Mobile Networks and Applications 9, 363–378, 2004

An End-to-End Approach for Transparent Mobility across Heterogeneous Wireless Networks *

HUNG-YUN HSIEH, KYU-HAN KIM and RAGHUPATHY SIVAKUMAR
School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA

Abstract. With the advent of a myriad of wireless networking technologies, a mobile host today can potentially be equipped with multiple wireless interfaces that have access to different wireless networks. It is widely perceived that future generation wireless networks will exhibit a similar trend in supporting a large variety of heterogeneous wireless access technologies that a mobile host can choose from. In this paper, we consider such a multi-homed mobile host and propose an end-to-end solution that enables the seamless use of heterogeneous wireless access technologies. The unique features of the proposed solution include: (i) a purely end-to-end approach to handle host mobility that requires no support from the underlying network infrastructure, (ii) seamless vertical handoffs when the mobile host migrates from one access network to another, (iii) ability to support different congestion control schemes for a live connection traversing different interfaces, and (iv) effective bandwidth aggregation when the mobile host has simultaneous access to multiple networks. We present the design and details of the proposed approach, and evaluate its performance through simulations and real-life field experiments.

Keywords: heterogeneous wireless networks, multi-homed mobile host, seamless handoff, bandwidth aggregation

1. Introduction

The proliferation in the number of mobile Internet users has been accompanied by the equally staggering increase in the number of wireless access technologies. A key reason behind the mushrooming of heterogeneous wireless access technologies is the performance tradeoffs they exhibit in terms of mobility support, network capacity, coverage area, and transmission power. Hence, a mobile user today can potentially be equipped with multiple wireless interfaces that have access to different wireless networks. While such a multi-homed mobile host may initially choose only one of its wireless interface to access the Internet, during periods of mobility, it is possible that it needs to migrate from one access network to another. The heterogeneity of different wireless access technologies and the diversity of the associated network infrastructure, however, pose great challenges to providing "anywhere, anytime" transparent host mobility to the mobile user.

Current approaches proposed to handle Internet host mobility are heavily dominated by infrastructure-based schemes. For example, vertical handoffs [1,2] assuming the support of Mobile IP [3] have been proposed to ensure continuous connectivity when the mobile host roams between different wireless networks. Various interworking architectures [4–6] have also been proposed to facilitate link-layer handoffs between the WLAN and GPRS (or 3G) systems. Other approaches include providing a basic access network [7] or a communication gateway [8] that the mobile host can access directly without being exposed to the heterogeneity of the backend network infrastructure. In [9] an end-to-end approach (without relying on the infrastructure support) is proposed to handle connection migration when the network address of the mobile host changes during the lifetime of the connection.

Although these approaches address the problem of network interface changes at the mobile host as it migrates from one network to another, we argue that they tackle only a minor subset of the problems a multi-homed mobile host faces when moving across heterogeneous wireless networks. We identify the following key drawbacks in existing approaches that handle host mobility:

(1) *Reliance on infrastructure support:* While Mobile IP has been proposed to handle host mobility at the network layer, a mobile host today cannot rely on Mobile IP if its home network does not provide for a home agent. In addition, the presence of security mechanisms such as ingress filtering can render a Mobile IP re-routed connection inefficient [3,10]. Note that the same problems exist even for IPv6. Although several approaches have been proposed recently to integrate heterogeneous wireless networks for achieving host mobility without incurring the overheads in Mobile IP [4–7], given the increasing heterogeneity of wireless access technologies, the effectiveness of these approaches is greatly limited by the specificity to the networks they are designed for.

(2) *Inability to leverage soft handoffs:* Even if soft handoffs are possible at the link layer, in the absence of any explicit support at a higher layer, an application cannot benefit from the soft-handoff capability when the mobile host moves across heterogeneous networks. Specifically, although TCP migration [9] achieves host mobility without relying on the network support, it performs a "hard handoff" at the transport layer between the new and old TCP states (i.e. transmission control blocks [11]). Hence, despite the soft handoffs achieved at the link layer, the mi-

* This work was funded in part by the National Science Foundation under grant ANI-0117840.

grated connection experiences packet losses and suffers from performance degradation during the handoffs.

(3) *Lack of support for network specific congestion control schemes:* Different congestion control schemes tailored to the markedly different characteristics of heterogeneous wireless environments have been proposed. For example, STP [12] is proposed for satellite networks, while WTCP [13] targets the WWAN environment. Despite the availability of network specific congestion control schemes that the mobile host can use for achieving performance improvement in the target environment, no existing solutions allow the mobile host to dynamically change the congestion control scheme used for a live connection when it migrates to a heterogeneous network.

(4) *No provision for resource aggregation:* When a multi-homed mobile host handoffs from one access network to another, it may still be within reach of the previous access network. If the mobile host wants to leverage the existence of both networks and simultaneously use both wireless interfaces, no existing approaches that address host mobility provide this solution.

Consequently, either further changes to the infrastructure – that are undesirable given the remarkable but justifiable stubbornness to changes the Internet infrastructure exhibits – are necessitated, or new end-to-end approaches are required. In this paper, we propose a multi-state transport layer solution called pTCP (parallel TCP) that not only handles the problem of interface changes due to host mobility without any infrastructure support, but addresses the aforementioned problems and ameliorates the performance experienced by a mobile host as it moves across heterogeneous wireless networks. Unlike other end-to-end approaches proposed to address host mobility [14], pTCP provides the same reliable and sequenced delivery semantics as TCP, which is used by most applications in the Internet [15]. The key features of pTCP include: (i) *An end-to-end approach for host mobility:* pTCP is a multi-state transport protocol that recognizes end-point address changes during the progress of a connection. (ii) *Provision for seamless handoffs:* Through the dynamic creation and deletion of transport-layer states, pTCP provides a clean solution for leveraging link-layer soft handoffs. (iii) *Support for multiple congestion control schemes:* pTCP decouples congestion control that is performed on a per-path basis from the reliability mechanism performed for the entire connection. It thus allows the use of congestion control schemes that are tailored to the network (or interface) used, by creating appropriate states when handoffs occur. (iv) *A flexible framework for bandwidth aggregation:* pTCP allows a connection to use multiple interfaces at the mobile host. The interfaces are used either in a best-effort fashion (that achieves the sum of individual bandwidths), or based on the performance tradeoffs and the policy supplied by the user.

The rest of the paper is organized as follows: In section 2 we discuss the target network environment, and the goals of the proposed solution. Section 3 describes the key tenets of the pTCP design. Section 4 provides an overview of the pTCP protocol and illustrates how it can be used to help achieve transparent host mobility. Section 5 presents the pTCP protocol in detail along with the state diagram and packet header format. In section 6 we evaluate the performance of pTCP using both simulations and testbed results. Finally, section 7 discusses related work and concludes the paper.

2. Target environment and goals

2.1. Target environment

While we propose a purely end-to-end solution that does not rely on the specifics of the underlying networks, in the following we highlight several characteristics of the target environment:

(1) *Different access technologies:* The mobile host uses different access technologies to connect to heterogeneous wireless networks. These networks usually exhibit vastly different characteristics in terms of bandwidth, delay, jitter, and loss rate. The dynamics of these characteristics also differ from one network to another (e.g., in terms of the amplitude and frequency of fluctuations). Note that such mismatches can result from the heterogeneity in the wireless access link or in the associated core network.

(2) *Different administrative domains:* The network addresses of the wireless interfaces that a mobile host uses are typically assigned and managed by different network operators. While several interworking architectures aim to orchestrate heterogeneous networks that belong to different administrative domains, in this paper we do not assume any coordination between these networks. Note it is possible that a mobile host is assigned network addresses belonging to different administrative domains when it roams between networks accessible using the same wireless access technology. (For example, consider a mobile host that moves from one IEEE 802.11 WLAN to another having a different ESSID [16].)

(3) *Different network models:* While existing wireless networks are primarily based on the cellular network model, several research endeavors have focused on proposing alternate network models that make use of the peer-to-peer relaying capability of the mobile hosts [17–20]. Different network models may provide different qualities of services to mobile hosts due to the performance tradeoffs involved [21]. In this paper we do not assume any particular network model used, as well as the behaviors of different network entities including the assistance from the base stations and the relaying capability of the peer hosts.

2.2. Goals

We aim to provide a comprehensive transport layer solution for a multi-homed mobile host when it moves across heterogeneous wireless networks. The followings are the goals of the proposed solution:

(1) *An end-to-end approach for host mobility:* The proposed solution uses a purely end-to-end approach that involves only the end hosts, without the need for a third party and the support from the underlying networks. The network layer protocol (IP) does not need to be changed to support host mobility. The application also does not need to be changed, except for specifying socket options to use the added functionalities supported by the proposed transport layer protocol. The challenges to achieving host mobility without network support stem from the fact that a connection (socket) is uniquely identified by the (source address, source port, destination address, destination port) connection 4-tuple. When the mobile host migrates to a different network and is assigned a different address, packets carrying the old address cannot be routed to the target host, whereas packets carrying the new address will not be recognized (and hence dropped) by existing sockets at both ends.

(2) *Provision for seamless handoffs:* A salient feature for supporting mobility at the end hosts is the ability to choose the right mode of mobility support depending on the application requirement [9]. In this paper we focus on transparent mobility support as it poses the greatest challenges to host mobility. Whenever the link layer allows, the proposed approach aims to achieve seamless connectivity as the mobile host moves across heterogeneous wireless networks. In particular, since TCP connections account for over 90% of the traffic in the Internet [15], the proposed solution targets applications requiring reliable and sequenced data delivery that TCP provides. Supporting transparent mobility in a TCP connection is non-trivial since TCP reacts adversely to connection stalls, packet reordering, losses, and duplicates that result from the handoffs. An end-to-end approach must be able to address these anomalies without relying on the support from the network.

(3) *Support for multiple congestion control schemes:* A peculiarity of TCP's congestion control mechanism is its assumption on the characteristics of the underlying network. While TCP is well-tuned to the wired network, it does not achieve optimal performance in the wireless environment. Many approaches have hence been proposed to improve the performance of TCP in wireless environments. However, these solutions are tailored (and limited) to specific network scenarios they target. For example, while *snoop* [22] has been proposed primarily for WLANs, it is inappropriate in WWANs due to its key assumption that the wireless link delay is insignificant when compared to the end-to-end delay. WTCP, proposed in [13] for WWANs, however will stand inappropriate in WLANs due to its reliance on inter-packet separation as the key congestion metric. Similarly, approaches such as [12,23] are designed for satellite networks with very large bandwidth-delay products (BDPs). Until a unified congestion control scheme is conceived, the mobile host will need to use network specific congestion control schemes for achieving optimal performance. While this is not an issue when handoffs are performed within a homogeneous wireless network, it is critical when the mobile host migrates across heterogeneous wireless networks.

(4) *A flexible framework for bandwidth aggregation:* While conventional approaches proposed to handle host mobility have thus far considered handoffs as transient behaviors by tearing down the original wireless link as soon as the new link is established, this is not the only option for handoffs across heterogeneous wireless networks. It is possible that the coverage areas of heterogeneous wireless networks overlap, and hence the old wireless interface remains active after the handoff is complete (e.g., consider WLAN hot spots inside a WWAN macrocell). In such a scenario, it will be advantageous for a bandwidth-craving mobile host to use multiple active interfaces simultaneously and enjoy the aggregate bandwidth. Note that bandwidth aggregation is not limited to achieving the sum of data rates available through individual interfaces. It can be used to leverage the performance tradeoffs between different wireless networks. For example, consider a mobile host with a certain bandwidth requirement equipped with both the Wi-Fi and 3G interfaces. While the mobile host can achieve a much higher data rate using the Wi-Fi interface, interferences and data rate fluctuations thereof are more pronounced compared to using the 3G interface. Hence the mobile host may decide to use the Wi-Fi interface exclusively as long as it delivers the target data rate, but incorporate the 3G interface when the achieved data rate falls short. Another example involves using multi-hop wireless networks. While a multi-hop wireless network using peer-to-peer relaying on average provides a higher network capacity than that provided by a single-hop cellular network, throughputs achieved by individual mobile hosts are sensitive to performance degradation due to host mobility (route failures and/or network partitions) and traffic locality [19,21]. A mobile host with both wireless interfaces, one for multi-hop access and one for single-hop access, thus can *opportunistically* use the two interfaces such that it can enjoy the throughput gain provided by the peer-to-peer network without suffering from degraded throughput lower than that provided by the cellular network. The proposed solution aims to provide a flexible framework for achieving various types of bandwidth aggregation.

3. pTCP design

We now present the key design elements of pTCP that allow it to achieve the goals of transparent host mobility across heterogeneous wireless networks as we discussed in section 2.2.

3.1. Dynamic state management

pTCP is a multi-state transport protocol that creates and maintains one TCP state (TCB) for each network interface used

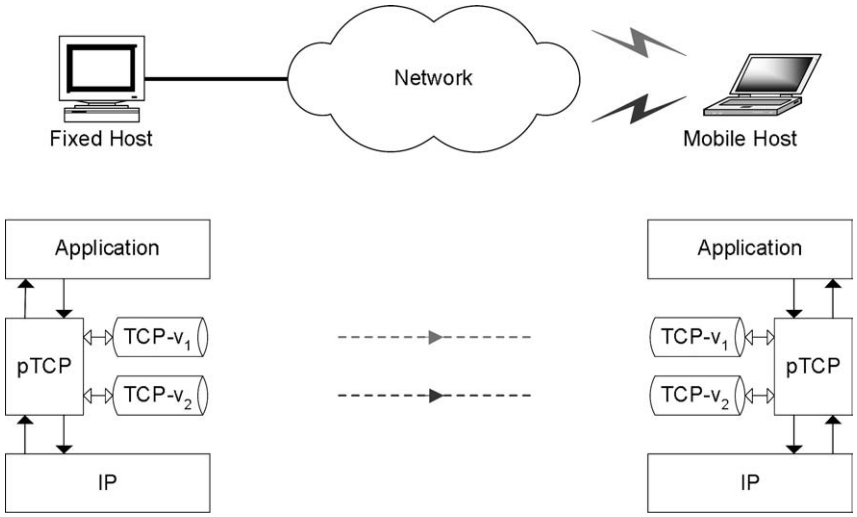


Figure 1. Multiple states in a pTCP connection.

by the application. As we show in figure 1, a TCP-v (*TCP-virtual*) pipe is created for each active interface used in a pTCP connection, to manage the per-path TCP state including the pair of IP addresses, TCP ports, and congestion control parameters. While each TCP-v pipe (and the corresponding state) is addressed using the conventional connection 4-tuple, a pTCP socket can be addressed through any of its component TCP-v pipe. All pipes in a pTCP connection are hashed to the same pTCP socket. pTCP dynamically adds or deletes states in a connection depending on the connectivity (and the change in addresses) between the end hosts. Therefore, although each state can only be associated with one pair of network addresses specified when the state is created, pTCP allows the addresses of the end points in a connection to be changed dynamically. Moreover, as a multi-state transport protocol, pTCP allows multiple pipes to co-exist in a connection, and hence soft handoffs of the transport layer states are possible.

3.2. Decoupling of functionalities

pTCP decouples the transport layer functionalities associated with per-path characteristics from those pertaining to the aggregate connection. The TCP-v opened for each path handles the per-path state, while the pTCP engine (which for simplicity we also refer to as pTCP) handles the aggregate connection. For example, congestion control, the mechanism that estimates the available bandwidth along a path, is a per-path functionality and is handled by TCP-v. On the other hand, the application interacts with the transport layer through the socket buffer, and hence the buffer management (including data resequencing) is handled by the pTCP engine.

The design of TCP-v is the key to decoupling of functionalities. TCP-v is a slightly modified version of TCP that handles only virtual packets, instead of the actual data segments.[1] We call a skeletal packet with only the TCP packet header (sans the data) as a "virtual" packet. Since pTCP controls the send and receive buffers, TCP-v does not need to process the application data. However, TCP-v needs the TCP packet header to perform the regular congestion control as TCP does. All virtual packets generated by individual TCP-v pipes are sent to pTCP to combine with the application data before transmission (we refer to this process as *binding*). On the other hand, all incoming (actual) packets received by pTCP will be stripped off the payload before given to the target TCP-v pipe. As we show in figure 1, the process is possible since the pTCP engine is the only component that interacts with the network layer (IP). An advantage resulting from the decoupling of functionalities is that virtual packets generated by a TCP-v pipe with the same sequence number can be *bound* to different application data, and hence a retransmission at individual TCP-v pipes does not need to be a retransmission of the application data.

3.3. Well-defined interface

The decoupling of functionalities allows network specific congestion control schemes to be used for individual TCP-v pipes without affecting the functionalities performed by the pTCP engine. pTCP does not mandate the specific mechanisms used by each TCP-v pipe to perform congestion control. Any congestion control scheme proposed for various wireless environments can potentially be "plugin" to pTCP for enhancing the performance of pTCP in heterogeneous wireless networks. Toward this end, pTCP provides a well-defined interface with TCP-v that does not depend on the details of the congestion control mechanism used. The interface functions between pTCP and TCP-v include opening and closing a TCP-v pipe, sending and receiving virtual packets, and controlling the amount of data sent through each pipe.

3.4. Effective bandwidth aggregation

While the multi-state design of pTCP allows it to leverage soft handoffs when the mobile host migrates from one access network to another, it also introduces problems of its own.

[1] We use the terms *packet* and *segment* interchangeably in this paper.

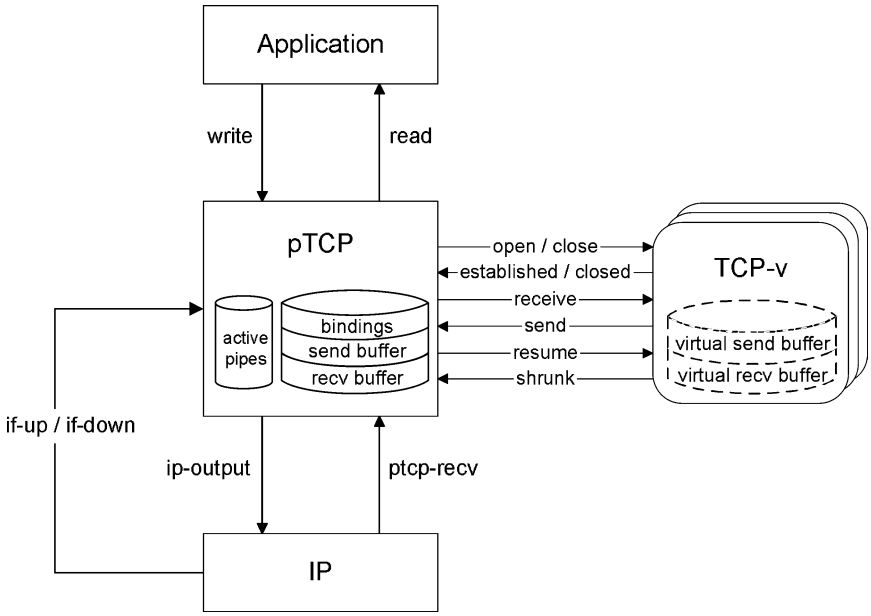


Figure 2. pTCP architecture and key data structures.

Specifically, since pTCP aims to provide the reliable and in-sequence semantics that TCP provides, head-of-line blocking due to mismatches in the capacities of individual TCP-v pipes to deliver data can significantly impact the throughput achieved in the aggregate connection. When multiple pipes of different bandwidths or round-trip times share the same send and resequencing buffers to collectively deliver one single stream of data, packets held up in the slower pipes prevent packets arrived through the faster pipes from being consumed by the application (waiting for the head-of-line "hole" to be filled), thus potentially causing buffer overflow and connection stalls after flow control takes effect [24]. In the target environment, the fact that multiple pipes belong to heterogeneous wireless networks and exhibit very different characteristics in terms of bandwidth fluctuations and blackouts further aggravates the problem.

pTCP uses the following mechanisms to avoid any performance degradation due to head-of-line blocking [24]:

(1) *Delayed binding:* In pTCP, data will be *bound* to a TCP-v pipe only when the pipe is ready to transmit (determined by its congestion control mechanism). pTCP does not allow data to be queued up inside each TCP-v pipe.

(2) *Dynamic reassignment:* If a TCP-v pipe reports losses or bandwidth fluctuations, pTCP immediately *unbinds* the data that is lost or overflows, such that other TCP-v pipes ready for transmission can timely deliver the concerned data.

(3) *Redundant striping:* A data segment can be carried by two TCP-v pipes to minimize the impact of delayed loss recovery. Specifically, the data carried by the first virtual packet of a TCP-v pipe that recently suffered from blackouts will be bound to another TCP-v pipe, such that the application can continue receiving data while the concerned TCP-v pipe probes for the duration of blackouts.

4. pTCP overview

In this section, we provide an overview of the pTCP protocol and illustrate how it can be used to achieve transparent host mobility.

4.1. Architecture overview

Figure 2 provides an overview of the pTCP architecture and key data structures. pTCP as a transport layer protocol interacts with the application and IP, and acts as a wrapper around TCP-v. When an application opens a pTCP socket, by default one TCP-v pipe corresponding to the current network interface in use is created. For each additional interface that becomes active during the lifetime of the connection (e.g., when the mobile host moves to within the coverage area of another network), pTCP creates one more TCP-v pipe. For each interface that becomes inactive, pTCP closes the corresponding TCP-v pipe and removes it from the aggregate connection. The number of TCP-v pipes that co-exist in a pTCP connection thus depends on the number of active network interfaces used by the application.

After opening a pTCP socket, the application writes data to the socket send buffer using the *write*() interface. pTCP uses the *open*() interface to initiate connection setup in each TCP-v pipe that it created. A TCP-v pipe returns with the *established*() call once established. It then builds a virtual packet (with only the TCP header) based on its state variables such as the congestion window size and initial sequence number, and gives the packet to pTCP using the *send*() interface. Upon receiving a virtual packet from a TCP-v pipe, pTCP finds the next data segment to transmit in `send_buffer`, appends the data to the virtual packet, maintains the binding between the application data and the virtual packet in the `bindings` data structure, inserts its own header, and sends the packet to the IP layer using *ip-output*(). The TCP-v pipe

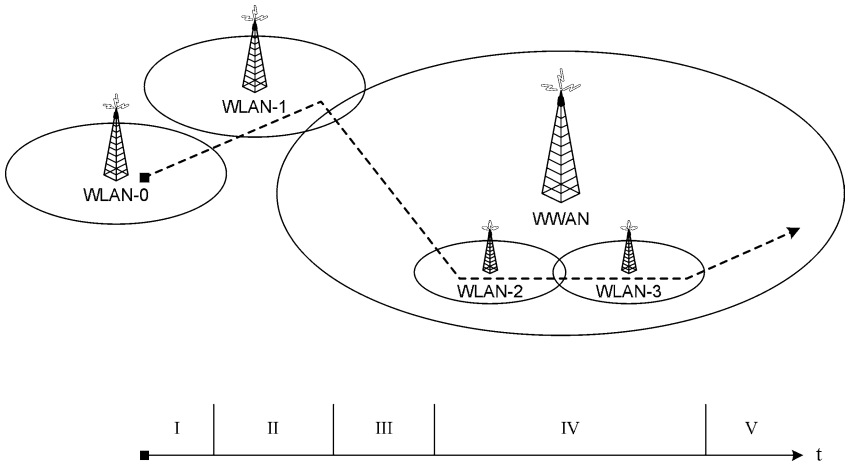


Figure 3. An illustration of mobility trajectory.

continues to issue *send*() calls until there is no more space left in the congestion window, or pTCP responds with a `FREEZE` return value. pTCP freezes TCP-v pipes when there is no unbound data left in the buffer, or when it decides to explicitly control the amount of data given to individual TCP-v pipes (e.g., for performing policy-based bandwidth aggregation discussed in section 2.2). pTCP keeps track of the "frozen" pipes by placing them in the `active_pipes` data structure.

When pTCP receives an incoming data packet via the *ptcp-recv*() interface, it strips both the pTCP header and the data, enqueues the data in the `recv_buffer`, finds the target TCP-v pipe using the connection 4-tuple, and provides the concerned TCP-v with the virtual packet using the *receive*() interface. The TCP-v pipe processes the virtual packet, updates its state variables, and generates cumulative or selective *ACKs.* In-sequence data enqueued in `recv_buffer` can be delivered to the application via the *read*() interface. When pTCP receives an *ACK* from IP, it strips the pTCP header and hands over the packet to the target TCP-v pipe as usual. The TCP-v pipe processes the *ACK* and updates its state variables including the virtual send buffer. The virtual send buffer is used by TCP-v to facilitate loss detection and loss recovery. TCP-v uses the *shrunk*() interface to notify pTCP of any change in the size of its pipe (e.g., change in the congestion window). Upon receiving the *shrunk*() call, pTCP performs dynamic reassignment as discussed in section 3.4. The pTCP header carries cumulative pTCP-level *ACK* information that pTCP can use to purge its send buffer. If any space in the send buffer opens up, pTCP uses the *resume*() interface to "de-freeze" the pipes in `active_pipes`, making them start requesting for transmissions as before.

Finally, pTCP uses the *close*() interface to tear down a TCP-v pipe. Once the TCP-v pipe returns with the *closed*() call, pTCP removes it from the socket. Note that pTCP binds application data to a TCP-v pipe only when the concerned pipe requests for transmissions. Hence adding one more pipe to the aggregate connection has the effect of draining the pTCP send buffer at a faster rate, and deleting a pipe implies a slower rate. The dynamic addition or deletion of TCP-v pipes does not influence the functionalities that pTCP performs including reliability.

4.2. Protocol illustration

We use the scenario shown in figure 3 to illustrate how pTCP can be used to achieve transparent host mobility for a multihomed mobile host with both the WWAN (e.g., 3G) and WLAN (e.g., Wi-Fi) wireless interfaces.[2] The mobile host starts from the WLAN-0 cell and uses its WLAN interface to initiate, say, file download from a remote host in the Internet. While the file download is in progress, the mobile host migrates from one access network (cell) to another according to the mobility trajectory shown in figure 3. We partition the period of mobility into 5 phases and explain the behavior of the mobile host and pTCP in each phase:

(I) The mobile host is in the coverage area of WLAN-0, and is assigned (e.g., via DHCP) an IP address IP_0 for use with the WLAN interface. The application opens a pTCP socket for file download from the remote server with IP address IP_S. pTCP creates the first TCP-v pipe (WLAN-0 pipe) using addresses IP_0 and IP_S. Note that pTCP with only one pipe behaves the same as TCP.

(II) The mobile host moves out of the coverage area of WLAN-0, before which pTCP closes the WLAN-0 pipe using the indication from the lower layer. Shortly the mobile host moves into the coverage area of WLAN-1 that belongs to a different operator, so it initiates registration and is assigned a new IP address IP_1 after the registration is complete. pTCP creates the WLAN-1 pipe using IP_1 and IP_S, and resumes the file download after the WLAN-1 pipe is established with the peer. Note that the application data henceforth is delivered through the WLAN-1 pipe instead of the WLAN-0 pipe. However, the application is unaware of such change due to the dynamic binding performed by pTCP.

[2] Although we use the WWAN and WLAN for illustration due to their popularity, the scenario can be easily extended to other heterogeneous wireless technologies the mobile host has access to.

(III) The mobile host handoffs to WWAN and is assigned an IP address IP_2 for use with the WWAN interface after some registration delay. pTCP then creates the WWAN pipe using the new address. Note that the WLAN-1 pipe remains open throughout the connection establishment process of the WWAN pipe, and hence the application data can still be delivered without any interruption through the WLAN-1 pipe. pTCP closes the WLAN-1 pipe after the WWAN pipe is established. Thereafter, unbound data is assigned to the WWAN pipe. No connection stalls or packet losses occur when pTCP "migrates" from one pipe to another.

(IV) The mobile host handoffs to WLAN-2 and is assigned a new IP address IP_3 for use with the WLAN interface. The mobile host chooses to use bandwidth aggregation during this phase. Hence pTCP creates the WLAN-2 pipe as usual, but does not close the WWAN pipe even after the WLAN-2 pipe is established. The effective striping techniques used by pTCP allow the mobile host to enjoy the aggregate bandwidth of the WWAN and WLAN-2 pipe, despite the bandwidth mismatches and fluctuations. Later, the mobile host undergoes a link-layer handoff to WLAN-3 (the IP address is not changed) that does not impact pTCP's behavior.

(V) Finally, the mobile host leaves the coverage area of WLAN-3. pTCP closes the WLAN-2 pipe and removes it from the aggregate connection after successful teardown. The WWAN pipe keeps delivering the application data without any interruption. The application experiences a change in the data transfer rate due to the deletion of the WLAN-2 pipe. However, it is unaware of the interactions between pTCP and individual TCP-v pipes.

We revisit this scenario in section 6 where we present simulation results showing that pTCP can indeed help the mobile host achieve improved throughput.

5. The pTCP protocol

pTCP provides the same reliable and sequenced delivery semantics as TCP. The transport layer functionalities associated with the aggregate connection are handled by the pTCP engine, while those that pertain to the per-path characteristics are handled by TCP-v. The pTCP engine is responsible for flow control, socket buffer management, and bandwidth aggregation. On the other hand, TCP-v is responsible for congestion control, virtual buffer management, and TCP header generation. In the rest of the section, we describe the different components of the pTCP protocol including interfaces with TCP-v, header format, connection management, congestion and flow control, and reliability.

5.1. TCP-v interface

As we showed in figure 2, pTCP uses the following eight functions to interact with TCP-v: *open*(), *close*(), *established*(), *closed*(), *receive*(), *send*(), *resume*(), and *shrunk*(). pTCP uses the *open*() and *close*() calls as inputs to the TCP-v state machine for opening and closing a TCP-v pipe, respectively. TCP-v uses the *established*() and *closed*() interfaces to inform pTCP when its state machine reaches the ESTABLISHED and CLOSED states, respectively [25]. The *send*() interface is used by TCP-v to send packet headers to pTCP which will then bind the virtual packets to real data. The *receive*() interface on the other hand is used by pTCP to deliver virtual packets to TCP-v. pTCP uses *resume*() to inform TCP-v that additional unbound data is available. TCP-v, upon receiving the call, attempts to send as much data as possible until it gets a `FREEZE` return value on its *send*() call and freezes. Finally, TCP-v uses the *shrunk*() interface to inform pTCP of any change in the available bandwidth (congestion window) of the pipe, such that pTCP can unbind data that overflows and perform reassignment to avoid head-of-line blocking.

5.2. Header format

Figure 4 presents the header format for the pTCP protocol. Note that the header is in addition to the regular TCP header that will be generated by TCP-v. The pTCP header consists of the following four fields: (i) source connection identifier (*pSRC*), (ii) destination connection identifier (*pDST*), (iii) pTCP sequence number (*pSEQ*), and (iv) pTCP acknowledgement number (*pACK*). The connection identifiers are used to uniquely identify the aggregate pTCP connection at both ends. The *pSEQ* field is used for the sequence number at the aggregate connection level and is independent of the sequence number (*SEQ*) field in the TCP header. The *pACK* field is used for the pTCP cumulative acknowledgement similar to the TCP acknowledgement (*ACK*) field. Because pTCP is responsible for performing flow control (given that it controls the buffer), it requires a field for window advertisement as in TCP. However, since TCP-v does not need to perform flow control (it merely maintains virtual buffers), pTCP reuses and overrides the TCP window advertisement field for performing flow control. The reuse does not interfere with the progress of individual TCP-v pipes due to the fact that the pTCP advertised window will always be greater than the actual window of any pipe (we elaborate on this in section 5.4).

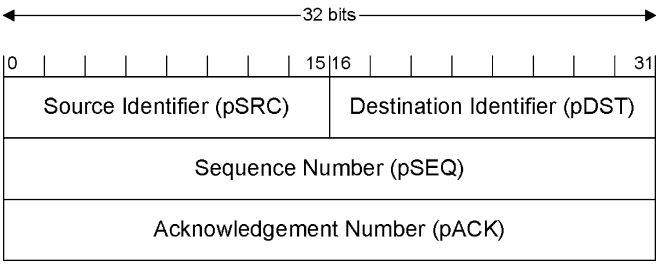


Figure 4. pTCP header format.

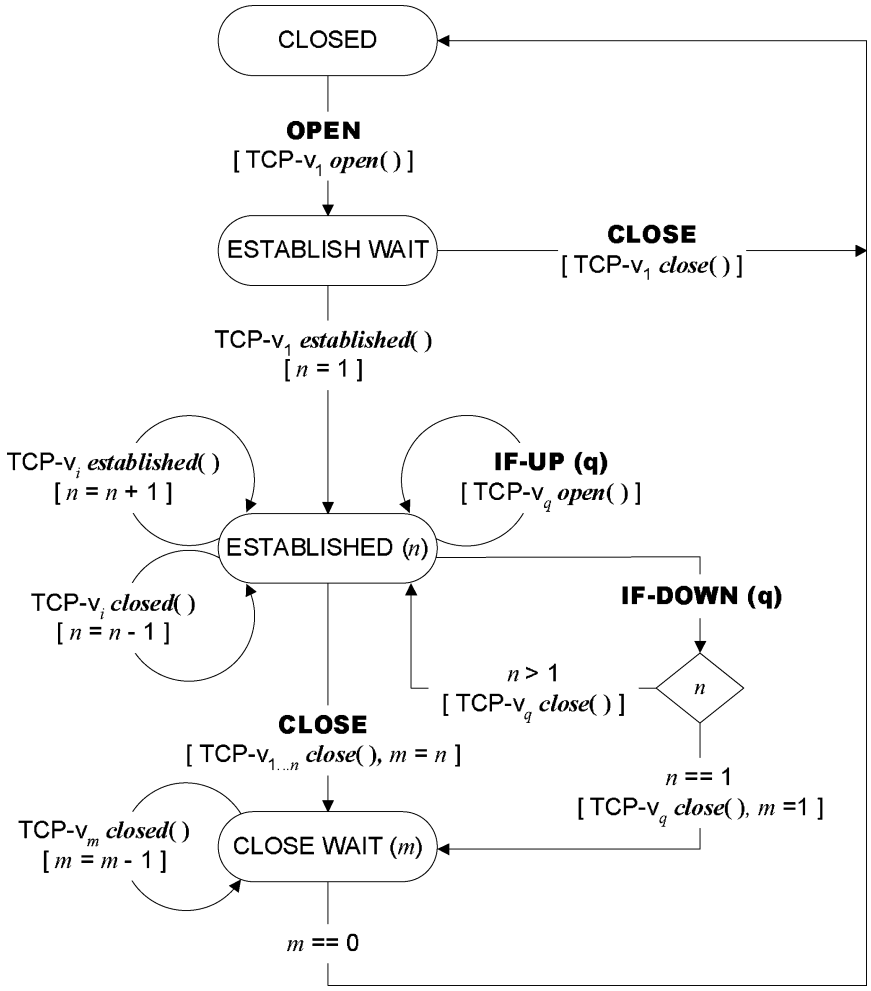


Figure 5. pTCP state machine.

5.3. Connection management

We use the pTCP state machine presented in figure 5 for discussing the connection setup and teardown processes. Note that the state machine of TCP-v is the same as that of default TCP, and the interface between pTCP and TCP-v is presented in section 5.1.

- *Establishment:*
  When an active open is issued by the client application, a pTCP socket with the key data structures introduced in section 4.1 is created. After the pTCP socket is created, pTCP creates one TCP-v pipe and issues the *open*() call to it. When the pTCP at the server end receives the *passive* open, it creates the first TCP-v pipe, issues the passive open to it, and in the process takes it to the SYN_RCVD state [25]. When the client TCP-v receives the $SYN + ACK$, it enters the ESTABLISHED state after sending back an *ACK* to the server. The *established*() callback from the first pipe brings pTCP to the ESTABLISHED (1) state, and allows pTCP to start accepting data from the application. When the first TCP-v at the server receives the *ACK*, it also enters the ESTABLISHED state and can participate in the data exchange with the client. The pTCP at the server end also enters the ESTABLISHED (1) state when the first pipe is established.
  For each interface that becomes active (indicated by the *if-up*() callback from the lower layer) during the connection, pTCP creates one more TCP-v pipe and issues the *open*() call to it. The newly created pipe then initiates the regular connection establishment handshakes with the peer. Note that the *SYN* of the new pipe can be identified and processed by the peer pTCP since it carries the correct pTCP identifiers (*pSRC* and *pDST*) used in the current pTCP connection. While pTCP can still send and receive data using existing TCP-v pipes, the *established*() callback from the newly created pipe causes pTCP to move down the state machine to state ESTABLISHED (n), where n equals to the number of established TCP-v pipes in the aggregation connection.
- *Termination:*
  The teardown of a pTCP connection is similar to the connection establishment. For each interface that becomes inactive (indicated by the *if-down*() callback), pTCP uses the *close*() interface to make the concerned TCP-v pipe close. Each pipe closes using TCP's regular closing handshake. When a TCP-v pipe enters the CLOSED state

in its state machine, it invokes the *closed*() callback to pTCP. pTCP keeps track of the number of closed TCP-v pipes and moves to the appropriate ESTABLISHED (*n*) state. When an application closes the connection, pTCP issues *close*() to all pipes. Upon successful shutdown in all TCP-v pipes, pTCP enters the CLOSED state and confirms the close to the application.

5.4. Congestion control and flow control

pTCP by itself does not perform any congestion control. The individual TCP-v pipes are solely responsible for controlling the amount of data transferred through each pipe. Since each TCP-v pipe has the whole control over the congestion control mechanism used for each path, it is possible to have different congestion control mechanisms (on a per-pipe basis) coexist in a pTCP connection. We assume that the choice as to which congestion control mechanism to be used for each wireless interface is an external decision, and is provided to pTCP through the system-wide configuration or socket option.

On the other hand, flow control in the pTCP protocol is performed by the pTCP engine that has control over the socket buffer. Since the buffer space is shared by all pipes, and the amount of space used by individual pipes is based on their respective BDPs, pTCP allows an efficient use of the buffer across multiple pipes similar to the mechanism proposed in [26]. Every packet sent out by pTCP carries the available space in the pTCP receive buffer, irrespective of which pipe the virtual packet belongs to. The pTCP sender keeps track of the number of outstanding bytes for the connection, and ensures that the receive buffer never overflows. Although all TCP-v pipes in a pTCP connection see the same available buffer space and hence can contend simultaneously for that space (provided there is space in their congestion windows), no excess data will be transmitted since pTCP has control over all data transmissions. pTCP uses the `FREEZE` return value in response to the *send*() call to freeze a TCP-v pipe if there is no more unbound data to transmit in the send buffer. Note that flow control allows pTCP not only to avoid buffer overflow, but also to achieve flexible bandwidth aggregation. Based on the "policy" supplied by the user or the application, pTCP can freeze a TCP-v pipe even when there is space in the send buffer, to explicitly control the amount of data transmitted through each pipe. We present a simple algorithm in section 6 to show the effectiveness of such a policy-based flow control.

5.5. Reliability

As we discussed in section 4.1, pTCP maintains the bindings between the actual data segments and the TCP-v virtual packets. Once the application data is bound to a particular TCP-v pipe, it is the concerned TCP-v pipe's responsibility to reliably deliver the data to the receiver (using the reliability mechanism in TCP). Therefore, reliable transport of the application data is achieved in pTCP as long as every data segment in pTCP's send buffer is bound to a virtual packet in one of its TCP-v pipe's virtual send buffer. However, note that the binding between the actual data segment and the TCP-v virtual packet can be altered when pTCP performs dynamic reassignment or redundant striping (refer to section 3.4). We now discuss how pTCP can still ensure reliability under these two special conditions:

(1) *Dynamic reassignment:* Whenever pTCP performs dynamic reassignment of a data segment from one TCP-v pipe to another, it unbinds the data segment from the old pipe, and reassigns (binds) it to the next available pipe (which can be the same pipe). Thereafter, the new pipe will assume the responsibility of reliably delivering the reassigned data segment to the peer TCP-v. Note that when the old pipe "retransmits" the previously bound virtual packet, it will be reassigned a different data segment to carry.

(2) *Redundant striping:* Redundant striping in pTCP is a special case of the dynamic reassignment where the old pipe still keeps a copy of the data segment. Since the data segment will be bound to a new pipe, its reliable delivery will be guaranteed by the new pipe. However, since the old pipe will also attempt to deliver the same data segment to its peer, there might be duplicates at the receiving pTCP. Such duplicates can be easily detected via the sequence number field (*pSEQ*) in the pTCP packet header.

We have thus far described the key components of the pTCP protocol. In the next section, we present performance of pTCP in achieving transparent mobility across heterogeneous wireless networks.

6. Performance evaluation

We evaluate the performance of pTCP through simulation and testbed results. In the following, we first present the simulation model and simulation results, and then show the experiment results using a real-life testbed.

6.1. Simulation model

We use the *ns-2* [27] network simulator and the network topology shown in figure 6 for simulations. We construct the network topology based on the scenario discussed in section 4.2, and hence $L0$ to $L3$ correspond to the four WLAN cells, and W corresponds to the the WWAN cell in figure 3. We use four gateways ($G0$ to $G3$) to connect the WWAN base station (W) and WLAN access points ($L0$ to $L3$) to the backbone network. The link delays are properly set such that the base round-trip time (not counting the queuing delay) from the mobile host (*MH*) to the fixed host (*FH*) is 200 ms using the WWAN interface with 2 Mbps data rate. The round-trip time using the WLAN interface ranges from 120 ms ($L0$) to 60 ms ($L2$ and $L3$), while the data rate ranges from 5 Mbps ($L0$) to 10 Mbps ($L1$ to $L3$).

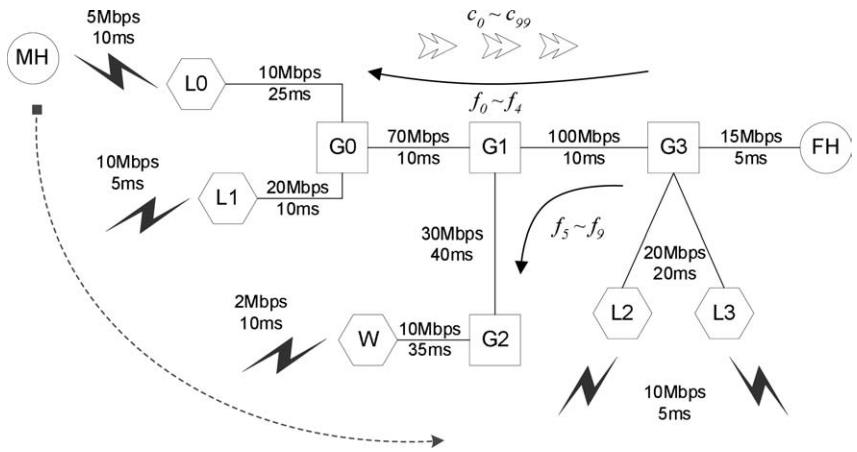


Figure 6. Network topology.

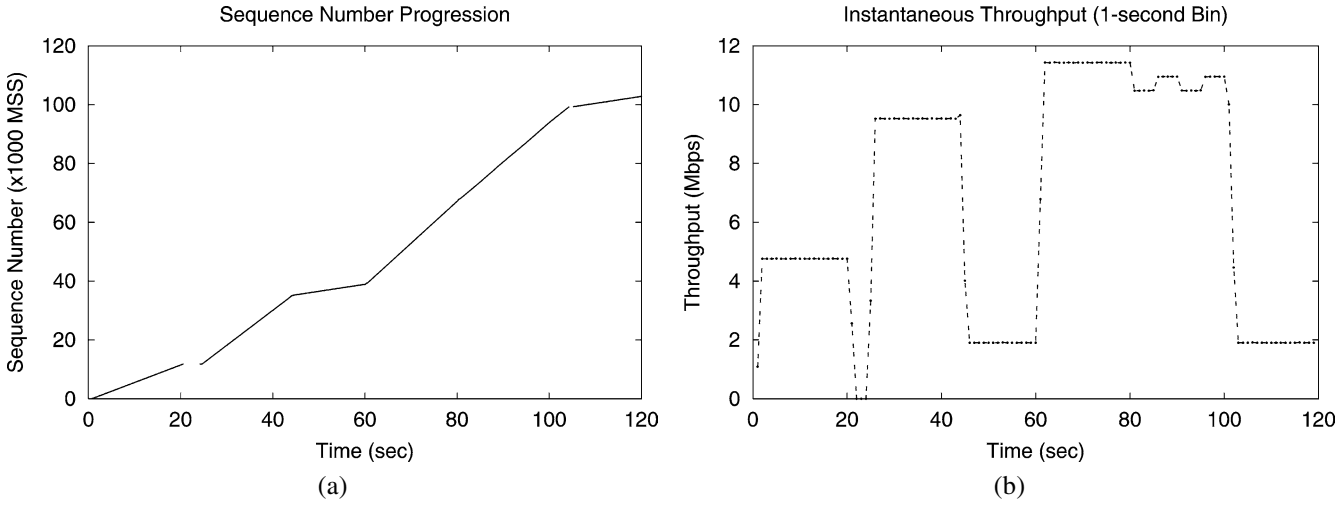


Figure 7. Performance of pTCP: an overview. (a) Sequence number progression. (b) Instantaneous throughput.

The mobile host *MH* opens a pTCP connection for file download from a backlogged FTP source *FH* in the backbone network. We introduce 10 background TCP traffic in the network: 5 TCP flows (f_0 to f_4) traversing from $G3$ to $G0$ and 5 TCP flows (f_5 to f_9) traversing from $G3$ to $G2$. All TCP flows, including the TCP-v pipes in the pTCP connection, use TCP-SACK with the maximum segment size (MSS) set to 1000 bytes. In addition to the background TCP traffic, we also introduce 100 on-off UDP flows (c_0 to c_{99}) traversing from $G0$ to $G3$ to emulate the flash crowds (e.g., WWW-like traffic) in the Internet. We use the Pareto traffic source for all UDP flows, where the mean burst time is set to 1 s, the mean idle time is set to 2 s, the data rate during burst time is set to 500 Kbps, and the shape parameter is set to 1.5. We use the sequence number plot and instantaneous throughput as metrics to show the performance achieved at the mobile host.

6.2. Simulation results

Figure 7(a) shows the maximum in-sequence segment number received at the mobile host (ready to deliver to the application), and figure 7(b) shows the corresponding instantaneous throughput using a bin size of 1 second. The ticks marked on the x-axis roughly partition the simulation period into the 5 phases that we described in section 4.2, except for phase IV that spans from $t = 60$ to $t = 100$. We show the performance of pTCP from the following two aspects: (i) seamless handoffs: from $t = 0$ to $t = 60$, and from $t = 100$ to $t = 120$, and (ii) bandwidth aggregation: from $t = 60$ to $t = 100$.

6.2.1. Seamless handoffs

The mobile host moves out the coverage area of WLAN-0 at $t = 21$, and shortly moves into the coverage area of WLAN-1 that belongs to a different operator. After some registration delay [28], the mobile host at $t = 24.10$ obtains a new IP address to be used in WLAN-1, and then it creates the corresponding TCP-v pipe. The WLAN-1 pipe is established at $t = 24.22$ following the three-way connection setup process. Since the mobile host left the coverage of WLAN-0 while the registration in WLAN-1 is in progress, there is a hard handoff at the link layer. Although pTCP cannot overcome the connection disruption during the hard handoff, it achieves host mobility by allowing the addresses of the end points to be changed in a connection. pTCP at the fixed host continues to

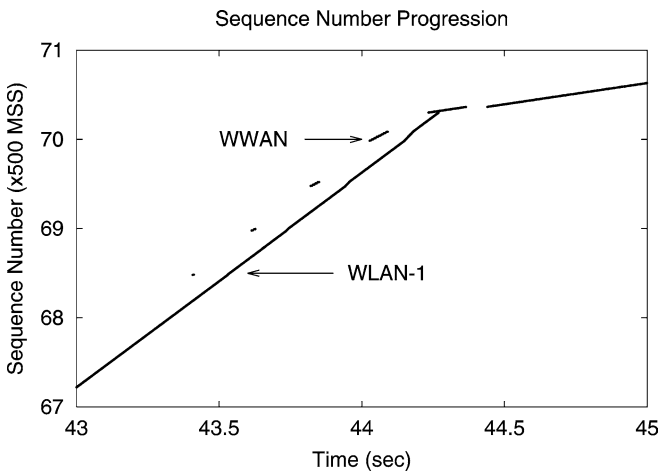


Figure 8. pTCP sequence number progression during soft handoffs.

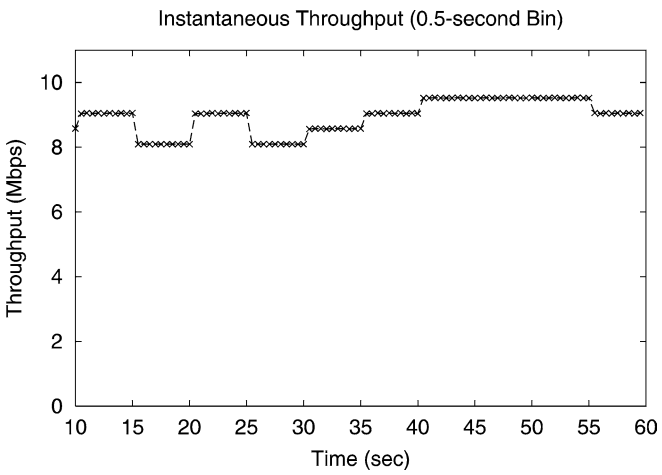


Figure 9. Bandwidth fluctuations in the WLAN-3 cell (sample).

send data from its socket buffer to the mobile host after the WLAN-1 pipe is established.

A more interesting scenario arises when the mobile host at $t = 40$ moves into the coverage area of WWAN, and decides to handoff to the WWAN cell. We assume that the mobile host is assigned a new IP address at $t = 43$ after it properly registers with the WWAN. The mobile host proceeds with creating the WWAN pipe, and then establishes the pipe at $t = 43.30$. While conventional approaches for handling handoffs need to shutdown the old link as soon as the new one is established (to prevent out-of-order delivery or packet duplicates from impacting the performance of TCP), pTCP can take a different approach due to its multi-state design. Since pTCP opens one TCP-v pipe for each interface (wireless link), it can operate over multiple paths without triggering the adverse reaction of the congestion control mechanism used in TCP. Therefore, it is upto pTCP to choose when to close the old pipe (provided that the link layer permits). The old pipe can be retained for sake of bandwidth aggregation, or it can be shutdown after the new pipe is able to steadily deliver the packets (e.g., after it leaves the slow start phase). In this simulation, we assume that pTCP closes the WLAN-1 pipe at $t = 44$, and removes the pipe from the connection at $t = 44.39$ after it returns closed. Figure 8 shows an enlarged plot of the sequence number (pTCP sequence number) received at the mobile host. From the two groups of segments received (the two pipes have different round-trip times) shown in the figure, it is clear that two pipes co-exist during the handoff. While the new pipe (WWAN) crawls in the slow start (as evident from the exponentially increased block of segments received), the WLAN-1 pipe keeps delivering data for the pTCP connection until it is closed. We observe a slight throughput increase during this period from figure 7(b), due to the co-existence of the two pipes.

Thereafter, the mobile host keeps moving inside the WWAN cell, and handoffs to the WLAN-2 cell at $t = 60$. From $t = 60$ to $t = 100$ the mobile host uses both the WWAN and WLAN interfaces simultaneously for bandwidth aggregation, which we will discuss in section 6.2.2. Finally, it leaves the WLAN-3 cell at $t = 100$ and closes the WLAN-2 pipe. The connection is then served exclusively by the WWAN pipe until the end of the simulation.

6.2.2. Bandwidth aggregation

At $t = 60$ the mobile host enters the coverage area of the WLAN-2 cell. Since it is still within coverage of the WWAN cell, it decides to enjoy bandwidth aggregation by simultaneously using the WWAN and WLAN interfaces. Therefore, the mobile host retains the WWAN pipe even after the WLAN-2 pipe is established at $t = 60.19$. It is clear from figure 7(b) that from approximately $t = 60.19$ to $t = 80$, the mobile host enjoys the aggregate bandwidth provided by the WWAN and WLAN interfaces. At $t = 80$ the mobile host moves to the coverage area of WLAN-3 and undergoes a link-layer handoff. WLAN-3 is a relatively crowded cell, and hence the mobile host experiences bandwidth fluctuations using the WLAN interface. While the available bandwidth x of the WLAN-2 pipe (recall that the pipe was established in WLAN-2) varies from 8.5 Mbps to 9.5 Mbps as illustrated by the sample trace shown in figure 9, we observe that the aggregate bandwidth closely follows $x + y$, where y is the bandwidth achieved using the WWAN interface.

We now show a variation of the "best-effort" bandwidth aggregation (that aims to achieve the bandwidth sum) in pTCP that allows the mobile host to avoid such undesirable bandwidth fluctuations. Assume that the mobile host has a target data rate and preferentially uses the WLAN interface for achieving the desired rate (e.g., due to the lower cost in accessing the WLAN compared to the WWAN). It uses the WWAN interface only when the offered data rate through the WLAN interface falls short. Figure 10 shows a simple algorithm that achieves such policy-based bandwidth aggregation. Briefly, pTCP periodically generates transmission tokens (lines 16–19) according to the target data rate. Only the primary pipe (WLAN-2 pipe) can consume the newly generated tokens (*quota*). However, unused tokens (*underflow*) carried to the next period can be consumed by both pipes. pTCP uses such information (in addition to the regular flow control based on buffer availability) to decide (lines 4–15) whether a pipe issuing the *send*() call should send or freeze.

```
     send(pipeid p):
 1     if throttle(p)
 2       return FREEZE
 3     perform binding and send out the packet

     throttle(pipeid p):
 4     if is-primary-pipe(p)
 5       if quota > 0
 6         decrease quota
 7         return 0
 8       else if underflow > 0
 9         decrease underflow
10         return 0
11     else
12       if underflow > 0
13         decrease underflow
14         return 0
15     return 1

     token-timer-expires():
16     underflow ← quota
17     quota ← TOKEN
18     foreach pipe p ∈ active_pipes
19       p→resume()
```

Figure 10. A simple algorithm for policy-based bandwidth aggregation.

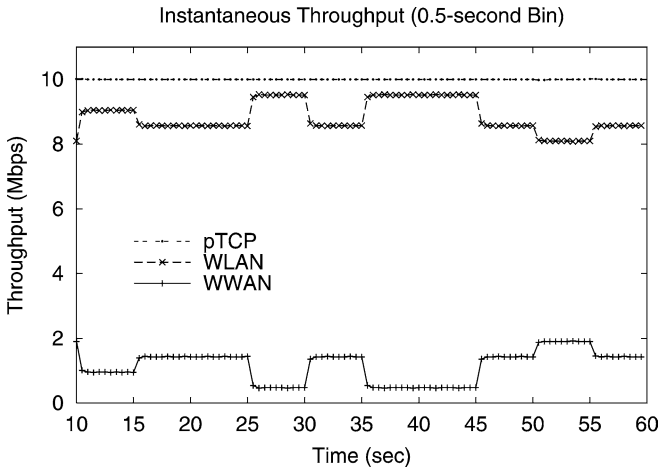


Figure 11. Policy-based bandwidth aggregation in pTCP.

Figure 11 shows the performance of the simple policy-based bandwidth aggregation. The target data rate of the mobile host is 10 Mbps, while the offered bandwidth of the WLAN interface varies from 8.5 Mbps to 9.5 Mbps. It is clear that the aggregated bandwidth enjoyed by the mobile host stays at the desired data rate when the policy-based bandwidth aggregation is used. The data rate achieved through the WWAN interface "fills" the deficient data rate that the WLAN interface fails to provide. Note that although we use the WLAN as an example to show the benefits of such policy-based bandwidth aggregation, any other wireless networks, especially the multi-hop peer-to-peer networks where data rate fluctuations are typical, can also be used.

For lack of space, we do not present in this paper the performance of pTCP to effectively achieve bandwidth aggregation when the component pipes exhibit very different characteristics in terms of large bandwidth differentials, significant bandwidth fluctuations, and blackouts. We refer interested readers to [24] for detailed presentations and discussions.

6.3. Testbed results

In this section, we present results from experiments conducted with the pTCP implementation in a real-life campus network. We use the topology shown in figure 12 for the experiment. The server is a Dell Optiplex GX110 desktop with a Pentium III 733 MHz CPU and 256 MB RAM, while the client is an IBM Thinkpad T-20 laptop with a Pentium III 700 MHz CPU and 128 MB RAM. The server has a 100 Mbps FastEthernet connection to the network, while the client is equipped with two Orinoco 802.11b WLAN cards with a maximum per-link data rate of 11 Mbps. The two WLAN cards are associated with two WLANs with different ESSIDs. The channel used by one WLAN (WLAN-2) experiences noticeable interferences that introduce a higher packet drop rate. Both the server and the client run the Linux operating system. The pTCP protocol is implemented in the kernel (version 2.4.18) of the two hosts. We consider file download from the server to the client. The FTP application opens a pTCP socket by using the *socket*(`PF_INET`, `SOCK_STREAM`, `IPPROTO_PTCP`) function call. We show

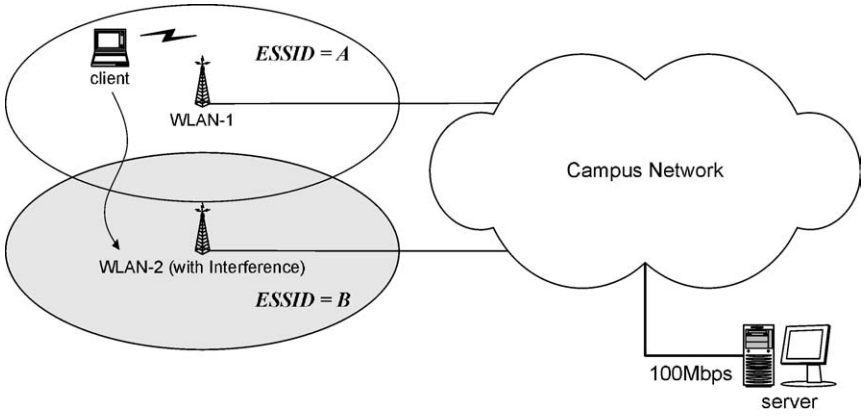


Figure 12. Testbed topology.

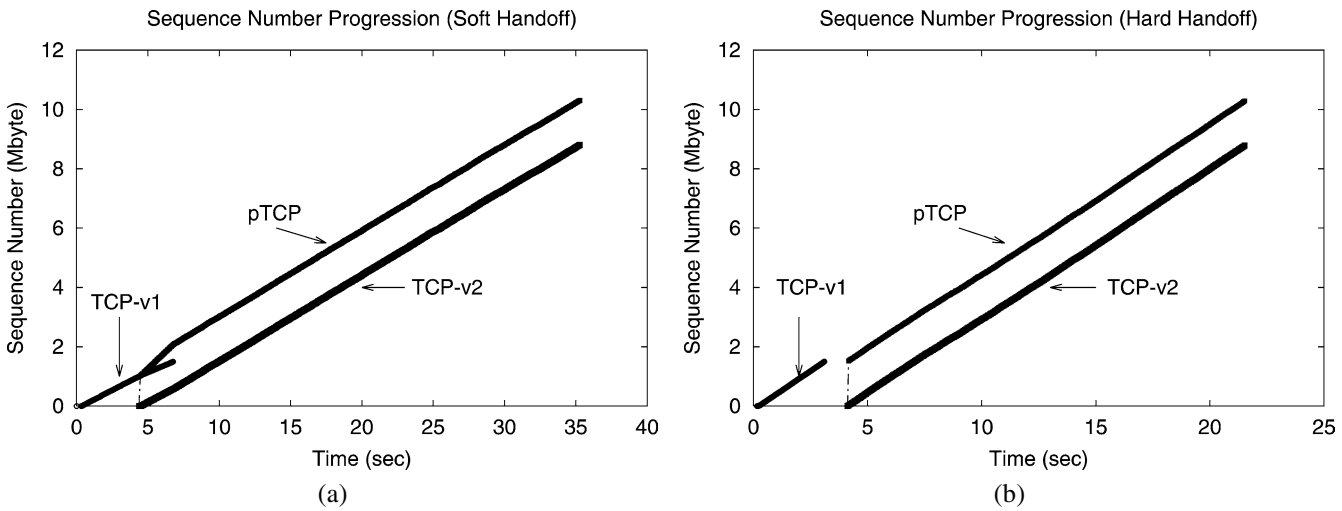


Figure 13. Performance of pTCP during handoffs. (a) Soft handoff. (b) Hard handoff.

the performance of pTCP using the following two scenarios: (i) *Handoffs:* We demonstrate the soft handoff capability of pTCP when the client moves across different access networks, by comparing the result against a hard handoff scenario. (ii) *Bandwidth aggregation using multiple congestion control schemes:* We study the performance of pTCP with respect to its bandwidth aggregation capability, specifically when the component pipes use different congestion control schemes.

6.3.1. Handoffs

Figure 13(a) shows the performance of pTCP when a soft handoff is performed from WLAN-1 to WLAN-2. When the connection is started at $t = 0$, only the interface associated with WLAN-1 is up. At $t = 4.5$, the second interface associated with WLAN-2 becomes active. The first interface is brought down only at $t = 7$, and hence for 2.5 s both interfaces are active. pTCP creates the first pipe TCP-v1 when the connection starts, and then it creates the second pipe TCP-v2 when the second interface comes up (using appropriate feedback from the link layer). pTCP's bandwidth aggregation capability comes into play during the 2.5 s when both interfaces are up, as can be observed from the increased slope in figure 13(a). Due to the soft handoff performed at pTCP, there is no disruption in the service enjoyed by the application. To see the disruption in service to the application when the transport layer performs a hard handoff, we consider another scenario where pTCP creates TCP-v2 only after it closes TCP-v1. As figure 13(b) shows, the service disruption time is at least one round-trip time of the connection.

6.3.2. Multiple congestion control schemes

We present in figure 14 the ability of pTCP to use multiple congestion control schemes for different TCP-v pipes belonging to the same aggregate connection. While packets traversing through the WLAN-2 interface experience a higher drop rate, for results shown in figures 14(a) and (b) we use the default congestion control in TCP for both pipes. It is clear that the second pipe (TCP-v2) experiences more losses and achieves a lower throughput. For results presented in figure 14(c), we modify the congestion control used by TCP-v2 to TCP-ELN. Essentially, using TCP-ELN the sender does not reduce its congestion window for random wireless losses, although it still uses the feedback for triggering a retransmission. It can be seen from the figure that (i) using the "wireless-aware" congestion control scheme improves the performance for TCP-v2, and (ii) irrespective of the specific congestion control schemes used, pTCP always achieves the aggregate performance of the two TCP-v pipes.

7. Related work and conclusions

7.1. Related work

The mushrooming of diverse wireless access technologies in recent years has prompted many research endeavors to address the problem of host mobility in heterogeneous wireless networks [4,5,7,8,29]. For example, in [4] a tightly coupled architecture is proposed to interwork HIPERLAN (WLAN) and UMTS (3G) for achieving seamless handoffs. In this architecture, the WLAN is connected to the 3G core network and is under the administration of the 3G operator such that it appears simply as another RAN (radio access network) to the 3G system. In this way, all traffic, be it from the UMTS access network or from the HIPERLAN access network, traverses through the 3G core network, and hence the session mobility can be performed in the same fashion as in a homogeneous network. In [7], the authors propose a MIRAI network architecture to connect all various access networks to a common core network such that AAA (authentication, authorization, and accounting) and mobility can be managed in a homogeneous fashion. In [8] a communication gateway is proposed to shield the mobile user from the heterogeneity of wireless access technologies such that the mobile user can achieve infrastructure-independent 4G wireless access. Since the network infrastructure is likely to keep diverging in future wireless systems, these approaches are not effective in

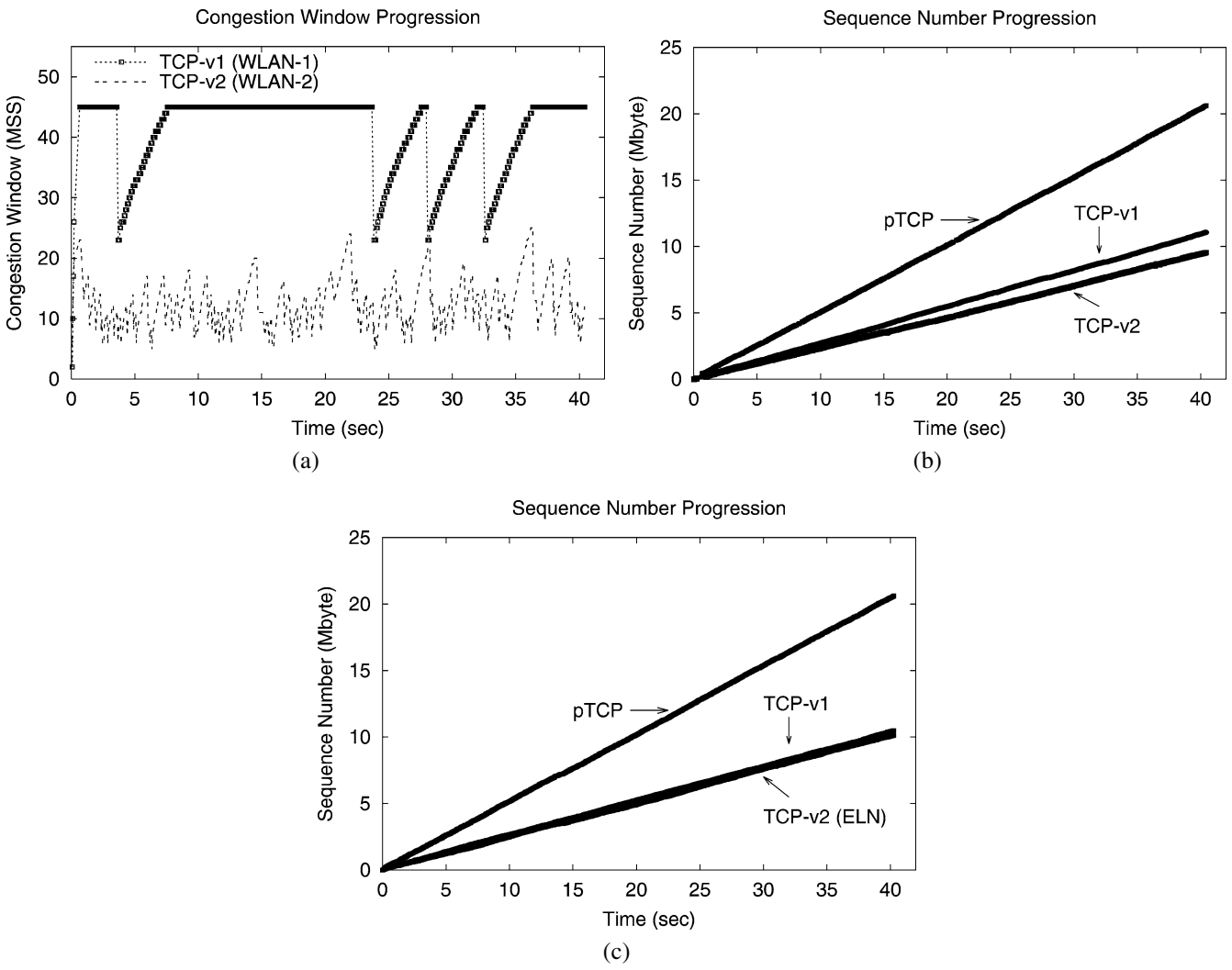


Figure 14. Performance of pTCP for bandwidth aggregation. (a) Individual congestion windows. (b) One congestion control scheme. (c) Two congestion control schemes.

addressing network heterogeneity as any new access technology introduced requires changes to the solutions proposed.

Related work that uses purely end-to-end approaches for achieving host mobility has also been proposed. In [9] the authors propose the TCP migration option such that an existing TCP connection does not get interrupted when the mobile host migrates to a different network and is assigned a different IP address. However, by using only one state in the connection, the TCP migration option in fact performs "hard handoffs" between the old and new states. It thus fails to leverage the advantages of soft handoffs that the link layer supports. In [14] the authors propose a multi-stream protocol called "Mobile SCTP" for achieving transport layer mobility at a multi-homed mobile host without the need to change the IP substrate. While the authors do make a case for achieving host mobility using only end-to-end mechanisms, the scope is different from that of pTCP. Mobile SCTP does not support in-sequence data delivery (across multiple streams) as TCP, and hence it is not applicable for applications that require in-sequence semantics. Moreover, Mobile SCTP does not allow multiple congestion control schemes to be used for different wireless interfaces, and it does not support bandwidth aggregation. Nevertheless, we believe that pTCP can benefit from the development and deployment of Mobile SCTP.

In terms of bandwidth aggregation, while there has been some related work proposed for network striping, they are not proposed in the context of host mobility. Specifically, the majority of the approaches proposed for bandwidth aggregation focus on link layer techniques [30–32]. However, link layer techniques do not perform well in the context of multi-homed mobile hosts, where the multiple interfaces are more likely to belong to different network domains altogether, and exhibit highly dynamic and vastly differing characteristics. In [33], the authors propose a mechanism for data streaming across multiple wireless links on a multi-homed mobile host using network layer techniques. The authors use IP-in-IP encapsulation such that packets from different interfaces are tunneled within the same IP address. However, since only one TCP state is used for the aggregation connection, the authors show that there is a scalability problem as to how differ-

ent the characteristics of component interfaces can be, before the out-of-order delivery kicks in to impact the performance of TCP. Finally, several application layer striping techniques have also been proposed to use multiple TCP connections in parallel for achieving a higher throughput [34–36]. We show in [24] through simulation results the disadvantages of such approaches.

7.2. Conclusions

In this paper, we consider the problem of achieving host mobility for a multi-homed mobile host using purely end-to-end techniques. We propose a transport layer protocol called pTCP that enables the seamless use of heterogeneous access technologies and achieves transparent host mobility. The unique features of the proposed solution include: (i) a purely end-to-end approach to handle host mobility that requires no support from the underlying network infrastructure, (ii) seamless vertical handoffs when migrating from one access network to another, (iii) ability to support different congestion control schemes for connections involving multiple interfaces, and (iv) effective bandwidth aggregation when the mobile host has simultaneous access to multiple networks. We present the design and details of the proposed approach, and evaluate its performance through simulations and real-life field experiments.

References

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

[2] K. Pahlavan, P. Krishnamurthy, A. Hatami, M. Ylianttila, J.-P. Makela, R. Pichna and J. Vallstrom, Handoff in hybrid mobile data networks, IEEE Personal Communications Magazine 7(2) (April 2000) 34–47.

[3] C. Perkins, Mobile IP, IEEE Communications Magazine 40(5) (May 2002) 66–82.

[4] ETSI, BRAN; HIPERLAN/2; Requirements and architecture for internetworking between HIPERLAN/2 and 3rd generation cellular systems, TR 101 957, V1.1.1 (August 2001).

[5] A. Salkintzis, C. Fors and R. Pazhyannur, WLAN-GPRS integration for next-generation mobile data networks, IEEE Wireless Communications Magazine 9(5) (October 2002) 112–124.

[6] M. Buddhikot, G. Chandranmenon, S.-J. Han, Y.-W. Lee, S. Miller and L. Salgarelli, Integration of 802.11 and third-generation wireless data networks, in: *Proceedings of IEEE INFOCOM*, San Francisco, CA (April 2003) pp. 503–512.

[7] G. Wu, M. Mizuno and P. Havinga, MIRAI architecture for heterogeneous network, IEEE Communications Magazine 40(2) (February 2002) 126–134.

[8] W. Kellerer, H.-J. Vogel and K.-E. Steinberg, A communication gateway for infrastructure-independent 4G wireless access, IEEE Communications Magazine 40(3) (March 2002) 126–131.

[9] A. Snoeren and H. Balakrishnan, An end-to-end approach to host mobility, in: *Proceedings of ACM MOBICOM*, Boston, MA (August 2000) pp. 155–166.

[10] G. Montenegro, Reverse tunneling for Mobile IP, IETF RFC 3024 (January 2001).

[11] J. Postel, Transmission control protocol, IETF RFC 793 (September 1981).

[12] T. Henderson and R. Katz, Transport protocols for Internet-compatible satellite networks, IEEE Journal on Selected Areas in Communications (JSAC) 17(2) (February 1999) 345–359.

[13] P. Sinha, N. Venkitaraman, R. Sivakumar and V. Bharghavan, WTCP: A reliable transport protocol for wireless wide-area networks, in: *Proceedings of ACM MOBICOM*, Seattle, WA (August 1999) pp. 231–241.

[14] M. Riegel and M. Tuexen, Mobile SCTP, Internet draft; draft-riegel-tuexen-mobile-sctp-02.txt (February 2003).

[15] CAIDA, Analysis of Internet traffic, `http://www.caida.org/analysis/performance`.

[16] IEEE, Wireless LAN medium access control (MAC) and physical layer (PHY) specifications, ANSI/IEEE Standard 802.11 (August 1999).

[17] Y.-D. Lin and Y.-C. Hsu, Multihop cellular: A new architecture for wireless communications, in: *Proceedings of IEEE INFOCOM*, Tel-Aviv, Israel (March 2000) pp. 1273–1282.

[18] H. Wu, C. Qiao, S. De and O. Tonguz, Integrated cellular and ad hoc relaying systems: iCAR, IEEE Journal on Selected Areas in Communications (JSAC) 19(10) (October 2001) 2105–2115.

[19] H.-Y. Hsieh and R. Sivakumar, On using the ad-hoc network model in cellular packet data networks, in: *Proceedings of ACM MOBIHOC*, Lausanne, Switzerland (June 2002) pp. 36–47.

[20] 3GPP, 3GPP TSG-RAN; Opportunity driven multiple access, TR 25.924, V1.0.0 (December 1999).

[21] H.-Y. Hsieh and R. Sivakumar, Performance comparison of cellular and multi-hop wireless networks: A quantitative study, in: *Proceedings of ACM SIGMETRICS*, Boston, MA (June 2001) pp. 113–122.

[22] H. Balakrishnan, S. Seshan and R. Katz, Improving reliable transport and handoff performance in cellular wireless networks, Wireless Networks (WINET) 1(4) (December 1995) 469–481.

[23] I. Akyildiz, G. Morabito and S. Palazzo, TCP-Peach: A new congestion control scheme for satellite IP networks, IEEE/ACM Transactions on Networking 9(3) (June 2001) 307–321.

[24] H.-Y. Hsieh and R. Sivakumar, A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts, in: *Proceedings of ACM MOBICOM*, Atlanta, GA (September 2002) pp. 83–94.

[25] G.R. Wright and W.R. Stevens, *TCP/IP Illustrated*, Vol. 2 (Addison-Wesley, Reading, MA, 1997).

[26] J. Semke, J. Mahdavi and M. Mathis, Automatic TCP buffer tuning, in: *Proceedings of ACM SIGCOMM*, Vancouver, Canada (September 1998) pp. 315–323.

[27] The Network Simulator, ns-2, `http://www.isi.edu/nsnam/ns`.

[28] H. Yokota, A. Idoue, T. Hasegawa and T. Kato, Link layer assisted mobile IP fast handoff method over wireless LAN networks, in: *Proceedings of ACM MOBICOM*, Atlanta, GA (September 2002) pp. 131–139.

[29] T. Otsu, I. Okajima, N. Umeda and Y. Yamao, Network architecture for mobile communications systems beyond IMT-2000, IEEE Personal Communications Magazine 8(5) (October 2001) 31–37.

[30] K. Sklower, B. Lloyd, G. McGregor, D. Carr and T. Coradetti, The PPP multilink protocol, IETF RFC 1990 (August 1996).

[31] H. Adiseshu, G. Parulkar and G. Varghese, A reliable and scalable striping protocol, in: *Proceedings of ACM SIGCOMM*, Palo Alto, CA (August 1996) pp. 131–141.

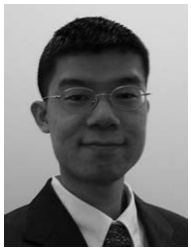
[32] A. Snoeren, Adaptive inverse multiplexing for wide-area wireless networks, in: *Proceedings of IEEE GLOBECOM*, Rio de Janeiro, Brazil (December 1999) pp. 1665–1672.

[33] D. Phatak and T. Goff, A novel mechanism for data streaming across multiple IP links for improving throughput and reliability in mobile environments, in: *Proceedings of IEEE INFOCOM*, New York, NY (June 2002) pp. 773–781.

[34] 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 IEEE Supercomputing (SC)*, Dallas, TX (November 2000).

[35] M. Allman, H. Kruse and S. Ostermann, An application-level solution to TCP's satellite inefficiencies, in: *Proceedings of Workshop on Satellite-Based Information Services (WOSBIS)*, Rye, NY (November 1996).

[36] 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 IEEE IPDPS*, Fort Lauderdale, FL (April 2002) pp. 434–443.



Hung-Yun Hsieh received the B.S. and M.S. degrees in electrical engineering from National Taiwan University, Taiwan, ROC. He is currently a Ph.D. candidate in the School of Electrical and Computer Engineering at Georgia Institute of Technology. His research interests include wireless systems, mobile computing, and network protocols. His thesis research focuses on addressing bandwidth scarcity and network heterogeneity in next-generation wireless data networks.

E-mail: hyhsieh@ece.gatech.edu

Kyu-Han Kim is currently a Ph.D. student in the Department of Electrical Engineering and Computer Sicence at Univeristy of Michigan at Ann Arbor. He received his M.S. degree in computer science from Georgia Institute of Technology, where he worked in the GNAN Research Group under the guidance of Prof. Raghupathy Sivakumar. His main research interests are mobile computing, wireless networks, and network performance evaluation.

E-mail: kyuhkim@eecs.umich.edu

Raghupathy Sivakumar received his Masters and Doctoral degrees in computer science from the University of Illinois at Urbana-Champaign in 1998 and 2000, respectively. He joined the School of Electrical and Computer Engineering at Georgia Institute of Technology as an Assistant Professor in August 2000. His research interests are in wireless network protocols, mobile computing, and network quality of service.

E-mail: siva@ece.gatech.edu