Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments

Ramón Cáceres and Liviu Iftode

Abstract— **We explore the performance of reliable data communication in mobile computing environments. Motion across wireless cell boundaries causes increased delays and packet losses while the network learns how to route data to a host's new location. Reliable transport protocols like TCP interpret these delays and losses as signs of network congestion. They consequently throttle their transmissions, further degrading performance. We quantify this degradation through measurements of protocol behavior in a wireless networking testbed. We show how current TCP implementations introduce unacceptably long pauses in communication during cellular handoffs (800 milliseconds and longer), and propose an end-to-end fast retransmission scheme that can reduce these pauses to levels more suitable for human interaction (200 milliseconds). Our work makes clear the need for reliable transport protocols to differentiate between motion-related and congestion-related packet losses, and suggests how to adapt these protocols to perform better in mobile computing environments.**

Keywords— **Wireless networks, cellular handoffs, congestion control, TCP, mobile IP.**

I. Introduction

RELIABLE transport protocols have been tuned for networks composed of wired links and stationary hosts. They adapt to prevailing end-to-end delay conditions throughout the life of a connection, and interpret unexpected increases in delay as packet losses caused by congestion. In response to perceived losses, protocols like the Transmission Control Protocol (TCP) [1] aggressively slow their transmissions to allow the network to recover. These congestion control policies have proven beneficial in improving the overall performance of networks like the Internet. The premise underlying these policies, that packet losses are largely due to congestion, is correct for existing networks.

Future networks, however, will include wireless links and mobile hosts. In particular, there will be local-area networks composed of wireless cells of a few meters in diameter. Such *microcellular* networks are desirable for three important reasons: they offer high aggregate bandwidth, they require low power from mobile transceivers, and they provide accurate location information. Users in microcellular environments will often carry hosts across cell boundaries without warning and in the midst of data transfers.

This work was performed at Matsushita Information Technology Laboratory.

Ramón Cáceres can be reached at AT&T Bell Laboratories, 101 Crawfords Corner Road, Holmdel, NJ 07733, USA, ramon@research.att.com.

Liviu Iftode can be reached at Princeton University, Department of Computer Science,Princeton, NJ 08544, USA, liv@cs.princeton.edu.

Transport-level connections will thus encounter types of delay and loss that are unrelated to congestion. First, communication may pause while the handoff between cells completes and packets can again be routed to and from the mobile host. Second, packets may be lost due to futile transmissions over the wireless network when a mobile host moves out of reach of other transceivers, especially in networks with little or no overlap between cells. Third, packets may be lost due to the relatively frequent transmission errors suffered by wireless links. Some performance degradation due to these delays and losses is unavoidable.

These events also trigger congestion control procedures that further degrade performance. In particular, TCP implementations continually measure how long acknowledgements take to return. They maintain a running average of this delay and an estimate of the expected deviation in delay from the average. If the current delay is longer than the average by more than twice the expected deviation, TCP assumes that the packet was lost. In response, TCP retransmits the lost packet and initiates congestion control procedures to give the network a chance to recover [2]. First, TCP drops the transmission window size to reduce the amount of data in transit through the network. Second, it activates the slow-start algorithm to restrict the rate at which the window grows to previous levels. Third, it resets the retransmission timer to a backoff interval that doubles with each consecutive timeout.

When motion is mistaken for congestion, these procedures result in significant reductions in throughput and unacceptable interactive delays for active connections. The degradation is readily apparent, for example, to users of emerging ubiquitous computing environments [3].

This paper quantifies the effects of motion on throughput and delay, identifies the factors that contribute to the loss of performance, and suggests an end-to-end approach for alleviating the problem. It shows how waits for TCP's retransmission timeouts cause pauses in communication that last 0.8 seconds and longer after each cell crossing. Other researchers have called attention to the long pauses caused by TCP's exponential backoff policy [4][5][6], but to our knowledge this is the first systematic treatment of this problem. This paper also describes how using TCP's fast retransmission procedure can reduce these pauses to 0.2 seconds. We focus on TCP because it is the most widely used reliable transport protocol and will be used in at least the first generation of mobile computing environments. Furthermore, lessons learned from TCP apply to other reliable transport protocols that must deal with both mobility and congestion.

The remainder of this paper is organized as follows. Section II describes the wireless networking testbed used to obtain our results. Section III presents the measured effects of host motion on the performance of reliable transport protocols. Section IV proposes and evaluates an end-to-end approach to alleviating the negative effects of motion. Section V discusses wireless transmission errors as an area for future work, and Section VI concludes the paper.

II. Wireless networking testbed

We explore the effects of mobility through measurements of transport protocol behavior in a wireless networking testbed. The testbed consists of mobile hosts (MH), mobility support stations (MSS), and stationary hosts (SH) deployed in an ordinary office environment. Mobile hosts connect to a 2-Mbit/second WaveLAN local-area wireless network. WaveLAN is a direct-sequence spread spectrum radio product from NCR. Stationary hosts connect to a 10-Mbit/second Ethernet local-area wired network. Mobility support stations connect to both networks. Figure 1 shows the minimum testbed configuration.

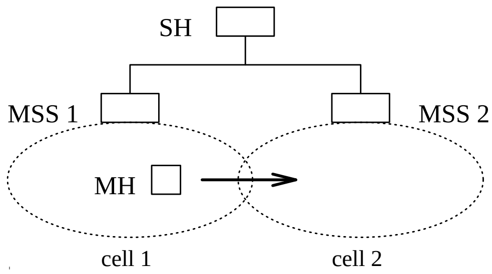

Fig. 1. Wireless networking testbed

All hosts and support stations are equipped with 50-MHz i486 processors, 330-Mbyte hard disks, 16 Mbytes of memory, and the necessary network interface hardware. They run the 4.3BSD-Tahoe version of TCP from the University of California at Berkeley, Mobile IP software from Columbia University [5], and the Mach 3.0 microkernel and Unix server (MK77/UX37) from Carnegie Mellon University [7]. 4.3BSD-Tahoe TCP is widely used throughout the Internet and implements exponential retransmission backoffs and the slow-start algorithm.

A. Cellular handoff procedures

Each MSS defines one cell and is responsible for the MHs in its cell. It acts as the default gateway for those MHs, routing packets that originate in an MH from the wireless to the wired part of the network. Similarly, it forwards packets destined to an MH from the wired to the wireless part of the network.

MHs and MSSs collaborate to perform handoffs between cells. MSSs make their presence known by broadcasting periodic beacons over the wireless network. An MH decides to switch cells when it receives a beacon from a new MSS with a stronger wireless signal than the beacon from the old MSS, or when it receives the first beacon from a new MSS after failing to receive beacons from the old MSS.

To switch cells the MH sends a greeting packet to the new MSS, and changes its own routing tables to make the new MSS its default gateway. It also notifies the new MSS of the identity of the old MSS. The new MSS acknowledges the greeting to the MH, adds the MH to the list of MHs for which the new MSS is responsible, and begins to route the MH's packets accordingly. The new MSS also informs the old MSS that the host has moved and can be reached through the new MSS. The old MSS then adjusts its routing tables in order to forward to the new MSS any packets that arrive for the MH, and acknowledges the handoff to the new MSS. Finally, the new MSS acknowledges the completion of the handoff to the MH. Further details of this protocol are found in [5].

B. Methodology

In our experiments, we initiate a reliable data transfer over a TCP connection between an MH and an SH, we cause the MH to cross cell boundaries while the connection is active, and we measure the performance of the connection.

We simulate motion across cell boundaries in software. The MH in our testbed is always in range of both MSSs, but we modified the Mobile IP software on the MH to ignore beacons from all but one MSS. After the MH spends a specified number of beaconing periods in that MSS's cell, the modified software listens for a beacon from the other MSS in order to initiate handoff procedures with the new MSS.

An important benefit of simulating motion in software is that it lets us study networks with overlapping cells as well as networks with non-overlapping cells. When adjacent cells overlap and an MH is in the region of overlap, packets can continue to flow between the MH and the old MSS while the handoff to the new MSS is in progress. When cells do not overlap, there is an unavoidable pause in network-level communication while the MH is out of reach from the old MSS and the handoff to the new MSS has not yet completed. The testbed allows us to explore the full range of handoff scenarios, from the case when the MH is in contact with both MSSs throughout the handoff, to the case when the MH cannot communicate with any MSS for an arbitrary interval of time after it leaves the old cell.

Another benefit of simulating motion in software is that it gives us precise control over the instant when handoffs begin. Under normal circumstances, handoffs begin at indeterminate times based on the time remaining in a cell's beaconing period when a host enters the cell, or on the relative strengths of two wireless signals. Our testbed makes this process deterministic and therefore allows us to reliably reproduce test conditions. Finally, simulating motion in software eliminates the need to physically move test equipment during experiments.

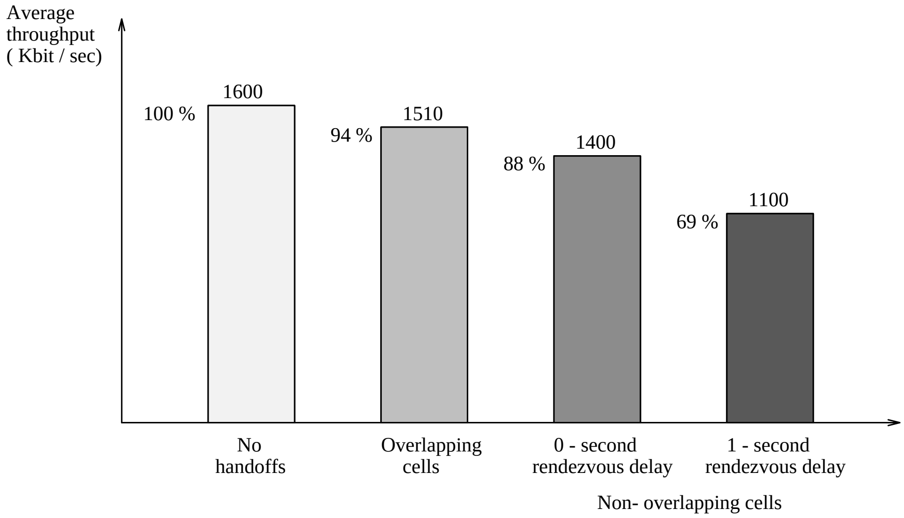

Fig. 2. Loss of throughput due to host motion

III. The effects of motion

We ran a number of experiments in the manner described above. We found that throughput dropped significantly in the presence of motion. We then analyzed the problem in more detail to determine the causes of the performance loss. We tracked the TCP sequence number and window size over the lifetime of a connection to determine how TCP behaved during handoffs. We also traced TCP and Mobile IP packets during the course of each handoff to determine if any packets were lost and why. This section presents our results.

Due to space limitations, we only present results for the case where data packets flow from the MH to the SH and acknowledgement packets flow from the SH to the MH. However, we also ran our experiments for the opposite case, with very similar results. We summarize our results for both cases in Section IV-D.

A. Loss of throughput

Figure 2 shows the average application-level throughput achieved when transferring 4 Mbytes of data between an MH and an SH. From left to right, the vertical bars represent the throughput obtained under four scenarios:

- The MH does not move.
- The MH moves between overlapping cells.
- The MH moves between non-overlapping cells and receives a beacon from the new MSS at the instant it leaves the old cell (0-second rendezvous delay).
- The MH moves between non-overlapping cells and receives a beacon from the new MSS one second after leaving the old cell (1-second rendezvous delay).

In the scenarios that involve motion, the beaconing period is 1 second and the MH switches cells every 8 beaconing periods. These parameters were chosen to allow TCP connections to attain maximum throughput between handoffs while also allowing us to observe multiple handoffs during a single data transfer.

We believe these four scenarios show a complete and fair picture of the problems introduced by host motion. We use the no-motion scenario as a base for comparison. The motion scenario with overlapping cells represents the best handoff performance possible with our hardware and software. It is realizable in a real network only if overlap regions are large enough, and hosts move slowly enough, for handoff operations to complete while a moving host is still in the overlap region. The scenario with zero rendezvous delay represents the minimum network-level interruption introduced by non-overlapping cell handoffs. It is realizable only if the MH does not have to wait for a beacon before it can communicate with the new MSS, for example in a network where MSSs announce their presence by means of a continuous signal. Finally, the scenario with a 1-second rendezvous delay shows what happens as the length of network-level interruptions increases. It is a realistic scenario when a periodic beaconing scheme is used, since an MH may have to wait up to a full beaconing period before it receives a beacon from the new MSS.

As shown in Figure 2, throughput degrades substantially in the presence of motion across non-overlapping cells. In the overlapping cell scenario, throughput degrades only slightly, by 6%. In the non-overlapping cell scenario with zero rendezvous delay, throughput drops by 12% even though only 3 handoffs occur in the roughly 24-second lifetime of the connection. Throughput drops much further with a 1-second rendezvous delay, by 31% with 3 handoffs in roughly 29 seconds.

In the rest of this section we study the causes of this performance degradation in increasing detail. We concentrate on single handoffs to eliminate from our results any dependencies on the parameters of the throughput test discussed above (4 Mbytes of data with handoffs every 8 seconds). Our results will thus apply to all cell handoffs in each motion scenario.

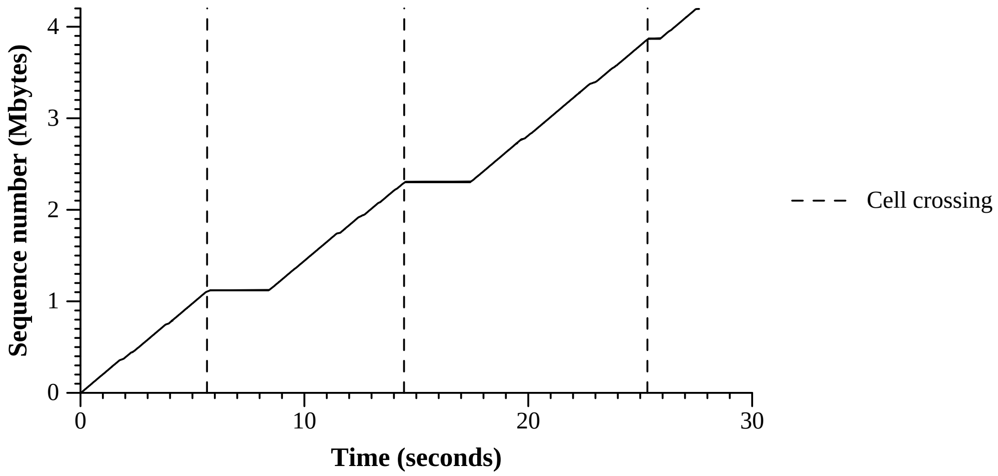

Fig. 3. Behavior of TCP sequence number in response to cell boundary crossings

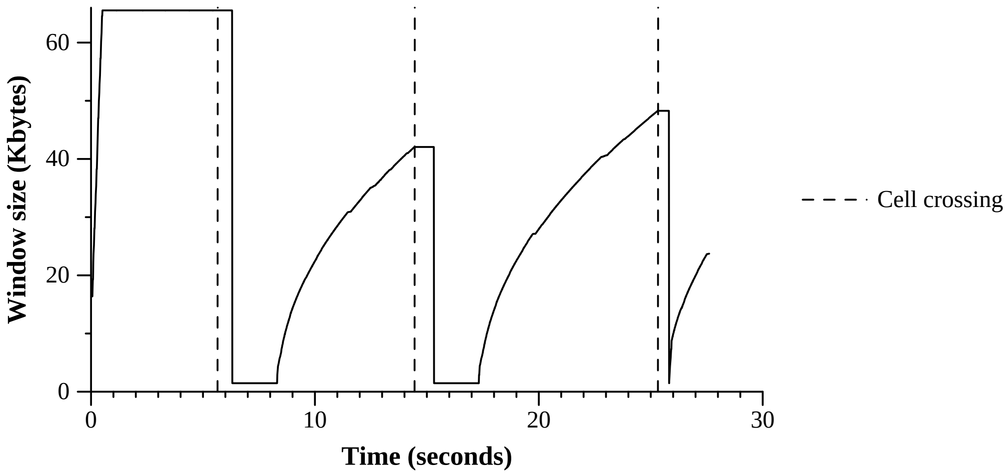

Fig. 4. Behavior of TCP congestion window in response to cell boundary crossings

B. Pauses in communication

Figure 3 shows how the TCP sequence number behaves over the life of a connection. In this example, the MH moves between non-overlapping cells with a 1-second rendezvous delay. As shown, the sequence number ceases to advance for roughly 3 seconds after the first two cell crossings, and for roughly 1 second after the last crossing. A 3-second pause is typical of a 1-second rendezvous delay, while a 1-second pause is more typical of a 0-second rendezvous delay. During these pauses, TCP transmits no new data and transport-level communication comes to a halt.

The effect is also visible in Figure 4, which graphs the TCP congestion window over the life of the same connection. The congestion window is an upper bound on the transmission window, which in turn controls how much unacknowledged data a TCP connection can have in transit over the network. As shown, the congestion window stops growing with every cell crossing. Some time after the crossing, the window shrinks to its minimum value and eventually begins to grow again. The intervals between when the window stops growing and when it begins to grow again correspond to the 3-second and 1-second pauses in communication noted above.

C. Packet losses

The long pauses in communication are caused by TCP's response to packet losses. Losses occur due to routing inconsistencies during non-overlapping cell handoffs. Consider the route from the MH to the SH. When the MH leaves a cell without warning, its routing tables continue to point to the old MSS as the default gateway. The MH does not know it has moved and therefore does not change its routing tables until a beacon arrives from the new MSS. Until then, the MH continues to send packets destined for the SH directly to the old MSS. These packets are lost because the MH can no longer reach the old MSS through the wireless interface.

Inconsistencies persist longer with the route from the SH to the MH. The old MSS does not know that the MH has left the cell until an explicit notification arrives from the new MSS, which cannot send the notification before it receives a greeting from the MH. Until the old MSS learns of the MH's motion, it continues to route packets directly to the MH. These packets are also lost because the old MSS can no longer reach the MH. Any other parts of the network involved in the handoff must also wait for explicit

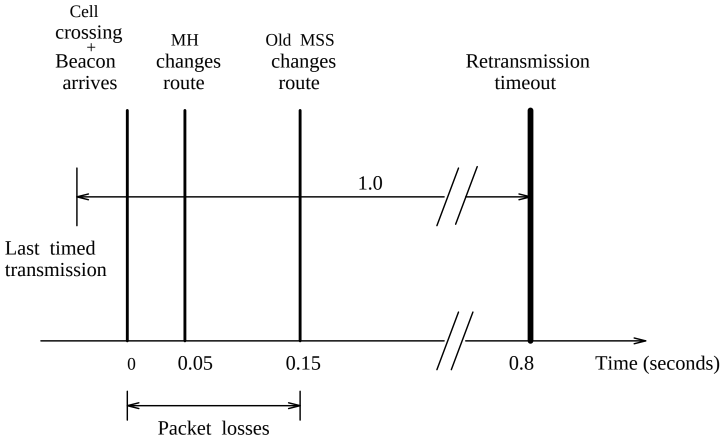

Fig. 5. Handoff latency and related packet losses with a 0-second rendezvous delay

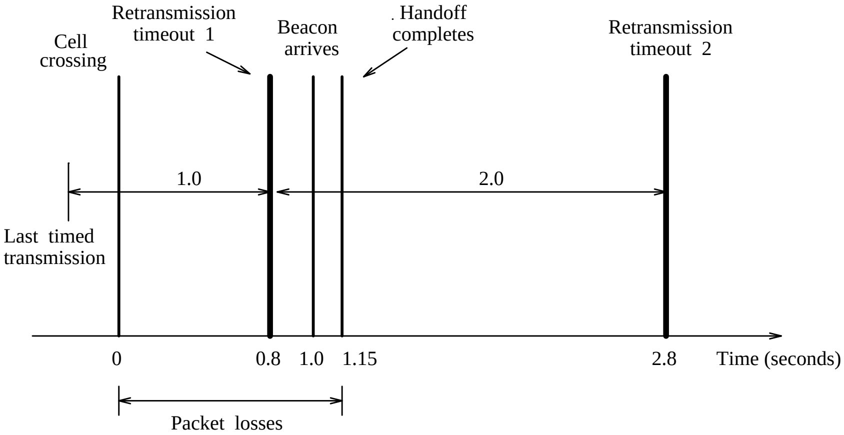

Fig. 6. Handoff latency and related packet losses with a 1-second rendezvous delay

notification that the MH has moved before they can change their routing tables to point away from the old MSS to the new MSS.

Figure 5 shows what happens during one handoff in the case of zero rendezvous delay. Although the beacon from the new MSS arrives concurrently with the cell crossing, the MH's routing tables do not point to the new MSS until 0.05 seconds after the cell crossing. Similarly, the old MSS's routing tables do not point to the new MSS until 0.15 seconds after the cell crossing. Although the system overhead implicit in these figures can be reduced through careful implementation, handoff latency cannot be altogether eliminated because at least two packet exchanges are needed to notify both the new MSS and the old MSS that the MH has changed cells. Because these packets incur unavoidable propagation delays, there will always be a window of opportunity during which both data and acknowledgement packets can be routed to unreachable wireless transceivers.

An active TCP connection thus loses up to a full transmission window's worth of packets and related acknowledgements during each handoff. Once the transmission window fills, communication stops until the retransmission timer expires. When a timeout occurs, TCP retransmits the earliest unacknowledged packet, doubles the retransmission interval, and resets the timer. If the handoff is not yet complete when the timeout occurs, the retransmitted packet is also lost and TCP waits for yet another timeout before retransmitting. A single timeout is typical of zero rendezvous delay, as shown on Figure 5. Two consecutive timeouts are typical of a 1-second rendezvous delay, as shown on Figure 6.

It is evident how waits for retransmission timeouts freeze transport-level communication for 0.8 seconds or more with each cell crossing across non-overlapping cells, and are responsible for a large part of the throughput losses reported earlier. In contrast, handoffs between overlapping cells do not cause the same long pauses in communication because the implementation of overlapping cells in our testbed insures that no packets are lost during those handoffs. The slight throughput losses reported earlier for the overlapping cell scenario are due only to encapsulation and forwarding delays during handoffs.

D. *Slow recovery*

As shown in Figure 4, the congestion window drops abruptly after a cell crossing when the retransmission timer goes off, but returns only gradually to its previous level once transport-level communication resumes. TCP's slow-start algorithm [2] is responsible for this behavior. As acknowledgements reach the TCP transmitter, slow start first grows the congestion window exponentially until it reaches a threshold, then grows it linearly. The threshold is set to one half of the window size at the time of the retransmission timeout. The slow start threshold thus decays exponentially with consecutive timeouts.

The slow recovery after each handoff contributes to the loss of throughput discussed earlier, but only moderately. Our measurements show that the algorithm throttles transmissions for approximately 1 second after communication resumes. At that point the connection again reaches its maximum throughput (1.6 Mbit/second), and the congestion window ceases to affect performance.

E. Unacceptable interactive response

Interactive delays are a concern in addition to throughput. Studies of human factors indicate that people perceive interactive response to be "bad" if it takes longer than 100 to 200 milliseconds [8]. As discussed above and shown in Figures 3, 4, 5, and 6, transport-level communication comes to a halt for 800 milliseconds or longer after non-overlapping cell crossings. Furthermore, these pauses grow exponentially with growing rendezvous delays due to TCP's exponential retransmission backoff policy. In interactive applications that use TCP for reliable data transport, user inputs and their responses will be unable to travel between mobile hosts and remote servers during these pauses.

Although users may not always interact with their computers while moving, there will certainly be times when they will do so soon after stopping. Our results show that pauses will persist from 650 milliseconds to several seconds after a host enters a new cell and the handoff completes. Motion will thus lead to unacceptable interactive response unless we solve the problems presented in this section.

IV. Alleviating the effects of motion

Our results demonstrate that we must improve the performance of reliable transport communication in mobile computing environments. Two approaches are possible: hiding motion from the transport level, and adapting the transport level to react better to motion.

A. Smooth handoffs

Cellular networks should strive to provide smooth handoffs in order to eliminate packet losses during cell crossings and thus hide motion from the transport level. As we have shown with our testbed, one way to achieve this goal is to implement "make then break" handoffs and to engineer enough overlap between cells to insure that handoffs complete before an MH loses contact with the old MSS. However, there are compelling reasons to build networks with little or no overlap between small cells:

- They offer high aggregate bandwidth because they can use the same portion of the electromagnetic spectrum in nearby cells. Bandwidth is scarce in wireless networks.
- They support low-powered mobile transceivers because signals need only reach short distances. Mobile computers have stringent power consumption requirements.
- They provide accurate location information because cells are small and sharply defined. Location information adds important functionality to distributed systems.

It is possible to provide smooth handoffs in spite of packet losses due to motion between non-overlapping cells. For example, MSSs could buffer packets they have recently sent to MHs. When an MSS is notified that an MH has moved out of the MSS's cell, the MSS can send the buffered packets for that MH to the MSS now responsible for the MH. The new MSS can in turn forward the packets to the MH. This technique increases the memory requirements of the MSSs, but may prove feasible because the amount of data that an MSS needs to buffer is bounded by the maximum handoff latency between adjacent cells.

However, it is unlikely that all cellular networks will provide perfectly smooth handoffs in the near future. It is therefore worthwhile to investigate transport-level techniques for alleviating the effects of packet losses during handoffs.

B. More accurate retransmission timers

The long pauses in communication presented in Section III are due partly to inaccurate retransmission timers. TCP implementations historically have used coarse timers with a 300- to 500-millisecond resolution. For example, the 4.3BSD-Tahoe implementation in our testbed uses a 500-millisecond resolution timer. The resulting minimum timeout value is twice the timer resolution, or 1 second (this 1-second value is evident in Figures 5 and 6). The retransmission timer is intended to track the round-trip delay experienced by a TCP connection, but actual round-trip delays are much smaller than 500 milliseconds. For example, connections in our testbed experience well under 1 millisecond of round-trip delay. It may appear that changing TCP implementations to use higher-resolution timers would result in more accurate round-trip time estimates and would thus reduce pauses in communication during cellular handoffs.

However, more accurate timers will not solve the problems introduced by motion across wireless cell boundaries. A timer that successfully tracks the round-trip delay will lead to timeout values on the order of 1 millisecond or less. These small timeout values will result in multiple timeouts while a handoff completes, which in turn will lead to the following three problems:

- Multiple reductions of the slow-start threshold. The threshold decays exponentially with consecutive timeouts and can quickly reach the minimum window size of one packet. When communication resumes after a handoff, connections will find themselves in the linear region of window growth dictated by the slow start algorithm, and will take many round-trip times before they reach maximum throughput. Our testbed avoided this problem because of its coarse timers.

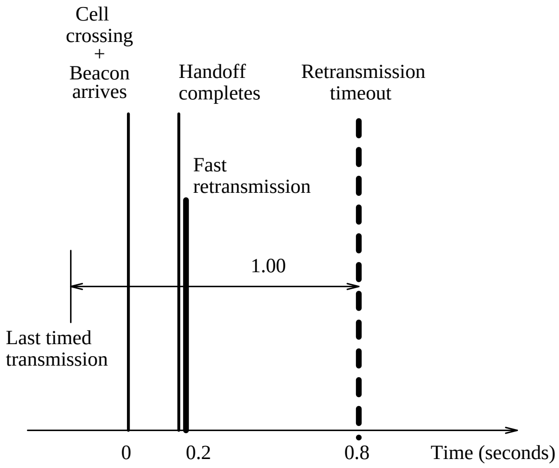

Fig. 7. Fast retransmission after a handoff with a 0-second rendezvous delay

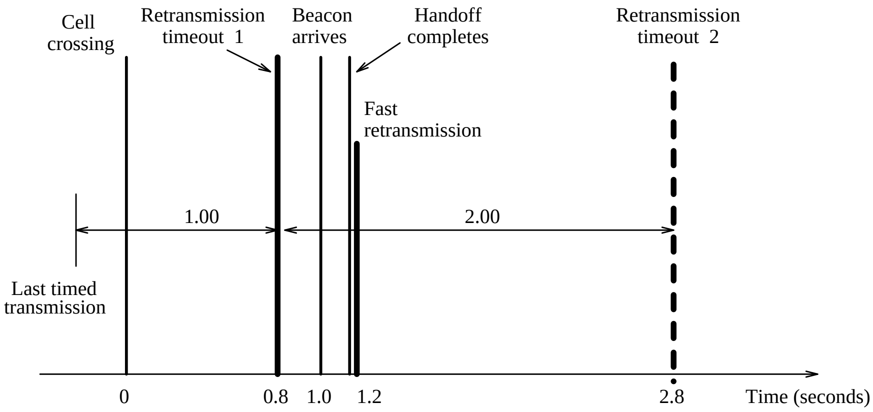

Fig. 8. Fast retransmission after a handoff with a 1-second rendezvous delay

- Multiple backoffs of the retransmission timer. Backoffs grow exponentially with consecutive timeouts and can quickly lead to the long pauses in communication we are trying to avoid.
- Multiple retransmissions before the routes become consistent. These futile retransmissions waste bandwidth in the slow wireless medium.

In general, it is difficult for a timer-based scheme to adapt to the abrupt changes in round-trip delay introduced by cellular handoffs.

C. Fast retransmissions

An attractive end-to-end solution [9] to the problems presented in Section III is for the transport protocol to resume communication immediately after handoffs complete, without waiting for a retransmission timeout. Modern TCP implementations, including the 4.3BSD-Tahoe implementation in our testbed, already perform similar *fast retransmissions* when a transmitter receives triplicate acknowledgements from a remote receiver. When activated, the fast retransmission procedure immediately retransmits the earliest unacknowledged packet, drops the transmission window, and initiates the slow-start algorithm. The rationale behind current fast retransmissions is that triplicate acknowledgements clearly indicate that packet loss has occurred, and thus there is no need to wait for a timeout before retransmitting.

We made modest changes to the TCP and Mobile IP software in our testbed to invoke the existing fast retransmission procedure as soon as routes become consistent following a cell crossing. First, the Mobile IP software on the MH signals the TCP software on the MH when a greeting acknowledgement arrives from the new MSS. Second, the TCP transmitter on the MH invokes the fast retransmission procedure when it receives such a signal. The signal is delivered through shared memory between TCP and IP software in the same host.

Figure 7 shows the measured effect of fast retransmissions after a non-overlapping cell handoff with a 0-second rendezvous delay. As shown, fast retransmissions cause a TCP connection to resume communication 50 milliseconds after the handoff completes. In contrast, the retransmission timeout would not have occurred until 650 milliseconds after the handoff completed.

An additional communication step is necessary to inform the TCP software on the SH of the events occurring at the other end of the connection. First, the Mobile IP software on the MH signals the TCP software on the MH of the completion of the handoff, as described above. Second, the TCP software on the MH forwards the signal over the

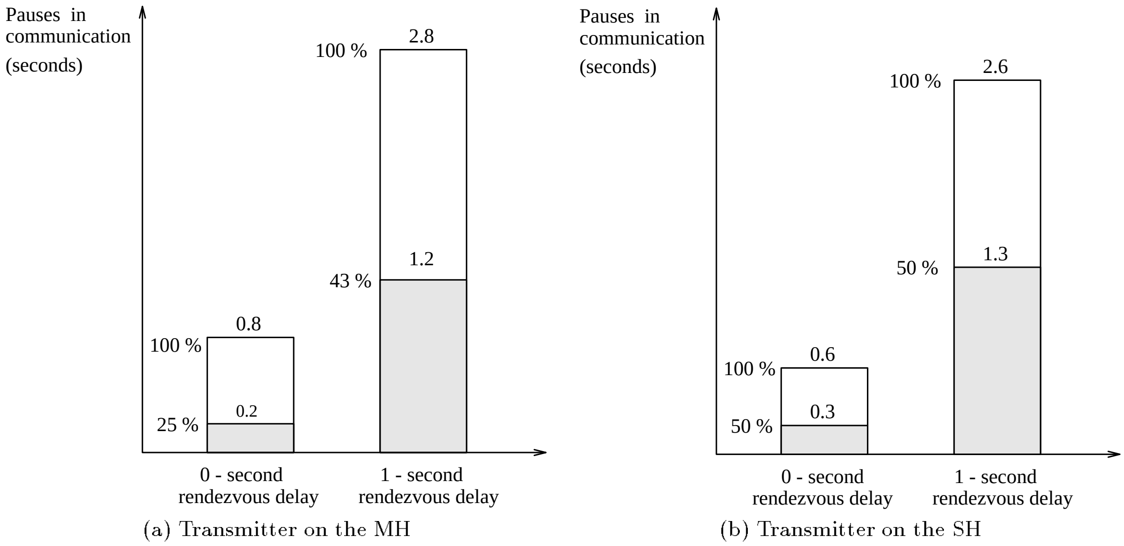

Fig. 9. Improvements in latency due to fast retransmissions

network to the SH. Third, the TCP software on the SH invokes the fast retransmission procedure when it receives such a signal. The signal travels from the MH to the SH through normal IP routes and can take either of two forms: It can be a specially marked TCP acknowledgement packet containing the sequence number of the last data packet successfully received by the MH, or it can be three identical but ordinary TCP acknowledgement packets. The triplicate acknowledgement approach consumes more resources but does not require modifications to TCP implementations on stationary hosts.

Figure 8 shows the measured effect of fast retransmissions after a non-overlapping cell handoff with a 1-second rendezvous delay. As shown, fast retransmissions again causes a TCP connection to resume communication 50 milliseconds seconds after the handoff completes. In contrast, the retransmission timeout would not have occurred until 1,650 milliseconds after the handoff completed.

The fast retransmission approach has three desirable features:

- It requires minimal changes to software on the end hosts. It changes Mobile IP only to propagate an end-of-handoff signal one layer up in the protocol hierarchy. It changes TCP only to invoke the existing fast retransmission procedure when the end-of-handoff signal arrives. It need not change TCP on stationary hosts if triplicate acknowledgements are used.
- It does not depend on special support from the network, including mobility support stations or other intermediate routers. It therefore does not depend on any one mobile networking environment and will work over an internetwork.
- It follows established congestion avoidance policies by closing the transmission window and using the slow-start algorithm after the initial retransmission. It thus avoids congesting the cell the MH has just entered. Gently probing the congestion state of a new route, such as the route to a new cell, is one of the principal motivations behind the slow start algorithm.

It is important to note that there is no need to initiate fast retransmissions in networks that guarantee smooth handoffs, that is, in networks that never lose packets during handoffs. In that case, the MH software involved in the handoff need not signal the transport level when handoffs complete. The fast retransmission scheme therefore coexists with any handoff scheme. The software that implements the scheme resides in the transport level and is exercised only when needed.

D. *Improvements in latency*

Figure 9 shows the pauses in transport-level communication caused by motion across non-overlapping cell boundaries, together with the improvements gained by applying the fast retransmission procedure. As shown, when the transmitter resides on the MH, fast retransmissions reduce these pauses from 0.8 to 0.2 seconds for a 0-second rendezvous delay, and from 2.8 to 1.2 seconds for a 1-second rendezvous delay.

Figure 9 also shows our results for the case when the TCP transmitter resides on the SH, where pauses drop to from 0.6 to 0.3 seconds for 0-second rendezvous delays, and from 2.6 to 1.3 seconds for 1-second rendezvous delays. Pauses before the improvements are shorter when the transmitter is on the SH (e.g., 0.6 vs. 0.8 seconds for 0-second rendezvous delays) because data packets incur added propagation delay before they are lost. Effectively, lost packets are sent earlier before the cell crossing, and thus retransmission timeouts occur earlier after the crossing. Pauses after the improvements are longer when the transmitter is on the SH (e.g., 0.3 vs. 0.2 seconds for 0-second rendezvous delays) because the fast retransmission must wait for an acknowledgement packet to travel between the MH and the SH after the handoff completes.

The fast retransmission scheme thus succeeds in reduc-

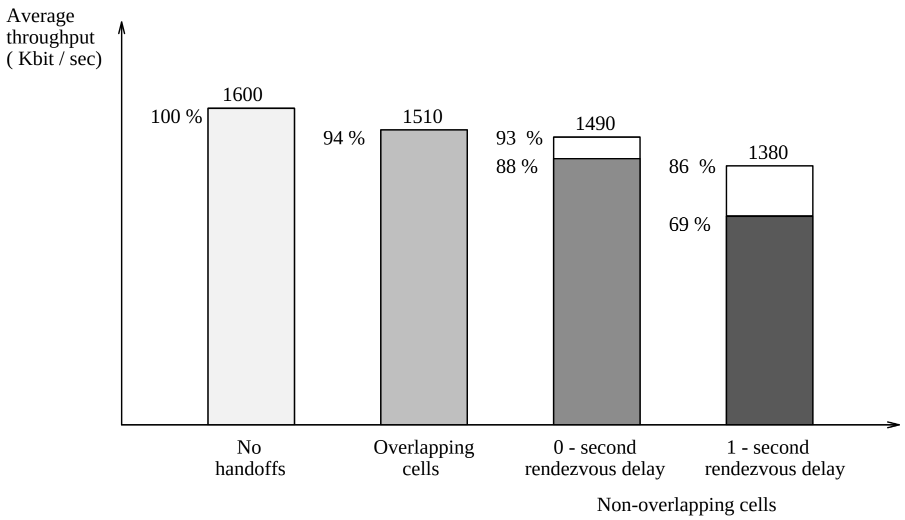

Fig. 10. Improvements in throughput due to fast retransmissions

ing interactive delays to 200-300 milliseconds beyond the rendezvous. Reducing handoff latency through careful implementation would further reduce this remaining delay. The Mobile IP software in our testbed is an early example of support for mobile networking and was not written with fast handoffs in mind. For example, it incurs substantial system overhead by employing application-level processes to process beacons, change routes, and perform other handoff-related functions. A more efficient implementation of handoffs combined with fast retransmissions should in all cases bring pauses in communication to 100 milliseconds or less after the rendezvous. If users do not attempt to interact with their mobile computers until they stop moving across cell boundaries, interactive delays will then drop to acceptable levels.

E. Improvements in throughput

We also measured significant improvements in throughput due to the fast retransmission scheme. As shown in Figure 10 for the test described in Section III-A, throughput improves from 1400 to 1490 Kbit/second for 0-second rendezvous delays, and from 1100 to 1380 Kbit/second for 1-second rendezvous delays. Some throughput losses remain because a transport-level scheme like fast retransmissions does not reduce network-level delays and packet losses, and because the slow-start algorithm throttles connections for some time after transport-level communication resumes.

V. Wireless transmission errors

Even in the absence of motion, the WaveLAN network in our testbed suffers from relatively frequent packet losses due to physical transmission errors. A separate measurement study found that WaveLAN exhibited excellent packet capture rates (over 99%) in an indoor environment [10]. However, in our environment, packet loss frequency varies widely even across short distances and depends on such factors as the positions of antennas in a room. Such problems are common in wireless communication because wireless media are vulnerable to ambient noise and multipath interference. Commonly cited bit error rates for radio and infrared links are 10^{-6} or worse, compared to 10^{-12} or better for fiber optic links.

Wireless transmission errors will also trigger the transport-level problems described in Section III. One possible solution is for the link-layer protocol that controls wireless links to retransmit packets lost on those links and thus hide the losses from higher layers. However, recent research shows that, under certain packet loss conditions, competing retransmission strategies in the link and transport layers can interact to reduce end-to-end throughput while increasing link utilization [11]. Alternative techniques such as selective retransmissions at the transport layer may prove more effective than link-layer retransmissions.

We wanted to isolate the effects of motion across cell boundaries from the effects of wireless transmission errors. We solved the problem by positioning the WaveLAN antennas physically close together in an area relatively free from ambient radiation and multipath problems. Packet losses in the absence of cell crossings then dropped to negligible levels. We also repeated all our handoff experiments using a wired network to emulate a wireless network; we substituted a second Ethernet for the WaveLAN in our testbed and found no fundamental differences in our results. We did not treat transmission errors any further in order to concentrate on handoffs. Nevertheless, the impact of wireless transmission errors on reliable transport protocols warrants further study.

VI. Conclusions

Mobility changes important assumptions on which existing systems operate. In particular, networks that include wireless links and mobile hosts suffer from delays and packet losses that are unrelated to congestion. Current reliable transport protocols react to these delays and losses by abruptly slowing their transmissions, a response that further degrades the performance of active connections. We have identified the factors that contribute to this performance degradation and have quantified their effects in detail. We have shown how waits for retransmission timeouts cause pauses in communication at least 650 milliseconds longer than the underlying network-level interruption. These pauses are readily noticed by interactive users and significantly reduce throughput.

We have also described a fast retransmission scheme that can reduce the pauses in communication to 50 milliseconds past the moment when transport-level communication resumes. Fast retransmissions thus reduce interactive delays to acceptable levels and regain much of the lost throughput. The fast retransmission approach is attractive because it calls for only minimal changes to end systems, relies on no special support from the underlying network or intermediate routers, follows established congestion avoidance procedures, and preserves end-to-end reliability semantics. The approach is thus applicable to a large and varied internetwork like the Internet.

Our work makes clear the need for reliable transport protocols to differentiate between motion-related and congestion-related packet losses. Our results can be used to adapt TCP to mobile computing environments. They also apply to other reliable transport protocols that must cope with both mobility and congestion.

Acknowledgments

Dan Duchamp and John Ioannidis provided the Mach 2.5 version of the Mobile IP software. Chuck Lewis helped to set up and maintain the testbed. Greg Minshall provided useful comments on an earlier draft of this paper.

References

[1] D. Comer, *Internetworking with TCP/IP*, Prentice Hall, Englewood Cliffs, New Jersey, 1988.
[2] V. Jacobson, "Congestion avoidance and control", *Proc. of ACM SIGCOMM '88*, Aug. 1988.
[3] S. Deering and M. Weiser, "Private communication", Xerox PARC, Oct. 1993.
[4] B. R. Badrinath, A. Bakre, T. Imielinski, and R. Marantz, "Handling mobile clients: A case for indirect interaction", *Proc. of IEEE WWOS-IV*, Oct. 1993.
[5] J. Ioannidis and G. Q. Maquire Jr., "The design and implementation of a mobile internetworking architecture", *Proc. of the USENIX 1993 Winter Conference*, Jan. 1993.
[6] A. Myles and D. Skellern, "Comparison of mobile host protocols for IP", *Journal of Internetworking Research and Experience*, vol. 4, no. 4, Dec. 1993.
[7] M. Accetta, R. Baron, W. Bolosky, D. Golub, R. Rashid, A. Tevanian, and M. Young, "Mach: A new kernel foundation for UNIX development", *Proc. of the USENIX 1986 Summer Conference*, July 1986.
[8] B. Shneiderman, *Designing the User Interface*, Addison-Wesley, 1987.
[9] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design", *Proc. of the 2nd International Conference on Distributed Computing Systems*, Apr. 1981.
[10] D. Duchamp and N. F. Reynolds, "Measured performance of a wireless LAN", in *Proc. of the 17th IEEE Conf. on Local Computer Networks*, September 1992.
[11] A. DeSimone, M. C. Chuah, and O. C. Yue, "Throughput performance of transport-layer protocols over wireless LANs", in *Proc. of Globecom '93*, December 1993.

Ramón Cáceres was born in Santo Domingo, Dominican Republic, in 1963. He received a B.Eng. in electrical engineering from McGill University, Montréal, Canada, in 1983, and a M.S. and a Ph.D. in computer science from the University of California at Berkeley in 1984 and 1992, respectively.

Between 1985 and 1987 he designed and developed several data communications products for Pyramid Technology Corporation. Between 1992 and 1994 he investigated networking and operating systems issues in mobile computing at Matsushita Information Technology Laboratory. In 1994 he joined AT&T Bell Laboratories, where he is pursuing research in wireless networking and mobile computing. He is a member of ACM, CPSR, and IEEE.

Liviu Iftode is a Ph.D. student in the Department of Computer Science at Princeton University. His research interests include distributed shared memory, operating systems, and mobile computing.

He received his B.S.E. from the Polytechnic Institute of Bucharest, Romania, in 1984 and a M.A. in computer science from Princeton University in 1994. Between 1984 and 1991 he worked at the Research Institute for Computers in Bucharest, where he led the project to develop System U, a UNIX-like operating system. He has also been on the faculty of the Department of Computer Science at the Polytechnic Institute of Bucharest.