one provided by the underlying routing infrastructure that can provide better service (in terms of bandwidth or delay) [11]. (b) The Internet infrastructure remains under-utilized in pockets, while it is overloaded in other areas [3]. A combination of the above two observations has led to research in exploiting such unbalanced use of the Internet and providing better quality of service (QoS) to connections. Since it is desirable for any Internet solution to be easily deployable, an approach that is based on an overlay network constructed through purely application layer switching holds great promise.

The idea is to construct an overlay network without changing the existing Internet infrastructure. The overlay network will comprise of only Internet end-hosts that will act as application layer switches.[3] The participating end-hosts also double up as QoS monitors, measuring the QoS observed through the "links" on the overlay network. Traffic on the overlay network can be routed through multiple "hops" based on the QoS observed on the different possible routes. Thus, although the default path from say a host in

[3]Note that although conceptually we refer to such a technique as application layer switching, a practical implementation can be more efficient (say using IPv4 source routing).

Chicago to a host in Washington might exist, traffic between the two hosts might be routed through another host in San Francisco if the alternate path provides better QoS.

A critical obstacle in realizing such an approach lies in the QoS monitoring at the participating end-hosts. Most of the studies that have identified the existence of better alternate paths have done so using invasive probing techniques, physically injecting data into the network paths. Such an approach as a long-term solution would however be quite undesirable since it consumes valuable network resources. Inferring available QoS from existing connections between the participating end-hosts might be a tall order as the traffic on a "link" between two end-hosts might be composed of traffic from non-participating end-hosts as well. Invasive probing in intervals can render the routing decisions potentially stale, and the overheads of probing still remain.

In this context, we consider a solution wherein k different paths on the overlay network (between the source and the destination) that are coarsely found to exhibit the best QoS are chosen, and the end-to-end connection is striped across the k paths. As long as the aggregate bandwidth can be delivered to the connection, such an approach would entail only finding the k best paths as opposed to the best path, and hence is more feasible. Even infrequent invasive probing can establish the k best paths on the overlay network. Such an approach would allow the striped connection to always enjoy the services along the best path as long as it lies within the k chosen paths.

Note that in order to alleviate unfairness issues, all connections on the overlay network can be striped over k paths. As k increases, not only should the throughput and fairness properties experienced by connections increase, but the utilization of network resources should also improve considerably. Even when the network is saturated, a larger k should result in better global fairness between the connections in the network.

4.3.1. Applicability

The applicability of pTCP in this domain is quite evident. Striping of connections is advantageous as long as the aggregate bandwidths can be achieved. The different paths available for a striped connection can exhibit both long-term and short-term bandwidth and delay fluctuations. pTCP is designed to exactly operate in such scenarios and provide the striped connection with the aggregate bandwidth.

4.3.2. Performance

Figure 7 shows the network topology used to simulate striping using application layer switching in an overlay network. We assume existence of 10 different paths between adjacent backbone routers. All of the 40 backbone links

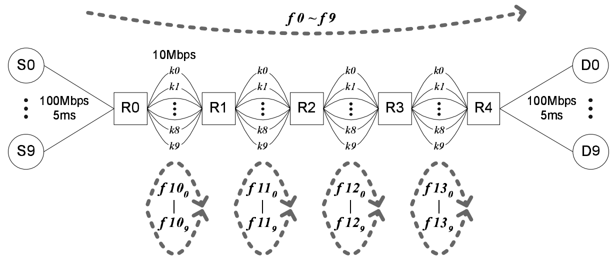
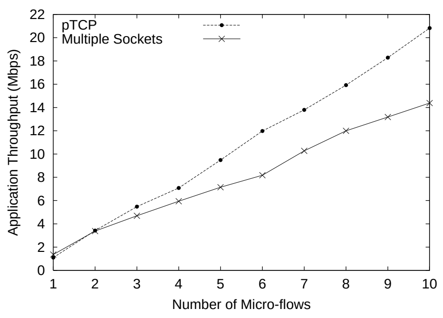


Figure 7. Network Topology for Application Layer Switching

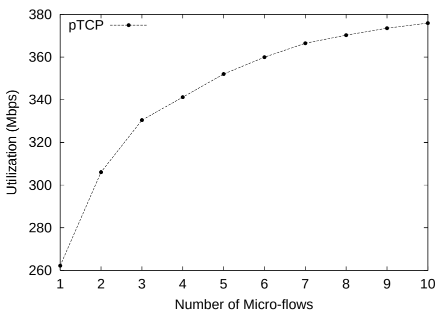
are 10Mbps links with propagation delay randomly chosen from 5ms to 30ms. We use a UDP flow on each of the backbone link (total 40 UDP flows) to emulate the background traffic. The data rates of the UDP flows are randomly chosen from 3Mbps to 7Mbps and is changed after every 10-second period. This configuration thus represents an average of 50% utilization on the backbone links. Note that in the *ns-2* simulation, because different links originating from the same node are in fact independent of each other (they do not share the same queue), using such network topology is not limited to the scenario when all routers have 10 fan-in and fan-out links. Instead, it emulates a more generic topology where there are 10000 different paths (each may be with a different round-trip time and traffic loads but not necessarily disjoint) between $R0$ and $R4$. We use 10 TCP flows and measure their throughput at the respective receiving applications. We use two scenarios to show the performance of pTCP in such an environment: (a) 1 pTCP flow with p micro-flows and 9 regular TCP flows, and (b) 10 pTCP flows each with p micro-flows. Each micro-flow takes on a path randomly chosen from the 10000 end-to-end paths existing in the backbone network.

Figure 8(a) shows the throughput enjoyed by the single pTCP flow when all the other 9 flows are regular TCP flows. As the the number of micro-flows composing the pTCP flow increases, the throughput for the pTCP flow increases by utilizing the unused resources in the network.

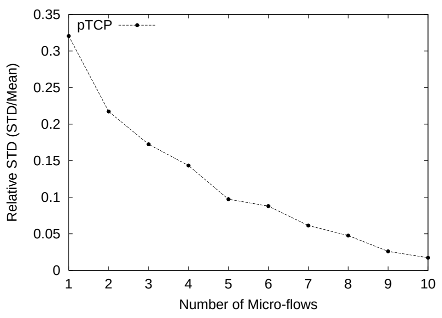
In Figure 8(b) and Figure 8(c) we show the simulation results when all the 10 flows are pTCP flows with the same number of micro-flows (each of them randomly takes on an end-to-end path in the backbone network). Figure 8(b) shows the network utilization that is measured by summing the throughput of background UDP traffic and the end-to-end throughput of all pTCP flows multiplied by the number of links (which is 4), and Figure 8(c) shows throughput unfairness among the 10 flows using the normalized standard deviation. *It is clear that not only does the network utilization increase (the upper bound is 400Mbps) as each pTCP uses more micro-flows, but the unfairness (due to micro-flows traversing different bottleneck links) also reduces.*



(a) Application Throughput (1 pTCP)



(b) Network Utilization (10 pTCPs)



(c) Network Unfairness (10 pTCPs)

Figure 8. pTCP Performance for Application Layer Switching

5. Related Work and Conclusions

Striping is a technique used for the transparent aggregation of multiple resources to obtain higher performance [16]. It has been typically used at the link layer to aggregate bandwidths of different links with similar characteristics [1, 5]. Recently, several application layer striping approaches that open multiple TCP sockets in parallel to achieve higher throughput have also been proposed [2, 6, 8, 12]. However, these approaches deal with increasing application throughput by using multiple TCP connections *through the same physical path*. If such approaches are used for striped connections over different paths, the problems identified in Section 2 would render them ineffective.

In this paper, we propose a transport protocol called pTCP that is suited for supporting striped connections. pTCP achieves aggregate bandwidths for striped connections even in the presence of disparities among the different paths, and fluctuations in path characteristics. We show the applicability of pTCP in three different application domains: (a) bandwidth aggregation on multi-homed mobile hosts, (b) end-to-end service differentiation, and (c) striping on overlay networks based on application layer switching. pTCP demonstrates considerably improved performance in each of the three settings when compared to existing approaches.

References

[1] H. Adiseshu, G. Parulkar, and G. Varghese. A reliable and scalable striping protocol. In *Proceedings of ACM SIGCOMM*, Palo Alto, CA USA, Aug. 1996.

[2] M. Allman, H. Kruse, and S. Ostermann. An application-level solution to TCP's satellite inefficiencies. In *Proceedings of WOSBIS*, Rye, NY USA, Nov. 1996.

[3] CAIDA. Internet measurement and data analysis. http://www.caida.org/analysis/performance.

[4] J. Crowcroft and P. Oechslin. Differentiated end-to-end Internet services using a weighted proportional fair sharing TCP. *ACM Computer Communication Review*, 28(3):53–69, July 1998.

[5] J. Duncanson. Inverse multiplexing. *IEEE Communications Magazine*, 32(4):34–41, Apr. 1994.

[6] T. Hacker, B. Athey, and B. Noble. The end-to-end performance effects of parallel TCP sockets on a lossy wide-area network. In *Proceedings of IPDPS*, Fort Lauderdale, FL USA, Apr. 2002.

[7] H.-Y. Hsieh and R. Sivakumar. A transport layer approach fo achieving aggregate bandwidths on multi-homed mobile hosts. In *Proceedings of ACM MOBICOM*, Atlanta, GA USA, Sept. 2002.

[8] J. Lee, D. Gunter, B. Tierney, B. Allcock, J. Bester, J. Bresnahan, and S. Tuecke. Applied techniques for high bandwidth data transfers across wide area networks. In *Proceedings of CHEP*, Beijing, China, Sept. 2001.

[9] L. Magalhaes and R. Kravets. Transport level mechanisms for bandwidth aggregation on mobile hosts. In *Proceedings of IEEE ICNP*, Riverside, CA USA, Nov. 2001.

[10] T. Nandagopal, K.-W. Lee, J.-R. Li, and V. Bharghavan. Scalable service differentiation using purely end-to-end mechanisms: Features and limitations. In *Proceedings of IWQoS*, Pittsburgh, PA USA, June 2000.

[11] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of Internet path selection. In *Proceedings of ACM SIGCOMM*, Cambridge, MA USA, Sept. 1999.

[12] H. Sivakumar, S. Bailey, and R. Grossman. PSockets: The case for application-level network striping for data intensive applications using high speed wide area networks. In *Proceedings of Supercomputing (SC)*, Dallas, TX USA, Nov. 2000.

[13] A. Snoeren. Adaptive inverse multiplexing for wide-area wireless networks. In *Proceedings of IEEE GLOBECOM*, Rio de Janeireo, Brazil, Dec. 1999.

[14] M. Stemm and R. Katz. Vertical handoffs in wireless overlay networks. *Mobile Networks and Applications*, 3(4):335–350, 1998.

[15] The Network Simulator *ns-2*. http://www.isi.edu/nsnam/ns.

[16] B. Traw and J. Smith. Striping within the network subsystem. *IEEE Network*, 9(4):22–32, July 1995.

[17] G. R. Wright and W. R. Stevens. *TCP/IP Illustrated, Volume 2: The Implementation*. Addison-Wesley Publishing Company, Reading, MA USA, Jan. 1995.