Implementation and Performance Evaluation of Indirect TCP

Ajay V. Bakre and B.R. Badrinath

Abstract—With the advent of small portable computers and the technological advances in wireless communications, mobile wireless computing is likely to become very popular in the near future. Wireless links are slower and less reliable compared to wired links and are prone to loss of signal due to noise and fading. Furthermore, host mobility can give rise to periods of disconnection from the fixed network. The use of existing network protocols, which were developed mainly for the high bandwidth and faster wired links, with mobile computers thus gives rise to unique performance problems arising from host mobility and due to the characteristics of wireless medium.

Indirect protocols [4] can isolate mobility and wireless related problems using mobility support routers (MSRs) as intermediaries, which also provide backward compatibility with fixed network protocols. We present the implementation and performance evaluation of I-TCP, which is an indirect transport layer protocol for mobile wireless environments. Throughput comparison with regular (BSD) TCP shows that I-TCP performs significantly better in a wide range of conditions related to wireless losses and host mobility. We also describe the implementation and performance of I-TCP handoffs.

Index Terms—Mobile computing, transport protocols, TCP, wireless medium, handoff, mobility support routers.

1 Introduction

Mobile internetworking involves adequately supporting network access from mobile (and wireless) computers. By adequate support we mean that a mobile user should be able to access the internetwork to obtain the same kind of services that were available to him/her from a desktop machine directly connected to the internetwork, albeit accounting for the slower speeds and higher error rates of the wireless links. Mobile internetworking thus involves addressing a broad gamut of issues related to routing, addressing, wireless medium and even disconnected operation.

It is possible to use existing fixed network transport protocols such as UDP [5] and TCP [6] with one of the mobile-IP proposals for communication between mobile hosts and the fixed network. This naive approach, however, has been shown to cause performance problems, especially when a mobile host switches cells or is temporarily disconnected [7]. In addition, all mobile-IP proposals attempt to hide mobility, disconnection and other features of mobile wireless computing from transport and higher layers thus ruling out any specialized handling of such features. In case of TCP for example, host mobility causes temporary disruption in the network layer connectivity resulting in loss of TCP segments. Error prone wireless links used by mobile hosts to communicate with the fixed network also contribute to the increased loss of TCP segments sent to or from mobile hosts. This loss of TCP segments triggers congestion control at the transmitting host which severely limits the end to end throughput. The congestion control mechanism used by TCP is clearly too conservative when faced with host mobility and wireless links.

One way to address the problem mentioned above is to modify the TCP specifications to take host mobility into account so that congestion control steps in only in case of a genuine network congestion. There are two problems with such an approach. First, it is infeasible to modify all the existing TCP implementations because of the sheer number of hosts on the Internet using TCP. Second, it is extremely difficult to determine the exact cause of a packet loss at the two end points of a TCP connection which may span multiple hops over the Internet. As an example, genuine congestion conditions may prevail on the fixed network when a mobile host switches cells. Attributing any packet loss in such a case purely to mobility can worsen the congestion on the fixed network.

Indirect TCP (or I-TCP), which is described in this paper, is based on an indirect protocol model [4], [8]. In this approach, an end-to-end TCP connection between a fixed host and a mobile host is split into two separate connections:

1) a regular TCP connection between the fixed host and the mobility support router (base station) currently serving the mobile host and
2) a wireless TCP connection between the mobility support router and the mobile host.

Use of mediation by the mobility support router (or indirection) at the transport layer allows special treatment of mobile hosts communicating over wireless links so as to address the problems mentioned earlier without sacrificing compatibility with existing fixed network protocols.

The remainder of this paper is organized as follows. Section 2 describes our system model for mobile wireless environments including the mobile computing testbed that

- *A.V. Bakre is with C&C Research Labs, NEC USA Inc., 110 Rio Robles, San Jose, CA 95134. E-mail: bakre@ccrl.sj.nec.com.*
- *B.R. Badrinath is with the Department of Computer Science, Rutgers University, Piscataway, NJ 08855. E-mail: badri@cs.rutgers.edu.*

For information on obtaining reprints of this article, please send e-mail to: transcom@computer.org, and reference IEEECS Log Number C97021.

we used to implement and test I-TCP. Section 3 gives a brief overview of indirect protocols. Section 4 describes the design rationale behind I-TCP. Subsequent sections deal with the implementation of various components of I-TCP including handoffs. Section 7 compares I-TCP throughput with that of regular TCP under a variety of conditions. Section 8 analyzes the handoff performance of I-TCP. Section 9 lists some example applications that we ported to use I-TCP. We compare our approach to some related work in Section 10. Finally, Section 11 lists our conclusions and some directions that we intend to pursue in the near future.

2 A System Model for Mobile Internetworking

Our system model for mobile internetworking consists of two separate network components:

1) A wired or fixed network which consists of local area networks interconnected by high speed links.
2) A wireless network which consists of separate wireless cells each of which is supported by a wireless base station and can provide connectivity to a few mobile wireless hosts (MHs). We assume that each base station (also known as a *mobility support router* or MSR) is directly attached to the fixed network and can route data packets to and from MHs in cooperation with other base stations (MSRs).

A mobile host can freely move into other wireless cells and it is the responsibility of MSRs to correctly route data packets to the mobile host (MH) regardless of where it is located. Each MSR may maintain some state information about the MHs that are currently in its cell.

In our system model, wireless cells occupy the periphery of the wired network as shown in Fig. 1, rather than forming a parallel network of their own. Wireless cells also depend on the wired network for routing data packets *between* cells. These cells thus form islands of wireless connectivity, interconnected by the fixed network.

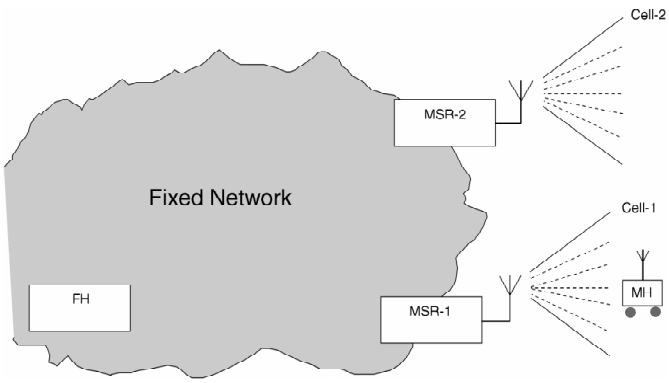


Fig. 1. A system model for mobile internetworking.

2.1 Mobile IP

Several mobile IP schemes have been proposed in the last few years to allow routing of IP datagrams to mobile hosts. One of the first such schemes was the Columbia Mobile IP protocol [9], [10] which used mobility support routers (MSRs) to support location management and routing of datagrams to mobile hosts within a *campus*. The Internet Engineering Task Force (IETF) recently completed standardizing a version of mobile IP, popularly known as the IETF Mobile IP protocol [11], which is expected to be widely deployed in the near future. The IETF scheme is slightly different from Columbia mobile IP in that it does not require a "campus" structure to be imposed on the wireless cells supporting mobile hosts. The key idea in mobile IP is that hosts sending data to a mobile host use its known IP address regardless of where the mobile is located. It is the responsibility of the home network of the mobile host to reroute the packets addressed to the mobile to its current location. We assume in our system model that a mobility aware IP layer protocol such as Columbia Mobile IP is available for routing and location management of mobile hosts.

2.2 Experimental Testbed

Our experimental testbed, which is shown in Fig. 2, consists of three wireless cells, each one of which is controlled by a mobility support router (MSR). All the MSRs are 33 MHz 486 PC-ATs with 16 MB memory and 400 MB disk drives. The wireless cells supported by these MSRs overlap with each other in terms of coverage area. The mobile hosts used in our experiments are 66 MHz 486 PC-ATs. All the mobile hosts and MSRs are equipped with 2Mbps NCR WaveLan cards for wireless communication. The WaveLan radio uses a configurable 16-bit MAC layer network ID in its transmissions and each such radio in turn only picks up transmissions that correspond to its own MAC layer identifier ignoring all others. This feature of the WaveLan radio is used to model nonoverlapped wireless cells by simply forcing each MSR to use its own unique MAC layer identifier. The mobile hosts can configure their radios to pick up the transmission from exactly one MSR at a given time.

The MSRs are also connected to 10 Mbps ethernet segments which are part of a single administrative domain. The MSRs run Mach 3.0 micro kernel from CMU with a Unix server (MK84/UX40) [12] that is patched with Columbia Mobile-IP source code dated July 1992. Additional modifications are needed in the MSR kernels (Unix servers) to support indirect TCP as described in a later section. The mobile hosts have a similar OS configuration but run a different version of the Unix server that has only minor modifications for supporting the protocols described in this paper.

3 Indirect Protocols

The basic idea behind the *indirect* protocol model [4], [8] is as follows: Whenever an interaction between two hosts on the internetwork, such as between a mobile host and a stationary host, involves communication over two drastically different kinds of media (e.g., wireless and wired), we split such an interaction into two separate interactions—one for each kind of communication medium. In our system model, described earlier, an indirect transport layer interaction between an MH and an FH consists of a fixed network protocol (e.g., TCP) used for communication between the FH and the MSR; and a wireless protocol (e.g., wireless

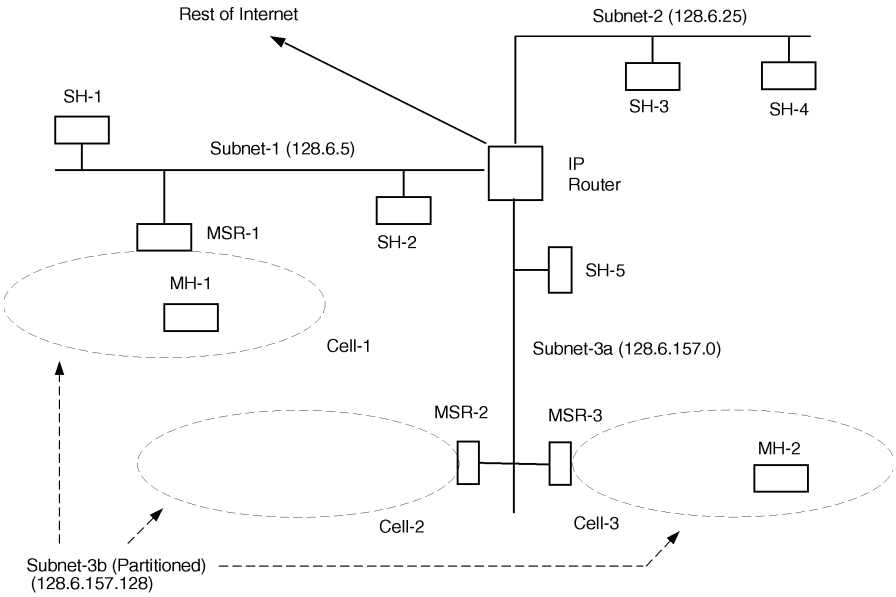


Fig. 2. Experimental mobile internetworking testbed.

TCP) for communication between the MH and the MSR. The highest protocol layer at which indirection occurs is determined by the MH application—an indirect transport layer can be used in conjunction with end-to-end session and presentation layer. On the other hand, if presentation requirements are different over wireless and wired links, then an indirect presentation layer protocol can be used. Furthermore, application layer proxies running on MSRs that support MH applications are examples of application layer indirection.

Our reasons for choosing the MSR currently serving a mobile host, as the point where the split in end-to-end protocols occurs, are as follows. Since the MSRs need modifications to the routing software for supporting mobile hosts anyway, adding modifications to support new protocols on the wireless side is relatively easier. Further, the MSR is a part of the fixed network that is closest to the wireless link. By splitting the protocol stack at the MSR, the corrective measures taken to address the change in the environment (from wired to wireless), are employed at the place where the change occurs. We assume that an MSR has enough resources (or at least that the required resources can be easily added) in terms of processing power and buffer space for supporting indirect interactions by mobile hosts within its cell. We also assume that MSRs are interconnected by a network that is at least an order of magnitude faster than the wireless links to allow sufficiently fast handoffs.

Notice that even though the indirect model replaces an interaction between a mobile host (MH) and a fixed host (FH) with one interaction between the MH and its MSR and another between the MSR and the FH, the FH does not see the MSR as its communicating peer. It actually sees the MH itself as its peer host. The MSR fakes an *image* of the MH which is used to communicate with the fixed hosts. This image is handed over to a new MSR in case the MH engaged in an indirect interaction switches cells.

4 I-TCP: An Indirect Transport Layer Protocol

I-TCP is a reliable stream-oriented transport layer protocol for mobile hosts which is based on the indirect protocol model. I-TCP is fully compatible with TCP/IP on the fixed network and is built around the following simple concepts:

1) A transport layer connection between an MH and an FH is established as two separate connections—one over the wireless medium and another over the fixed network with the current MSR being the intermediate point.
2) If the MH switches cells during the lifetime of an I-TCP connection, the center point of the connection moves to the new MSR.
3) The FH is completely unaware of the indirection and is not affected even when the MH switches cells, i.e., when the intermediate point of the I-TCP connection moves from one MSR to another.

When a mobile host (MH) wishes to communicate with some fixed host (FH) using I-TCP, a request is sent to the current MSR (which is also attached to the fixed network) to open a TCP connection with the FH *on behalf of* the MH. The MH communicates with its MSR on a separate connection using a variation of TCP that is tuned for wireless links and is also aware of mobility.

4.1 Indirect Transport Layer Advantages

At the transport layer, use of indirection results in the following benefits:

1) It separates the flow control and congestion control functionality on the wireless link from that on the fixed network. This is desirable because of the vastly different error and bandwidth characteristics of the two kinds of links.
2) A separate transport protocol for the wireless link can support notification of events such as disconnections,

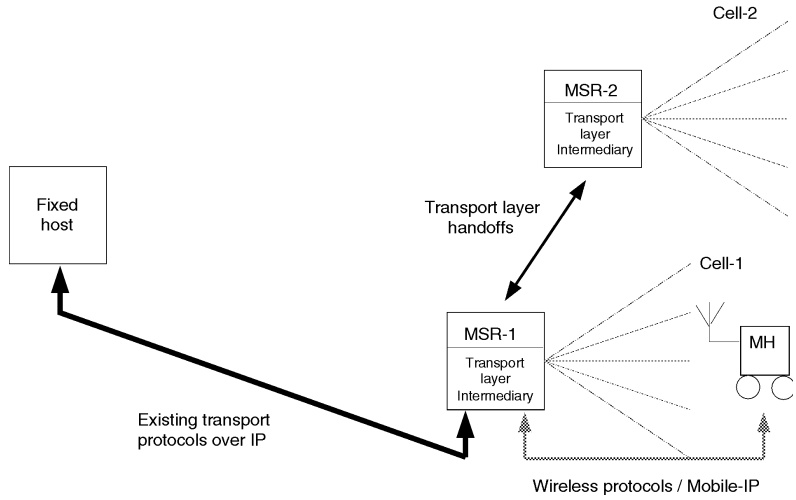


Fig. 3. Indirect transport layer.

moves, and other features of the wireless link such as the available bandwidth, etc., to the higher layers which can be used by *link aware* and *location aware* mobile applications.

3) Indirection at the MSR allows faster reaction to mobility and wireless related events compared to a scheme in which the remote communicating host tries to react to such events.
4) An indirect transport protocol can provide some measure of reliability over the wireless link for those applications which prefer to use unreliable transport over the fixed network.
5) Indirect transport protocols provide backward compatibility with the existing wired network protocols thus obviating modifications at fixed hosts for accommodating mobile hosts.
6) Indirection allows an MSR to manage much of the communication overhead for a mobile host. Thus, a mobile host (e.g., a small palmtop) that only runs a very simple wireless protocol to communicate with the MSR can still access fixed network services such as *WWW* which may otherwise require a full TCP/IP stack running on the mobile.
7) Indirect transport protocols allow the use of different MTUs over the wired and the wireless part of the connection. Since the wireless links have lower bandwidth and higher error rate, the optimal MTU for the wireless medium may be smaller than the smallest MTU supported by the wired network.

4.2 I-TCP Components

The I-TCP support software consists of the following components:

1) **MH side**—I-TCP can be accessed as a transport protocol by the applications running on a mobile host using special library calls. These library calls are similar in the interface they provide and the function they perform, to the socket calls made by an applications using regular end-to-end TCP. The I-TCP library is currently available for Unix systems based on 4.3 BSD [13]. This library hides from the applications the communication needed with the MSR for the establishment and tearing down of an I-TCP connection.
2) **MSR side**—Most of the functionality for supporting I-TCP connections lies with the MSR. The actual *bridge* connecting the wired and wireless parts of the connection consists of a user level Unix process pumping data from one part of the connection into the other. Handoff support for I-TCP connections is implemented in the MSR kernels.

4.3 Establishing I-TCP Connections

When a mobile host (MH) requests for an I-TCP connection to be established with a fixed host (FH), the MSR under which the MH is currently registered performs the following steps. It first establishes a regular TCP connection with the FH named in the connection request on behalf of the MH *using the IP address and port number of the MH for local endpoint parameters*—this constitutes the wired part of the I-TCP connection. Subsequently another connection is established between the MSR and the MH using a transport protocol that is tailored for mobile hosts and wireless links—this forms the wireless part of the I-TCP connection. Currently we use TCP itself for the wireless part of the connection with some modifications that are described later. For the wireless side TCP connection, the MSR uses its own IP address and port number to identify the local endpoint. Thus an I-TCP connection between the MH and the FH consists of two separate TCP connections; and is uniquely identified by the following three-tuple: <*mh-address, mh-portnumber*>, <*fh-address, fh-portnumber*>, <*msr-address, msr-portnumber*>.

4.4 Handing Off I-TCP Connections

As an example, Fig. 4 shows the setup for an I-TCP connection. In the figure, a mobile host (MH) which had first established a connection with a fixed host (FH) through MSR-1, moves to another cell under MSR-2. When the MH requests an I-TCP connection with the FH while located in the cell of MSR-1, MSR-1 establishes a socket with the MH address and MH port number to handle the connection with the fixed host. It also opens another socket with its own address and

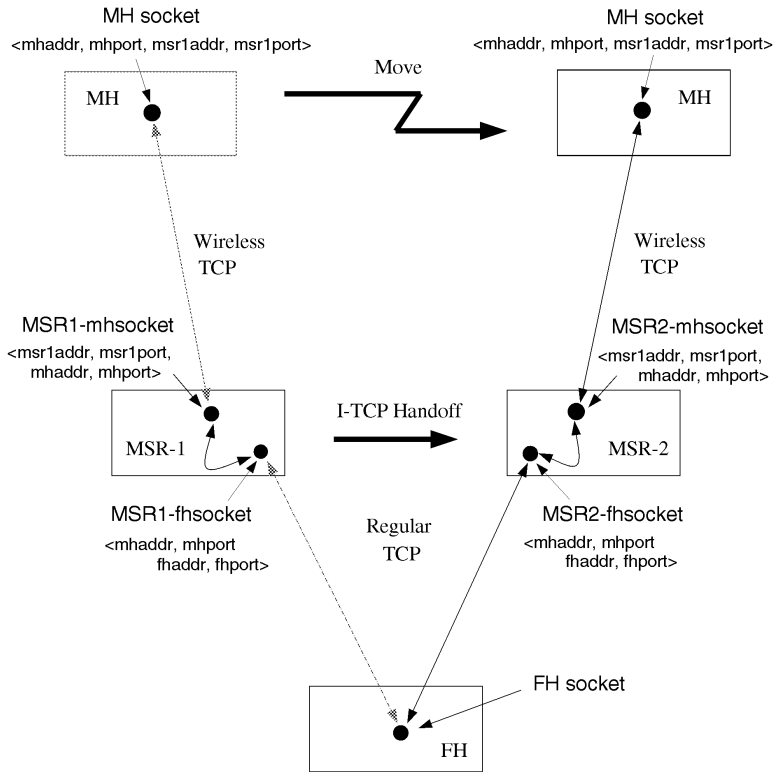


Fig. 4. I-TCP connection setup.

some suitable port number for the wireless side of the I-TCP connection to communicate with the MH.

When the MH switches cells, the state associated with two sockets of the I-TCP connection at MSR-1 is handed over to the new MSR (MSR-2). MSR-2 then creates two sockets corresponding to the I-TCP connection with the *same endpoint parameters* that the sockets at MSR-1 had associated with them. Since the connection endpoints for both wireless and the fixed parts of the I-TCP connection do not change after a move, there is no need to *re-establish* the connection at the new MSR. This also ensures that the indirection in the transport layer connection is completely hidden from the FH. We currently use a modified version of TCP itself for the wireless part of the I-TCP connection, although in a future version we plan to use a transport protocol that is optimized for the one hop wireless link.

4.5 End-to-End Semantics

Since I-TCP uses separate transport layer (TCP) connections for the wired and the wireless links, applications using I-TCP cannot rely on end-to-end transport layer acknowledgments. Many TCP-based applications such as **ftp**, however, use application layer acknowledgments in addition to the end-to-end acknowledgments provided by TCP. This is at least in part because TCP does not provide a mechanism to notify a sending application when data is actually removed by the receiving application from its socket buffers. *Thus, assuming that there are no MSR failures and that an MH does not stay disconnected from the fixed network indefinitely, using I-TCP instead of regular TCP does not compromise end-to-end reliability*. An MSR failure and subsequent reboot, however, results in the loss of I-TCP connections established via that MSR whereas an end-to-end TCP connection can typically survive such a failure. I-TCP is therefore well suited for applications such as **ftp** and **Mosaic**, in which a higher layer protocol (or the user) can retry failed connections in case of (hopefully rare) MSR failures. On the other hand, those applications which depend on the end-to-end transport layer acknowledgments, e.g., **telnet** are better off using regular TCP. We expect the former kind of applications to predominate in a mobile computing environment where mobile hosts will need to access information services from the fixed network. Such applications are also typically throughput intensive which we expect will benefit the most from the use of I-TCP.

5 I-TCP Implementation

Fig. 5 shows an outline of various I-TCP components. The MSRs in our testbed run a version of UX server patched with Columbia's Mobile IP and modifications to support I-TCP connections and handoffs. The mobile hosts in our testbed run a version UX server with minor modifications to support I-TCP connections. User lever **mhmicp** and **msrmicp** processes modified for I-TCP run on the mobile hosts and the MSRs respectively and execute the Mobile Internetworking Control Protocol (MICP) [9], which takes care of beaconing and registration of mobile hosts in Columbia's Mobile IP. In addition, a user level I-TCP daemon running at each MSR manages I-TCP connections originating from all the MHs within its cell. The I-TCP daemon also participates in handoffs with other MSRs when an MH switches cells.

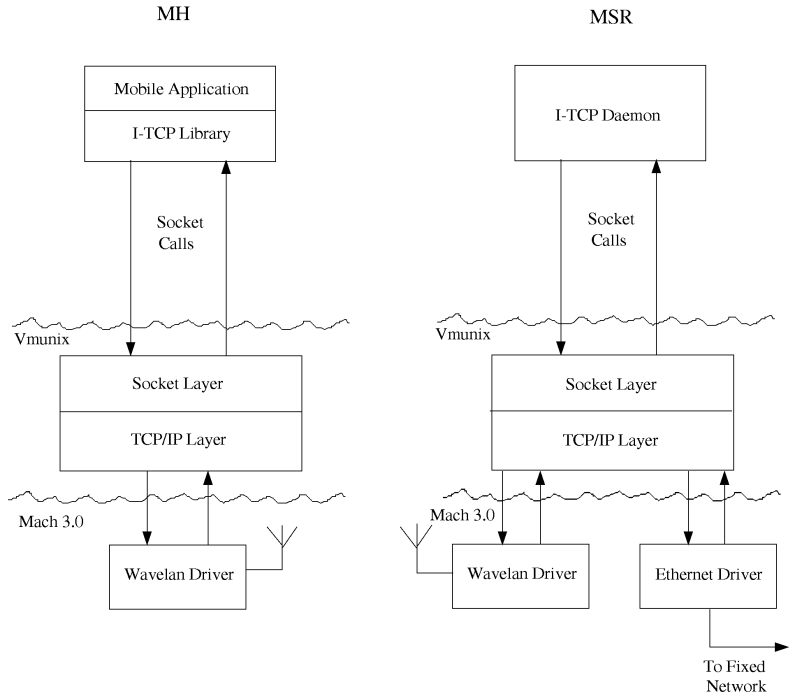


Fig. 5. I-TCP implementation outline.

5.1 I-TCP Interface at the MH

To establish an I-TCP connection instead of a regular TCP connection from an MH to a fixed host, we provide special I-TCP library calls which are similar to the socket interface provided in Unix 4.3 BSD. These library calls must be used by the MH applications instead of the regular socket system calls (**connect**, **listen**, **accept**, and **close**) for opening and closing an I-TCP connection. To send or receive data on an I-TCP connection however, the MH can use the regular send and receive primitives in Unix (e.g., **send** or **write** and **recv** or **read** system calls). The I-TCP calls provide a wrapper around the regular socket system calls to perform the necessary handshake with the MSR, the parameters of these library calls being the same as the corresponding socket system calls. A TCP-based application thus needs minimal changes to run on an MH with I-TCP. In addition, no modification is needed to the applications running on a fixed host to communicate with an MH using I-TCP.

The I-TCP library calls also inform the local **mhmicp** process about active I-TCP connections. This information is used in the registration protocol executed by an MH entering a new cell to assist in I-TCP handoffs as described later. The details of I-TCP calls are given below:

1) **itcp_listen**—Similar to the **listen** system call except that an indirect listening socket is also created at the current MSR by its I-TCP daemon on behalf of the MH. This listening socket at the MSR is bound to the same address and port number which identify the listening socket at the MH. Any connection attempt made by a remote host after the **itcp_listen** call by the MH is intercepted by the indirect listening socket at the MSR. Following this, the I-TCP daemon makes a connection over the wireless link to the listening socket at the MH thus completing two parts of the I-TCP connection.
2) **itcp_accept**—Similar to the **accept** system call except that the accepted connection request is received from the current MSR after a connection attempt made by a remote host is intercepted by the listening socket at the MSR. The wrapper around the **accept** system call provided by the I-TCP library makes indirection at the MSR transparent to the calling process by returning the address and port number of the remote host that initiated the connection (and not of the MSR).
3) **itcp_connect**—Similar to the **connect** system call except that the connection request is sent to the I-TCP daemon at the MSR which in turn makes a connection attempt to the remote host address specified in the **itcp_connect** call. If the remote connection attempt by the MSR succeeds, i.e., if the fixed network part of the I-TCP connection is successfully established, the I-TCP daemon then creates the wireless part of the connection with the MH process that issued the **itcp_connect** call.
4) **itcp_close**—Similar to the **close** system call for a socket, except that both (wireless and wired) parts of the I-TCP connection are closed.

We had the following choices in implementing the I-TCP interface at the MH:

1) As a library that emulated the socket interface in 4.3 BSD by trapping access to system calls **listen**, **accept**, **connect**, and **close** and calling I-TCP functions instead of the code for the system call. This would require no modification in MH applications but only one of regular TCP or I-TCP could be used by an application. Further the MH application would have to be relinked with the I-TCP library.
2) As a separate transport protocol accessible at the socket creation time. This would require a small

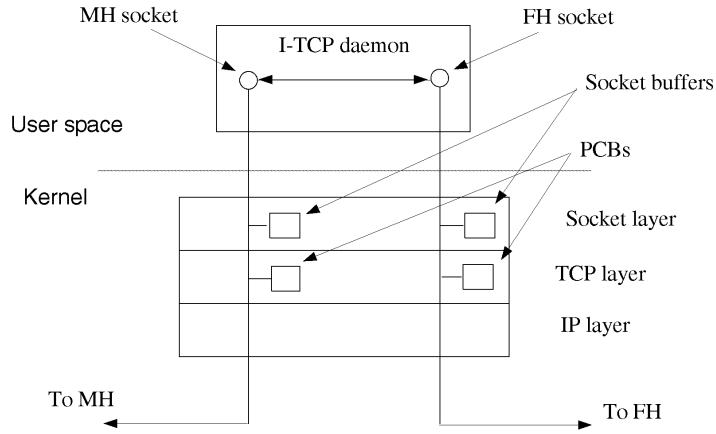


Fig. 6. I-TCP connection state at the MSR.

modification in MH applications and would also require kernel changes at the MH.

3) A kernel-based mechanism that trapped all TCP connection calls and redirected them to the I-TCP code. This would allow us to run existing MH applications to run without even the need to relink them with any library.
4) As a library with its own set of calls which would be identified by their names as distinct from but yet similar to the socket based connection establishment calls for TCP. This would require minimal changes in the MH applications, but would allow accessing TCP and I-TCP from the same application without requiring kernel changes at the MH to implement a new protocol.

We chose alternative 4 because it was easier to implement than the kernel-based mechanism required by alternative 3 and a separate protocol required by alternative 2. We also had an application in mind (described in a later section) that needed both TCP and I-TCP—that ruled out alternative 1.

5.2 MSR I-TCP Daemon

The I-TCP daemon running on every MSR is responsible for managing I-TCP connections through that MSR for all the MHs that are currently local to the MSR. The daemon maintains two open sockets for each active I-TCP connection between a locally registered mobile host (MH) and a non-MSR fixed host (FH) as shown in Fig. 6. One of these sockets is used for communication with the FH on the wired network and the other for communication with the MH on the wireless link. The I-TCP daemon binds its FH side socket to the address and port number of the MH using an extended **bind** system call. Such a binding allows the MSR to fake an image of the MH to the FH which is unaware of the indirection. Binding to the MH address and port number also provides a mechanism in the IP layer of the MSR kernel to grab the TCP segments that are sent by the FH to the MH address and port number.

Managing I-TCP connections at the MSR from a daemon running in user space involves additional copying overhead on each half of a duplex connection. At the MSR data received from the wireless side (MH) on an I-TCP connection has to go up through the TCP and socket layers in the Unix kernel to the I-TCP daemon in user space; and down again on the fixed side of the connection through the socket and TCP layers of the kernel to the IP output routine. On the other hand, a data packet received from the MH over an end-to-end TCP connection would be forwarded by the IP layer in the kernel to the fixed network with nominal processing overhead.

We chose to build a user level daemon to manage I-TCP connections for two reasons. First, it is easier to build an indirect higher layer protocol such as RPC on top of an indirect transport layer with a user level implementation. Second, a user level daemon is easier to modify and debug than a kernel resident implementation. The additional data copying overhead is not so critical for throughput intensive applications which are the primary focus of our implementation. This overhead does however cause an additional end-to-end latency of approximately 20 ms as seen by MH applications.

5.3 MSR Kernel Support

We implemented the support for transport layer handoffs in the Unix kernel since efficient handoffs are critically important for the performance of indirect transport layer. Modifications were also needed to the Unix kernel at the MSR to allow indirect connections to be established on behalf of local MHs.

5.3.1 IP Layer Support

At any MSR, we allow binding sockets to the addresses and port numbers of MHs that are currently local to the MSR. This is essential to capture the TCP packets on a per connection basis, which originate from fixed hosts and are addressed to a local MH. Binding to MH addresses also provides a mechanism using which the MSR can act as a proxy for the local MHs. For the wireless side of the I-TCP connection, the MSR uses its own IP address and port number. When an MH with an open I-TCP connection moves to another MSR, the MSR address on the wireless side of the connection needs to be changed to reflect the new MSR's address. Currently however, we keep the connection endpoints on the wireless side fixed even after a move for the sake of simplicity. For this purpose, at any MSR we also

allow binding sockets to addresses and port numbers of *other MSRs*. A small change was needed in the IP input routine at the MSR to send the IP packets that are addressed to MH address and I-TCP port numbers up to the TCP layer at the MSR instead of forwarding them to the MH using mobile IP routing. A list of such I-TCP port numbers (which correspond to active connections) is maintained on a per MH basis by the MSR along with the *mhinfo* entry of the MH used for routing in mobile IP [9].

5.3.2 Primitives for I-TCP Handoff

If a mobile host with an open I-TCP connection switches cells, handoff in mobile IP [9] causes rerouting of IP packets addressed to the MH so that they are forwarded via the new MSR. To maintain the indirect nature of the I-TCP connection, the I-TCP daemon at the new MSR needs to take over the connection from the previous MSR where the MH was located earlier. Transferring the state of the I-TCP connection from one MSR to another involves transferring the state of two sockets associated with the connection. Furthermore, this movement of sockets needs to be completely transparent from the perspective of the fixed host with which the MH is communicating. The socket state handoff also needs to be fast enough so as not to become a performance bottleneck even with frequent moves. While the transport layer handoff is in progress, the data segments that are in transit need to be buffered at one of the two MSRs. This buffering is necessary to avoid congestion control being triggered on either of the two sides (wired and wireless) of the I-TCP connection due to lost segments.

To achieve the objectives for a socket state handoff listed above, we designed and implemented our own handoff primitives in the form of special **ioctl** calls in the MSR Unix kernels. Complete details of a transport layer handoff using these primitives are given in a later section. Implementation details of these primitives can be found in [4]. SIOCSETSTATE are used by the I-TCP daemons to reliably transfer the state information corresponding to a connected socket from the kernel of one MSR to that of another with minimum copying overhead. We also provided two other **ioctl**s namely SIOCCREATE and SIOCDELETE to create a connected skeleton socket that does not have full state information available with it and to silently delete a socket whose state has been transferred to another MSR, respectively. Table 1 summarizes the four **ioctl**s. The **ioctl** mechanism was given preference over system calls primarily because it is much easier to add an **ioctl** than it is to add a system call to Unix. SIOCCREATE and SIOCDELETE operate on I-TCP sockets whereas SIOCGETSTATE and SIOCSETSTATE operate on a handoff socket which is used for MSR-to-MSR state transfer. The latter two take an additional argument which lists the I-TCP socket descriptor whose state is to be sent or received.

The SIOCGETSTATE **ioctl** *avoids unnecessary copying of data between kernel and user spaces by packing the send/receive buffers and state of the I-TCP socket inside the kernel and queuing it directly in the send buffer of the handoff socket. Similarly, SIOCSETSTATE avoids copying by retrieving the state and send/receive buffers of the I-TCP socket directly from the receive buffer of the handoff socket.*

TABLE 1
NEW SOCKET IOCTLS

Ioctl	Parameters	Description
SIOCCREATE	<target socket, connection endpoints>	Establish the connection endpoints of the target socket as if **bind** and **connect** have been called on the socket but without any communication with the peer host.
SIOCDELETE	<target socket>	Delete the target socket without any communication with the peer host.
SIOCGETSTATE	<target socket, moving socket>	Pack the state of the moving socket and place it in the send buffer of the target socket.
SIOCSETSTATE	<target socket, moving socket>	Unpack and establish the state of the moving socket from the receive buffer of the target socket.

5.4 TCP for Wireless Links

Though the indirection at the transport layer gave us the option of using a different transport protocol for the wireless part of I-TCP, we chose to modify and use TCP itself because of a readily available implementation. In the current version of I-TCP, the wireless side comes with the following modifications to the standard (4.3 BSD) TCP:

1) We added a new flag by the name TF_INDIRECT to mark I-TCP connections and a user level socket option TCP_INDIRECT to set this flag. The TF_INDIRECT flag is automatically set by the I-TCP library on the MH side and by the I-TCP daemon on the MSR side at the time of establishing an I-TCP connection. This flag is checked by the TCP code in the kernels of MH and MSR to make sure that the actions listed below are performed only on I-TCP connections.
2) When a move by an MH is detected by the TCP code on both sides of the wireless link, the retransmission timers for I-TCP connections are cleared at the MSR and the MH following which the wireless part of I-TCP immediately enters the *slow start* phase, thus forgetting the past congestion behavior. This allows the wireless part of I-TCP to come back up to normal speed faster than what normal congestion control and recovery in TCP would allow. Since the wireless part of I-TCP spans only one network hop, resetting the retransmission timers cannot have any adverse effect on network congestion that might occur in the fixed network at the same time when an MH switches cells.
3) When an MH reconnects after a disconnection, similar actions are performed over I-TCP connections as described above in case of a move. This makes sure that disconnections do not cause long pauses in I-TCP connections after wireless contact is reestablished.

Detection of moves on the MSR side is easy since after a move, the I-TCP daemon at the new MSR receives the state related information from the previous MSR about the I-TCP connections opened by the MH. Thus, the TCP code at the MSR resets congestion related parameters as mentioned above when a connection handoff is completed in the kernel.

Detection of moves at the MH is slightly more difficult. We have implemented notifications for these events that travel the TCP/IP protocol stack upwards from the bottom using the protocol-to-protocol *control input* interface in 4.3 BSD. More details on this notification mechanism can be found in [4].

6 Mobility Management—Interaction with Mobile IP

In Columbia mobile IP protocol, a user level **msrmicp** daemon at the MSR is responsible for sending periodic beacons and for registration and expiration of mobile hosts. Similarly, a user level **mhmicp** daemon running at each mobile host is responsible for listening to MSR beacons, sending greeting messages, and informing the MSR about its previous location so that the MSR can send a forwarding pointer to the old MSR which was earlier routing IP packets for the MH. We integrated I-TCP handoff with the handoff in mobile IP since both need the information about MHs entering a cell and their previous locations.

6.1 MICP *i*-Entries

We extended the Mobile Internetworking Control Protocol (MICP) [9] to carry information about I-TCP connections in the initial greeting that an MH sends to an MSR on entering the MSR's cell. At every MH, the **mhmicp** daemon maintains the endpoint information for each active I-TCP connection opened by the MH. This information is maintained in what are known as MICP *i*-entries (*i* for indirect) and includes the end point parameters (corresponding to the MH, the peer FH and the first MSR where the connection was initially established) for the I-TCP connection. An *i*-entry is created when the I-TCP library linked with an MH application sends a message to the **mhmicp** process as part of an I-TCP call establishing a connection. When an MH enters a new cell, i.e., when the **mhmicp** process at the MH hears a beacon in the new cell after losing contact with the previous MSR, a greeting message (called MICP_GREET packet in mobile IP) is sent to the new MSR. The **mhmicp** process sends its list of MICP *i*-entries in this greeting message in addition to the address of the previous MSR.

6.2 Handoff Sequence at MSRs

On the MSR side, when the **msrmicp** process receives a greeting from an MH entering its cell that contains MICP *i*-entries, it sends a copy of the greeting message to the local I-TCP daemon. The I-TCP daemon establishes skeleton sockets for each I-TCP connection from the *i*-entries using SIOCCREATE calls. It also sends an ACK to the **msrmicp** daemon after which a forwarding pointer (MICP_FWDPTR) is sent to the previous MSR named in the MH greeting message indicating that the new MSR's I-TCP daemon is ready for handoff.

When the **msrmicp** process at the previous MSR receives a forwarding pointer from the new MSR, it updates its data structures to reflect the new location of the MH. It also sends a copy of the forwarding pointer to its local I-TCP daemon which then establishes a handoff connection with the I-TCP daemon at the new MSR. The I-TCP daemon at the previous MSR then sends the state of each I-TCP connection to the I-TCP daemon at the new MSR over the handoff connection using SIOCGETSTATE calls to transfer the state of individual I-TCP sockets. The I-TCP daemon at the new MSR concurrently executes SIOCSETSTATE calls to receive the state of each I-TCP socket over the handoff connection and restarts each I-TCP connection. The handoff algorithms executed by the new MSR and the previous MSR are shown in Fig. 7. The I-TCP handoff sequence is graphically shown in Fig. 8. The intermediate stages of an I-TCP connection at the two MSRs during handoff can be seen in Fig. 9.

```
New MSR:
    wait for the handoff request from old MSR
    receive the MH address
    lookup the MH entry in the tables
    for every I-TCP connection do {
        receive the i-entry for connection
        lookup the connection entry
        receive and install the state of MH side socket
        restart the wireless part of connection
        receive and install the state of FH side socket
        restart the wired part of connection
        receive pending user level data buffers
        install MH and FH side user level buffers
        fork worker threads connecting two parts
            of the I-TCP connection
    }

Old MSR:
    locate the entry for MH that moved out
    for every I-TCP connection do {
        signal worker threads to terminate
    }
    make a handoff connection to the new MSR
    send the MH address
    for every I-TCP connection do {
        send the i-entry for the connection
        freeze and send the state of MH side socket
        freeze and send the state of FH side socket
        send MH and FH side user level data buffers
        delete MH and FH side sockets
        free resources and delete the MH entry
    }
```

Fig. 7. Handoff algorithms for MSRs.

7 I-TCP Throughput Performance

We present performance figures for experiments conducted using the **ttcp** benchmark which measures TCP throughput between two hosts. The throughput experiments were conducted on our wireless testbed which was described earlier in Section 2. We experimented with two distinct cases to study the performance of I-TCP for connections spanning over local area and wide area networks, i.e.,

1) when the FH to MH communication involved only a few hops within the Rutgers LCSR administrative domain, and
2) when the FH to MH communication involved a long-haul link over the Internet.

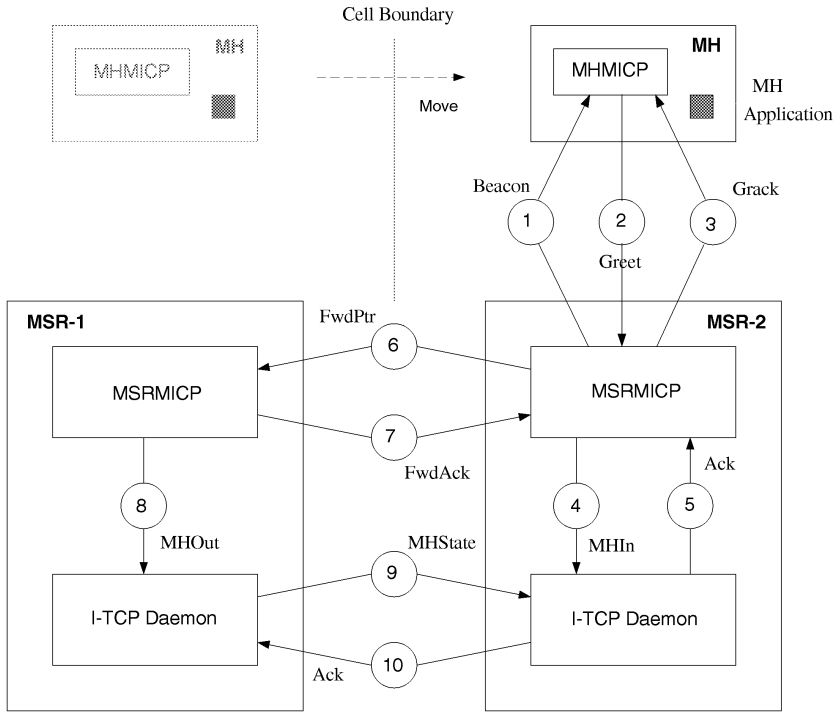


Fig. 8. I-TCP / Mobile-IP handoff sequence.

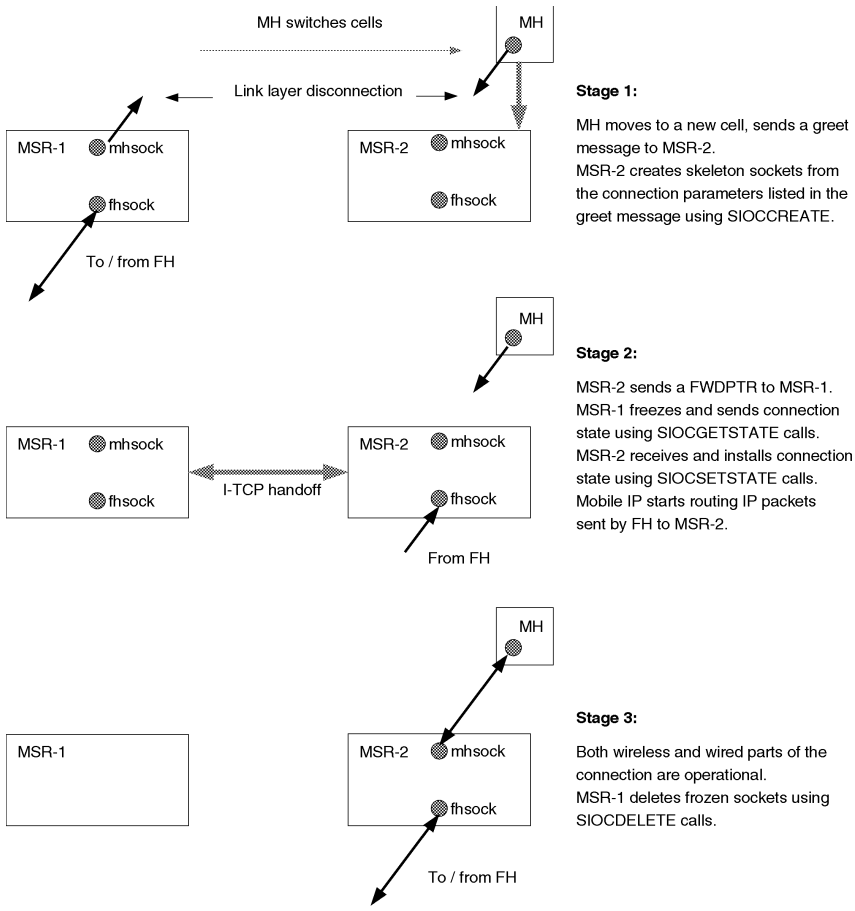


Fig. 9. Intermediate stages in I-TCP connection handoff.

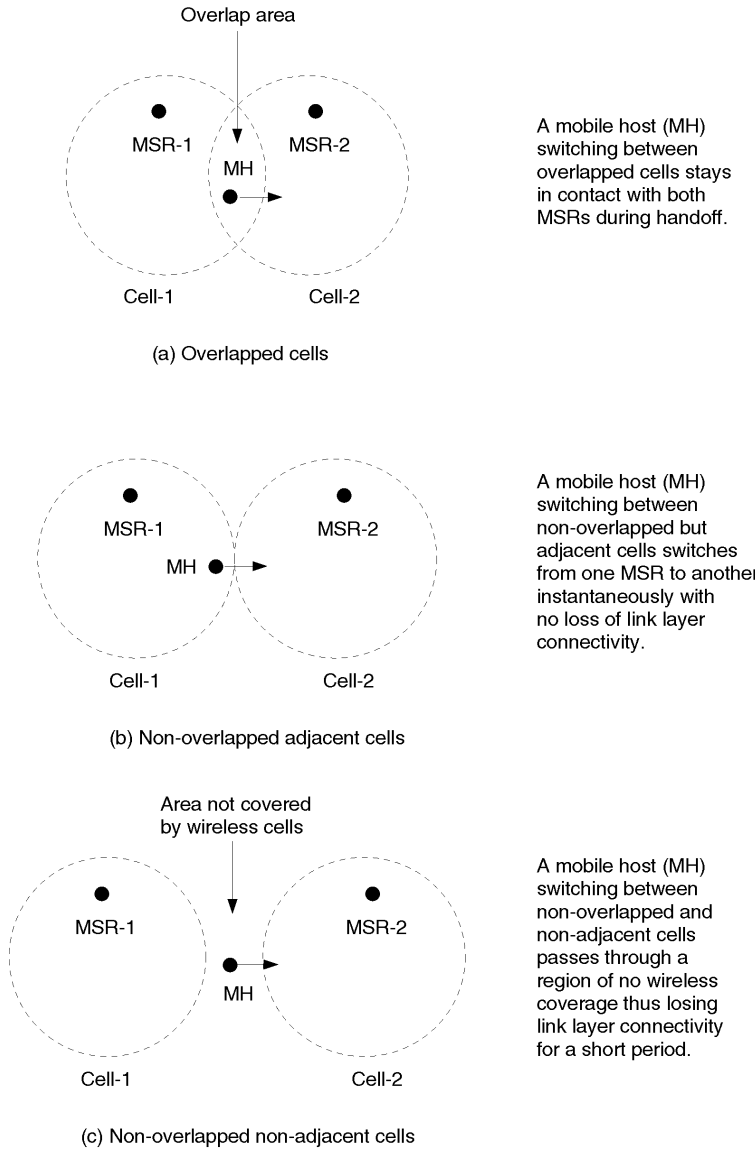


Fig. 10. Wireless cell configurations.

7.1 Experiments to Study the Effects of Mobility

Our experiments were inspired by similar experiments reported by Cáceres and Iftode [7] to study the effect of mobility on reliable transport protocols. In all our experiments the end-to-end throughput was measured at the receiving host with four different cell configurations:

1) **No Moves**—The MH stayed in one wireless cell during the lifetime of a connection.
2) **Moves between overlapped cells**—MH switching between overlapped cells every eight seconds such that the MH stayed in contact with the previous MSR during handoff. For a brief period after switching cells, the MH continued to receive packets from the previous MSR before the Mobile-IP routing adjustments take effect. Overlapped cell configuration is shown in Fig. 10a.
3) **Moves between nonoverlapped cells with 0 second between cells**—In case of nonoverlapped cells, the cell boundaries were sharply defined and therefore no communication was possible with the previous MSR after a move to another MSR. This is shown in Fig. 10b. The MH started looking for a beacon from the new MSR immediately after a move and thus in the worst case the link layer connectivity could be lost for one full interval between successive beacons which was one second in our testbed. The cell switching again occurred every eight seconds.
4) **Moves between nonoverlapped cells with one second between cells**—Same as in 3 above but, in this case, the MH started looking for a beacon one second after moving out of the previous cell. As in the previous case, an additional one second could elapse before a beacon was received by the MH and the link layer connectivity was reestablished. This configuration is shown in Fig. 10c where MH loses connectivity for a brief period while crossing the cell boundary.

Geographically, the two MSRs serving the two wireless cells were close together in our setup so that the two cells

TABLE 2
LOCAL AREA PERFORMANCE COMPARISON WITH HANDOFFS

Protocol	No moves	Overlapped cells	Nonoverlapped cells with 0 sec. b/w cells	Nonoverlapped cells with 1 sec. b/w cells
FH to MH throughput in Kbytes/sec.				
Regular TCP	65.5	62.6	38.7	23.7
I-TCP	70.1	65.4	44.8	36.3
MH to FH throughput in Kbytes/sec.				
Regular TCP	76.3	71.5	53.1	35.9
I-TCP	87.6	74.3	67.9	58.0

TABLE 3
WIDE AREA PERFORMANCE COMPARISON WITH HANDOFFS

Protocol	No moves	Overlapped cells	Nonoverlapped cells with 0 sec. b/w cells	Nonoverlapped cells with 1 sec. b/w cells
FH to MH throughput in Kbytes/sec.				
Regular TCP	13.3	13.3	8.9	5.2
I-TCP	26.8	28.0	19.1	16.0
MH to FH throughput in Kbytes/sec.				
Regular TCP	31.0	30.0	16.9	10.6
I-TCP	71.3	61.7	57.4	46.4

had a large overlap area. Non-overlapped cells were simulated with the two MSRs transmitting using different Wavelan (MAC layer) network IDs. Cell switching was implemented in software to allow better control on the timing of cell crossovers.

Table 2 shows local area throughput comparison between I-TCP and regular TCP for 4 MB data transfer between a mobile host (MH) and a fixed host (FH), both when MH is the receiver and the sender. Table 3 shows similar comparison for 2 MB data transfer over a wide area connection between an FH and an MH both when MH is the receiver and the sender. The experiments were conducted using a TCP window size of 16 KBytes and read/write buffer sizes of 2 KBytes.

7.1.1 Performance Over Local Area

With local-area experiments, we observed that I-TCP performed slightly better compared to regular TCP when the MH stayed within one cell. In the second case when the MH switched between two completely overlapped cells, the link-layer connectivity was maintained at all times since the MH was in contact with the new MSR as well as with its previous MSR during handoff. There was still some degradation in TCP throughput since the TCP segments that are in transit during handoff are delayed because of IP layer routing adjustments by the MSRs. I-TCP performance suffered only marginally in this case despite the additional overhead of I-TCP state handoff between the two MSRs on every move. We believe that the main reason for improved performance with I-TCP in the first two test cases was that the sending host (FH) saw more uniform round-trip delays for data segments with I-TCP than with regular TCP. In other words, the MSR receiving data over the fixed network (in case of I-TCP) resulted in a better pacing of the sending fixed host than the MH receiving data over the wireless link (in case of regular TCP). Loss of data segments over the wireless link, though infrequent, was also responsible for the difference in performance since I-TCP seemed to recover faster from a packet loss than regular TCP.

The two cases of nonoverlapped cells, where the MH temporarily lost contact with the fixed network (for 0 and one second, respectively) before such contact was reestablished at the new MSR, affected the end-to-end throughput more severely. With regular TCP, congestion control kicked in at the FH on every handoff because of packet loss and it took some time after a cell crossover before the FH was able to send data again at full speed. In addition, the exponential back off policy of TCP resulted in the FH going into long pauses that continued even after the MH was ready to communicate in its new cell.

In case of I-TCP however, a cell crossover by the MH caused MSR buffers to be filled with data sent by the FH which was not sent over wireless because the MH moved out of the MSR's cell. When MSR buffers were full, flow control on the wired part of the connection prevented the FH from sending more data. This was achieved by the MSR advertising a shrinking receive window size. After a handoff the new MSR, which receives the indirect TCP connection from the previous MSR, could send more data to the MH over the wireless link. When MSR buffers started clearing up again, TCP flow control over the wired network allowed the FH to send more data on the connection. In case of I-TCP, loss of data segments because of mobility thus caused flow control to be kicked in on the wired network; whereas, in case of regular TCP, the loss of data segments caused congestion control to be kicked in. Congestion control did kick in on the wireless link between the MSR and the MH, however, and so did exponential backoff. We found that resetting the TCP retransmission timer at the new MSR immediately after an I-TCP handoff forced the MSR to initiate a slow-start on the wireless link, and was enough to quickly get the wireless part of I-TCP out of the congestion recovery phase. In the worst case, when the MH lost connectivity with the fixed network for one second, I-TCP showed an improvement by a factor of about 1.5 over regular TCP.

In the experiments where the MH sent data to the FH, the performance in all categories for both TCP and I-TCP was slightly better than corresponding numbers in the case

when MH was the receiving host. There are three main reasons for this improved performance when MH was the sender:

1) In our experiments, MH was a 486 machine running at 66 MHz whereas the MSR was a 486 machine running at 33 MHz. This means that the host transmitting over the wireless link was a faster machine in the case when MH was the sending host, which is likely to result in a better throughput.
2) When wireless link is the bottleneck in an end-to-end connection, attaching the sender directly to the wireless link is likely to yield better performance than the case where the sender is a few hops away from the wireless link. In other words, better performance can be expected when the bottleneck link is the first one rather than the last one because in the latter case, the host transmitting over wireless (MSR) can only transmit data as fast as it receives it from the fixed network whereas in the former, the transmitter (MH) always has data to transmit.
3) I-TCP connection state at the MSR, which was handed off to another MSR after a move, was much lighter in the case when MH was the sender than when MH was the receiver, thus requiring less time for handoff. This is because the bottleneck link was the first hop when MH was sending data—the MSR was always able to send away any data received from the MH to the FH over the wired network.

7.1.2 Performance Over Wide Area

Our wide area experiments highlight the benefits of I-TCP even more clearly. Due to the relatively long round-trip delays with wide area connections, any packet loss over the wireless link severely affects the end-to-end throughput of regular TCP. This is because *the time needed to recover from falsely triggered congestion control increases with the round-trip delay*. Similarly, any perturbations such as cell crossovers or transient changes in the observed round-trip delay, have a more drastic effect over wide area connections than over local area connections.

For the first two test cases, i.e., when the MH stayed within one cell and when it switched between overlapped cells, the observed performance of I-TCP was about two times better than that of regular TCP. Since there was no packet loss because of mobility in these two cases, the performance improvement with I-TCP comes entirely from separating the TCP connections over wired and wireless links. This separation is beneficial in two respects. First, since the retransmissions due to lost segments over wireless (even though such losses were infrequent in our experiments) were restricted to the wireless link, the recovery from such losses was much faster compared to the end-to-end retransmission and recovery done by regular TCP. The second factor for improvement in the throughput was the aggregating effect at the MSR which received TCP segments of size 512 bytes[1] from the FH and sent segments of 1440 bytes[2] over the wireless link to the MH. This points to another parameter, namely the segment size, that can be tuned to suit a particular wireless link regardless of the segment size chosen by the TCP implementation on the wired network. We did not observe any significant degradation in performance with the MH switching between overlapped cells either with I-TCP or with regular TCP which suggests that the effect of variation in round trip delay because of handoff related IP level routing changes was negligible for wide area connections.

With the MH moving between nonoverlapped cells, the throughput with regular TCP dropped to almost a third (61% degradation) of the no-moves throughput in the case when the MH lost contact with the fixed network for one second. With I-TCP, the corresponding degradation in throughput was only 40%. The net effect was that I-TCP throughput in this case was three times better than that of regular TCP. The main reason for this improved performance with I-TCP was that retransmissions due to packets lost on the wireless link (due to moves and due to wireless errors) were confined only to the wireless part of I-TCP which can recover much faster from the congestion control phase because of the following two factors:

1) much shorter round-trip delay between MH and MSR as compared to the delay between MH and FH, and
2) resetting the retransmission timer by I-TCP at the MSR immediately after a handoff.

Wide area performance numbers for handoff experiments with MH as the sender show that I-TCP outperformed regular TCP by factors ranging from about 2.3 to 4.5. The improvement over TCP was even better than in wide area experiments with MH as the receiver, which can be attributed to the fact that the separate wireless connection between MH and MSR was not constrained by the lack of data on the sending side. Thus, when conditions on the wireless link were good (i.e., from the time a handoff was completed until it was time to switch cells again for the MH), I-TCP could keep the MSR buffers full, since the maximum observed end-to-end throughput of 70 KBytes/sec was easily sustainable on the 2 Mbps wireless link. I-TCP throughput thus suffered only during the period when the MH was disconnected (while switching cells) and during I-TCP handoffs. As soon as an I-TCP handoff was completed, the sender reset its congestion state and performed a slow start which brought the connection over wireless to full speed reasonably fast because of the short round trip delay between the MH and the MSR. TCP throughput on the other hand, suffered not only during periods of disconnection but even during periods with good link conditions after the MH had established wireless connectivity in the new cell because of exponential back-off and congestion control that was triggered during a cell crossover. Another observation that can be made from the wide area throughput numbers is that the throughput when MH was the sender was twice as much the throughput when MH was the receiver. Although we expected the throughput in the former case to be a little better, as explained in performance experiments over local area, the main reason for throughput difference in the two cases was that these experiments were performed at different times and thus under different traffic conditions possibly including different routes over the wired network.

1. Maximum segment size for wide area TCP connections.
2. To allow an additional IP header with IPIP encapsulation.

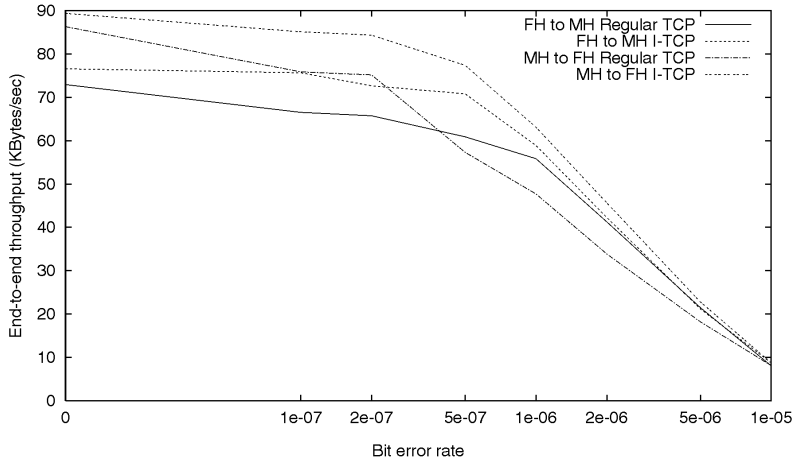


Fig. 11. Local area performance comparison with wireless losses.

7.2 Experiments to Study the Effects of Wireless Losses

In addition to studying I-TCP performance under host motion, we also performed experiments to study the effects of packet loss due to bit errors over the wireless link on I-TCP and regular TCP. Bit errors for these experiments were simulated by introducing a packet dropping routine in the input processing code of the ethernet (IF) layer in Unix both at the MH and at the MSR. The packet dropping routine was based on a pseudorandom number generator that produced uniformly distributed numbers in a specified range. The probability of finding a packet in error was determined from the length of the packet and the configured bit error rate. Such a simple model for simulating bit errors provides reasonable error characteristics for low error rates but becomes inaccurate as the reciprocal of the bit error rate approaches the length of the packet. Since the error simulation was used both at the MH and at the MSR, it affected the data traffic (traveling in one direction) as well as the acknowledgments (traveling in the other).

7.2.1 Performance Over Local Area

Fig. 11 shows a comparison of I-TCP throughput with that of regular TCP for 2 MB data transfer between a fixed host (FH) and a mobile host (MH) for different bit error rates (BER) when the FH is located only a couple of ethernet hops away from the wireless subnet. The BER of 0 corresponds to *no simulated packet loss*. The comparison shows that I-TCP performed better than regular TCP for error rates of up to 2×10^{-6}. For excessive wireless losses that characterize even higher error rates, the throughput of I-TCP and regular TCP was about the same. The TCP implementations used by the FH, as well as those used by the MSR and the MH in our experiments did not have the fast retransmit and fast recovery mechanisms recently proposed [14]. Also, the minimum TCP retransmission timeout was 500 msec. even for the wireless link, which clearly did not allow early detection of lost segments. In such circumstances, there was not much difference between retransmitting lost segments from the MSR and from an FH that was only a couple of ethernet hops away. Employing a TCP implementation (or a different transport protocol) for the wireless part of I-TCP, which uses smaller timeouts and possibly selective acknowledgments, should further improve I-TCP performance over regular TCP.

Local area experiments with wireless losses, when MH was the sending host, showed performance improvement with I-TCP in much the same way as in the case when MH was the receiver. The only difference was that the throughput for both TCP and I-TCP was slightly better than the case when MH was the receiving host. The reason for this was explained earlier with mobility experiments.

7.2.2 Performance Over Wide Area

Fig. 12 shows a comparison of I-TCP throughput with that of regular TCP when the communication between the FH and MH involved a long haul link. It can be seen that I-TCP throughput was about twice as much as that of regular TCP for error rates of up to 5×10^{-6}. Even for a BER as high as 10^{-5}, I-TCP performance was significantly better than regular TCP. The reasons for this improvement are the same as mentioned earlier in case of wide area mobility experiments, viz.,

1) faster recovery from wireless losses because of shorter round trip delay over wireless, and
2) aggregating effect at the MSR.

The aggregating effect (i.e., combining 512 byte segments from the wide area connection over wired network into 1,440 byte segments over local area wireless connection) worked in favor of I-TCP when error rates were low but with higher error rates, when smaller packets were more likely to get through than the larger packets, the same effect worked against I-TCP. This can be seen in Fig. 12 as the throughput of I-TCP dropped faster than that of regular TCP when error rates were high. In practice, the packet size (MTU) over wireless will be decided by the expected error rate and the link speed. Nevertheless, I-TCP can be used to match the MTU of the wireless link with that of the wired network. With regular TCP, any mismatch in the MTUs will cause inefficient utilization of available packet size in one direction and IP layer fragmentation in the other.

Wide area experiments with wireless losses when MH was the sending host show performance improvements

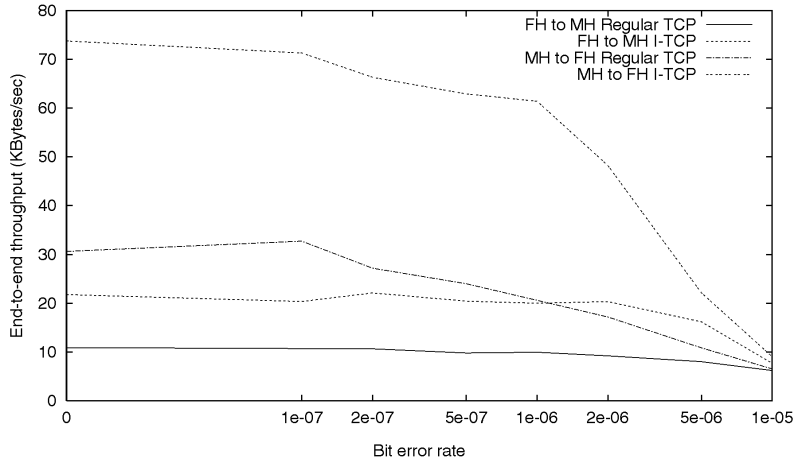


Fig. 12. Wide area performance comparison with wireless losses.

similar to the case when MH was the receiving host. The absolute throughput numbers are higher both for TCP and I-TCP, mainly because the two experiments were performed at different times and therefore under different traffic conditions on the fixed network. The performance improvement with I-TCP was somewhat better when MH was the sender than when MH was the receiver especially with low error rates. This is again because I-TCP was able to exploit the periods with reasonably good wireless link conditions better when MH was the sender than when MH was the receiver.

7.3 Performance Summary

From the performance comparison shown earlier between I-TCP and end-to-end TCP we can draw the following conclusions:

1) Performance improvement with I-TCP is small to moderate when data is transferred over a local area connection.
2) Performance improvement with I-TCP is moderate to large when data is transferred over a wide area connection.

It is not very difficult to pinpoint the reason for the difference in performance improvement between local area and wide area connections. First, with local area connections, wireless link is the part with lower bandwidth than the wired network. On the other hand, the long haul wired network spanning the continent is the part with lower available bandwidth than the wireless part because the aggregate bandwidth over the backbone network is shared among a large number of users. This means that the best throughput we can expect in case of local area connections is determined by the capacity of the wireless link, whereas, in the wide area cases, it is determined by the conditions prevalent over the backbone network at the time of experiment. The best throughput we saw for data transfer between an MH and its MSR over a TCP connection between two user level processes with the hardware and software configuration described in Section 2 was of the order of 100 KBytes/sec. From the local area performance figures given earlier, we see that I-TCP throughput was fairly close to this number when conditions over the wireless link were good. TCP performance was only slightly inferior because of the increased delays in receiving end-to-end acknowledgments from the receiver. With wide area experiments we saw that the best throughput we could achieve with TCP between MH and FH was only 50–70% of the throughput between the MSR and FH. This degradation was introduced entirely by the wireless link even though it added only about 20 ms to the end-to-end delay of 100 ms.

When we introduced host mobility, which caused disruption in the end-to-end data transfer, we saw that degradation in I-TCP performance in all the categories of local and wide area experiments was slower than the corresponding degradation in TCP throughput. One major reason for the slower degradation was the corrective action taken by the host transmitting over wireless (MH or MSR), which made the transmitter forget the past congestion history immediately after a move by the MH.

With artificially introduced bit errors over the wireless link, we saw small performance gains with I-TCP in the local area experiments, although the corresponding gains in the wide area experiments were impressive. These performance benefits were due entirely to the reason that I-TCP was able to isolate the lossy wireless link from the rest of the connection resulting in faster recovery from a lost segment. We did not incorporate any special measures to recover from wireless losses for the I-TCP connection over wireless, but we believe that using a smaller retransmission timeout and fast retransmissions possibly coupled with selective acknowledgments should result in a much better performance by I-TCP.

8 Handoff Performance

For handoff performance, we measured the handoff time for an MH that had one I-TCP connection established with a fixed host two Ethernet hops away from both the MSRs used in the experiments. The MH kept switching back and forth between the two MSRs every 10 seconds. The handoff time was measured at the new MSR as the time elapsed between the instant when a greeting message arrives from the MH entering the cell and the instant when the I-TCP

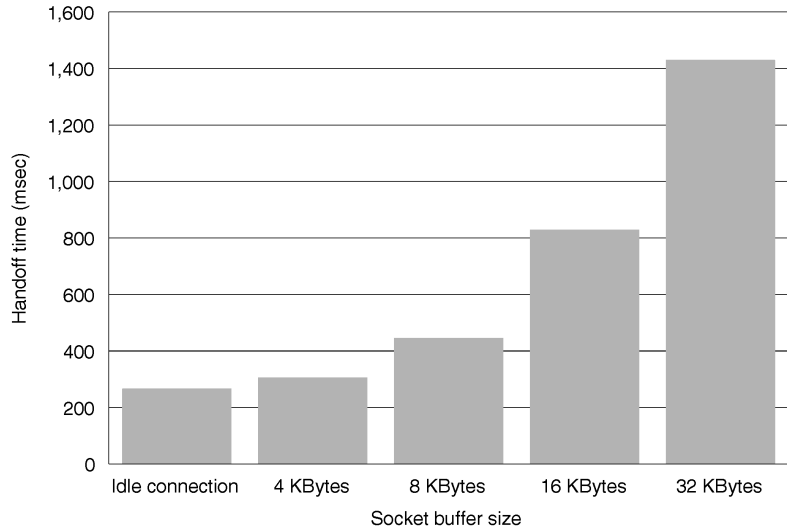


Fig. 13. Handoff times for different socket buffer sizes.

handoff is completed. Handoff measurements were performed for different socket buffer sizes which determine the amount of state to be transferred.

Fig. 13 shows the time taken by I-TCP connection handoff using different socket buffer sizes in a nonoverlapped cell configuration where cell boundaries are sharply defined and cell switching is instantaneous. The case marked as *idle connection* in the figure denotes the handoff time for a connection on which no data is being sent. The graph shows that it took about 265 ms just to transfer the connection state with no (empty) socket buffers. The size of such an idle connection state was about 500 bytes which included the socket data structures and the TCP control blocks for the two (MH side and FH side) sockets plus some control information. In all the other cases, the handoff times were measured with the FH pumping data as fast as it could, given that the maximum window size for both parts of the I-TCP connection was equal to the socket buffer size.

Our measurements show that the handoff time increased with the size of socket buffers. This can be explained as follows: Because of the large difference in the bandwidths of the wireless and the wired media (2 Mbps WaveLAN versus 10 Mbps Ethernet), the socket buffers at the MSR remained full most of the time since data was being pumped from a higher bandwidth link to a lower bandwidth link. Therefore, when the MH switched cells, the I-TCP connection handoff involved transferring a full send buffer for the MH side socket and a full receive buffer for the FH side socket from one MSR to the other. Thus the I-TCP handoff in case of 32 Kbyte socket buffers added up to 64 Kbytes (and possibly some more due to any pending user level buffers) to the idle connection state of about 500 bytes. The handoff time using 32 Kbytes of socket buffers was observed to be 1,430 ms.

8.1 Handoff Analysis

We analyzed the I-TCP handoffs to determine how much time was spent in each step. The results of our analysis are summarized in Fig. 14 for two representative cases:

1) idle connection and
2) active connection with 32 Kbyte socket buffers.

All timing data shown is with reference to the instant when a greeting was received by the MSR from an MH entering its cell. In the case with idle connection, the new MSR took about 60 ms to establish the socket skeletons from the information contained in the MH greeting message after which it sent a forwarding pointer (MICP_FWDPTR) to the old MSR. It took an additional 150 ms before the handoff request arrived at the new MSR. The actual state transfer took only 55 ms.

In the second case, with 32 Kbyte socket buffers, we see that after the handoff request was received by the new MSR, another 1,230 ms elapsed before the handoff was completed. Out of these, the state of MH side socket was transferred in 790 ms, whereas the state of FH side socket took 410 ms. Thus with large socket buffer sizes, a major part of the handoff time is spent in transferring the buffered data. The time to transfer MH side socket was larger than the corresponding time for the FH side socket mainly because the TCP handoff connection between the two MSRs was still in a slow start phase while transferring the MH side socket state.

The above analysis of handoff events suggests that the bandwidth available for handoff traffic between two MSRs is an important factor in determining the time required for handoffs, especially if a mobile host has multiple indirect connections at the time it switches cells. The latency incurred by the pre-handoff control traffic should also be minimized. Pre-existing handoff connections between MSRs can significantly speed up I-TCP handoffs.

9 Example Applications

We have used I-TCP to improve the performance of TCP based applications running on mobile hosts in our indoor wireless LAN environment. Some of these applications are listed below along with a description of the changes needed for I-TCP:

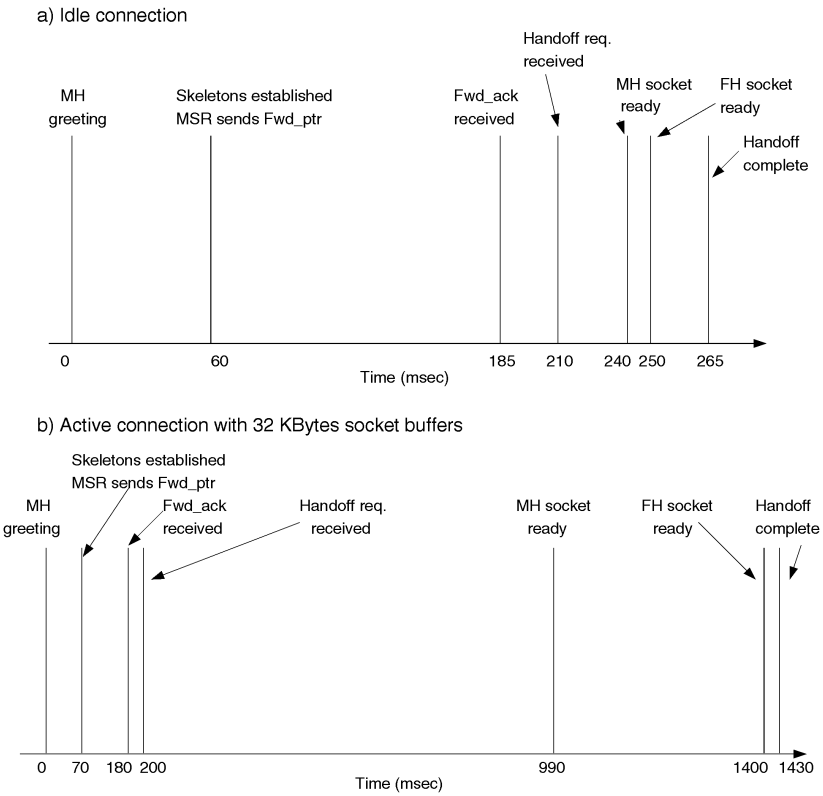


Fig. 14. Analysis of I-TCP handoff events.

1) **ttcp** benchmark—We used the **ttcp** benchmark to perform the handoff experiments reported in the previous section. The regular socket calls for **connect**, **listen**, **accept**, and **close** were replaced by their I-TCP equivalents. Throughput experiments conducted using the **ttcp** benchmark comparing the performance of I-TCP with that of regular TCP have been reported earlier in Section 4.
2) **ftp**—**ftp** uses two kinds of TCP connections—one for control and the other for data. The control connection is used to exchange application layer control messages between the **ftp** client and the server while a separate data connection is opened for every file transfer. We developed a hybrid **ftp** implementation called *i*-ftp to use I-TCP for data connections but regular TCP for control connections. Using I-TCP for data connections gives us throughput advantages over regular TCP while using regular TCP for the control connection gives us protection against MSR failures which may cause loss of I-TCP (data) connections.
3) **Chimera** WWW browser—Chimera,[3] which is a *World Wide Web (WWW)* browser, uses the Hypertext Transfer Protocol (HTTP) [15] to access WWW services over the Internet. HTTP itself uses TCP to transfer data. We modified Chimera sources to use I-TCP instead of TCP for data transfer at mobile clients, which considerably improved the time needed to download large files.

We mentioned earlier that I-TCP is not particularly suited for applications such as **telnet** which rely heavily on end-to-end reliability of TCP. An additional end-to-end latency incurred by I-TCP connections (about 20 ms) could also be significant for **telnet**. On the other hand, **telnet** performance in the presence of moves could improve with the use of I-TCP, since lost segments during cell crossovers will force regular TCP into retransmission timer backoff.

10 Related Work

Cáceres and Iftode [7] demonstrated in their experiments that TCP throughput deteriorates rapidly when a mobile host switches cells at a constant rate. Their experiments showed that the loss of throughput was much higher when there was a loss of link layer connectivity during the cell crossover than with overlapped wireless cells. The main reason for the loss of throughput was the packet loss accompanying cell crossovers, which triggers congestion control at the sending host. The loss of throughput can also occur because of packets lost over the wireless medium due to bit errors. Fast retransmission [14] can be used over wireless links to recover from packet loss due to occasional wireless errors. Cáceres and Iftode [7] also suggested sending duplicate acknowledgments from a mobile host after a move to trigger fast retransmission at the sending host so that the data rate over a TCP connection can be quickly brought to the normal speed. Such modifications are not adequate to recover from multiple losses per window because the transmitting host still performs a *slow start* if more than one segment is lost per window, thus limiting the effective throughput. Further, sending duplicate acknowledgments from the mobile host at a time when the fixed network is congested, can exacerbate the congestion.

3. Chimera was developed by John Kilburg.

Link layer retransmission (LLR) can bring the error rate of error-prone wireless links on par with that on the wired networks. Such an approach however, interferes with the end-to-end retransmissions of TCP and does not result in improved performance [16] for certain loss characteristics. A somewhat better approach that uses retransmissions on the (last) wireless link was suggested in [17], which falls somewhere between LLR and our approach. In this method, a snoop layer at the base station (MSR) caches TCP segments sent by a (fixed) sending host to the mobile host. The snoop layer also intercepts acknowledgments flowing from the MH to the sending host. If the acknowledgments (e.g., duplicate ACKs) indicate that a TCP segment was lost over wireless, the lost segment is retransmitted over the wireless link from the base station. The snoop layer approach has an advantage over indirect TCP in that the end-to-end semantics of TCP are preserved. However, any method that attempts to correct for wireless errors locally *without* the knowledge of the sending host can help very little if the sending host on the fixed network times out while the retransmission mechanism on the wireless link is trying to get a data packet to the mobile host. In contrast, the MSR in I-TCP acknowledges TCP segments received from the fixed sending host promising to deliver those to the mobile host as and when the conditions over the wireless link are favorable. The sending host is thus isolated from loss of data over the wireless link.

Experiments with split TCP [18] have shown performance improvement over regular TCP where smaller MTU is used over the wireless part of the split connection. A similar scheme to connect mobile hosts to the Internet using digital cellular network has been described in [19]. Semiconnected TCP [20] provides a split TCP connection only when a mobile host is roaming away from its home network. In this scheme, the mobile host uses end-to-end TCP when it is connected to its home network. The same TCP connection is however split at a mobility support gateway (MSG) when the MH moves away.

11 Conclusions and Future Work

We have shown that *indirection* or mediation by mobility support routers (MSRs), can be used as a robust approach to improve transport layer performance in a mobile wireless environment. Our approach first confines the mobility related performance problems to the wireless link and then attempts to alleviate such problems by adapting the transport layer protocols on the wireless link in a way that requires no modifications to the hosts on the fixed network. We designed and implemented I-TCP, a TCP compatible indirect protocol, which is particularly suited for throughput intensive applications. Experiments with I-TCP on our testbed showed greatly improved throughput in comparison to regular TCP under simulated mobility conditions and wireless losses. The performance improvement for wide-area connections was much higher than for local-area connections. We presented some measurements of the time needed for I-TCP handoffs as a function of the size of the state to be transferred. Our measurements show that operating system support in the form of suitable handoff mechanisms can help in achieving efficient transport layer handoffs. We also analyzed the handoff data to determine the time consuming activities in I-TCP handoffs. A kernel resident implementation of I-TCP, though less flexible than a user level implementation such as ours, should further cut down on the copying overhead incurred by I-TCP connections.

Attempts are under way to port I-TCP to the mobile IP standard being developed by the Mobile IP working group of the Internet Engineering Task Force (IETF) [11]. To fully realize the potential of the indirect model at the transport layer, we are also planning to develop a flexible and light-weight transport protocol for the wireless side of I-TCP which can adapt to changes in the wireless environment and can support voluntary disconnections. Such a wireless protocol can be optimized with the knowledge that it will be used only on one wireless hop.

Acknowledgments

A preliminary version of this paper appeared in [1]. Some of the I-TCP performance numbers reported earlier in [2] and [3] have been included in this paper as well for completeness. This research work was supported in part by ARPA under contract number DAAH04-95-1-0596, the U.S. National Science Foundation under grant numbers CCR 95-09620 and IRIS 95-09816, and the sponsors of WINLAB. This work was done when Ajay V. Bakre was a graduate student in the Department of Computer Science at Rutgers University.

References

[1] A. Bakre and B.R. Badrinath, "Handoff and System Support for Indirect TCP/IP," *Proc. Second USENIX Symp. Mobile and Location-Independent Computing*, pp. 11-24, Apr. 1995.

[2] A. Bakre and B.R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," *Proc. 15th Int'l Conf. Distributed Computing Systems*, May 1995.

[3] A. Bakre and B.R. Badrinath, "Indirect Transport Layer Protocols for Mobile Wireless Environment," *Mobile Computing*, T. Imielinski and H. Korth, eds., chapter 8, pp. 229-252. Kluwer Academic Publishers, 1996.

[4] A. Bakre, "Design and Implementation of Indirect Protocols for Mobile Wireless Environments," PhD dissertation, Rutgers Univ., 1996.

[5] J. Postel, "User Datagram Protocol," Request for Comments 768, Aug. 1980.

[6] J. Postel, "Transmission Control Protocol," Request for Comments 793, Sept. 1981.

[7] R. Cáceres and L. Iftode, "The Effects of Mobility on Reliable Transport Protocols," *Proc. 14th Int'l Conf. Distributed Computing Systems*, pp. 12-20, June 1994.

[8] B.R. Badrinath, A. Bakre, T. Imeilinski, and R. Marantz, "Handling Mobile Clients: A Case for Indirect Interaction," *Proc. Fourth Workshop Workstation Operating Systems (WWOS-IV)*, Oct. 1993.

[9] J. Ioannidis, "Protocols for Mobile Internetworking," PhD dissertation, Columbia Univ., 1993.

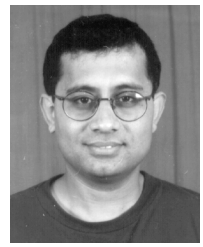
[10] J. Ioannidis, D. Duchamp, and G.Q. Maguire, "IP-Based Protocols for Mobile Internetworking," *Proc. ACM SIGCOMM*, pp. 235-245, Sept. 1991.

[11] "IP Mobility Support," C. Perkins, ed., Request for Comments 2002, Oct. 1996.

[12] M. Accetta, R. Baron, W. Bolosky, D. Golub, R. Rashid, A. Tevanian, and M. Young, "Mach: A New Kernel Foundation for UNIX Development," *Proc. USENIX 1986 Summer Conf.*, July 1986.

[13] S.J. Leffler, M.K. McKusick, M.J. Karels, and J.S. Quarterman, *The Design and Implementation of the 4.3 BSD UNIX Operating System*. Addison Wesley, 1989.
[14] V. Jacolbson, R. Braden, and D. Borman, "TCP Extensions for High Performance," Request for Comments 1323, May 1992.
[15] T. Berners-Lee, R. Fielding, and H. Frystyk, "Hypertext Transfer Protocol—HTTP/1.0," Internet Draft, Feb. 1996.
[16] A. DeSimone, M.C. Chuah, and O.C. Yue, "Throughput Performance of Transport-Layer Protocols Over Wireless LANs," *Proc. Globecom '93*, Dec. 1993.
[17] E. Amir, H. Balakrishnan, S. Seshan, and R. Katz, "Efficient TCP Over Networks with Wireless Links," *Proc. Fifth Workshop Hot Topics in Operating Systems (HoTOS-V)*, May. 1995.
[18] R. Yavatkar and N. Bhagwat, "Improving End-to-End Performance of TCP Over Mobile Internetworks," *Proc. IEEE Workshop Mobile Computing*, Dec. 1994.
[19] M. Kojo, K. Raatikainen, and T. Alanko, "Connecting Mobile Workstations to the Internet Over a Digital Cellular Telephone Network," *Proc. Mobidata Workshop Mobile and Wireless Information Systems*, Nov. 1984.
[20] J.S. Hansen, T. Reich, and B. Andersen, "Semi-Connected TCP/IP in a Mobile Computing Environment," *Proc. Information Visualization and Mobile Computing Workshop (IMC '96)*, Feb. 1996.

Ajay V. Bakre received his BE in electrical engineering from the University of Rajasthan, Jaipur, India, in 1986 and his MTech in computer and information rechnology from the Indian Institute of Technology, Kharagpur, India, in 1989. He completed his PhD from the Computer Science Department at Rutgers University in 1996. Dr. Bakre is currently a research staff member at NEC-USA Inc. C & C Research Labs in San jose, California. His research interests are mobile wireless computing, networking protocols, and distributed systems.



B.R. Badrinath is currently an associate professor in the Computer Science Department at Rutgers University. He is a co-principal investigator of the DataMan project at Rutgers University. His research interests include wireless networking, distributed systems, and wireless mobile computing. As part of his research, he is working on developing communication protocols for mobile hosts; protocols that can be integrated with existing internet protocols but are still optimized for wireless and mobility. He is also investigating issues in building environment awareness into operating systems designed for mobile hosts and disconnected operation in databases.