WTCP: An Efficient Mechanism for Improving Wireless Access to TCP Services

KARUNAHARAN RATNAM
Cisco Systems
300 Apollo Drive
Chelmsford, MA 01824, USA
kratnam@cisco.com

IBRAHIM MATTA
Dept. of Computer Science
Boston University
Boston, MA 02215, USA
matta@cs.bu.edu

Abstract— **The Transmission Control Protocol (TCP) has been mainly designed assuming a relatively reliable wireline network. It is known to perform poorly in the presence of wireless links because of its basic assumption that *any* loss of a data segment is due to congestion and consequently it invokes congestion control measures. However, on wireless access links, a large number of segment losses will occur more often because of wireless link errors or host mobility. For this reason, many proposals have recently appeared to improve TCP performance in such environment. They usually rely on the wireless access points (base stations) to locally retransmit the data in order to hide wireless losses from TCP. In this paper, we present WTCP, a new mechanism for improving wireless access to TCP services. We use extensive simulations to evaluate TCP performance in the presence of congestion and wireless losses when the base station employs WTCP, and the well-known Snoop proposal [4]. Our results show that WTCP significantly improves the throughput of TCP connections due to its unique feature of hiding the time spent by the base station to locally recover from wireless link errors so that TCP's round trip time estimation at the source is not affected. This proved to be critical since otherwise the ability of the source to effectively detect congestion in the fixed wireline network is hindered.**

I. INTRODUCTION

Technology advances in communication, hardware and software, and the ever increasing need for un-tethered access to the Internet and intranets have created new opportunities for wireless data communication. However, the transmission quality of wireless channels is characterized by bursty errors with high bit error rate (BER) as opposed to randomly occurring errors with low BER in wired links. The transmission quality of wireless networks is further degraded by the hand-off related intermittent delay or loss of the connection. Hence, the design of a transmission protocol for a network with wireless links must take into account these additional losses on wireless links. The design of such transmission protocol must account for the fact that wireless hosts commonly communicate with hosts in the fixed part of the network, and the transmission procedures used over the wireless link must be compatible with their peer procedures employed in the fixed network. When used in a wireless environment, the Transmission Control Protocol (TCP) [7], developed for fixed networks with relatively reliable links, wrongly assumes segment loss in the wireless portion of the network to be a sign of network congestion and invokes congestion control mechanisms that curb the flow of segments on that connection. However, for better throughput, a wireless loss must be detected and retransmitted as quickly as possible.

We are proposing a new scheme, we call WTCP, to enhance the performance of the transport protocol in networks with wireless links. In our scheme the base station, through which the mobile user is connected to the rest of the network, plays a pivotal role by buffering the TCP segments and locally retransmitting them based on timeouts *as well as* duplicate acknowledgments.[1] WTCP uses efficient flow control for the wireless link, and maintains end-to-end TCP semantics[2]. Furthermore, WTCP uniquely hides the time spent by the base station for local recovery so that TCP's round trip time estimation at the source is not affected. This is critical since otherwise the ability of the source to effectively detect congestion (i.e. losses due to buffer overflow) in the wireline network can be impaired.[3]

The paper is organized as follows. Section II discusses related work. In Section III, we present the proposed WTCP architecture. Section IV describes the details of the simulations we have performed. In Section V, we present the results of the simulation and compare the simulated protocols. Results on handoff and fairness are presented in Sections VI and VII, respectively. Section VIII concludes the paper.

II. RELATED WORK

Many approaches have been proposed to improve TCP performance over networks with wireless links. These approaches can be divided into two major categories, ones that work at the transport level, and others that work at the link level.

Transport level proposals

- **Indirect-TCP**

The Indirect-TCP (I-TCP) [1] proposal splits the transport link at the wireline–wireless border. The base station maintains two TCP connections, one over the fixed network, and another over the wireless link. This way, the poor quality of the wireless link is hidden from the fixed network. By splitting the transport link, I-TCP does not maintain end-to-end TCP semantics, i.e. I-TCP relies on the application layer to ensure reliability. It is argued that many applications provide end-to-end reliability, and one can choose I-TCP if the application provides reliability, and choose TCP otherwise. Thus, the mobile host must be aware of an application's ability to provide reliability when choosing a proper protocol for the transport layer. Provided the mobile host

This work was supported in part by NSF grants CAREER ANI-0096045 and ANI-0095988.

Corresponding author: Ibrahim Matta. Address: Department of Computer Science, Boston University, 111 Cummington Street, Boston, MA 02215. Tel: (617) 358-1062, fax: (617) 353-6457, e-mail: matta@cs.bu.edu.

[1] We point out that congestion due to contention on the wireless link is resolved by a lower layer (MAC) protocol and losses detected at the higher/transport layer of the base station are assumed to be due to wireless errors.

[2] By end-to-end semantics we mean that an acknowledgment to a data segment comes only from the destination host when data is successfully delivered, and not from any other intermediate system.

[3] This paper expands our work in [14], [15].

can choose the appropriate transmission control protocol, the base station should be informed about the mobile host's selection as it is where I-TCP executes. (Relying on the base station to decide on the transport protocol is out of the question as this requires the base station to be aware of the type of application passing through it.)

• **MTCP**

The MTCP proposal [9] is similar to I-TCP, except that the last TCP byte of the data is acknowledged to the source only after it is received by the mobile host. In MTCP, although the source falsely believes that every data byte except the last byte is received by the receiver, it can take remedial action based on whether the last byte is received or not. In particular, if the acknowledgement for the last byte is not received by the source, it has to resend all the data including those that may have already been received by the mobile host.

• **Explicit Bad State Notification (EBSN)**

Like other approaches, EBSN [3] uses local retransmission from the base station to shield wireless link errors and improve throughput. However, if the wireless link is in error state for an extended duration, the source may timeout causing unnecessary source retransmission. The EBSN approach avoids source timeout by using an explicit feedback mechanism. If the wireless link is in bad state, the base station sends an EBSN message to the source for every retransmission of a segment to the mobile host. The EBSN message causes the source to reinitialize the timer. If the base station sends the EBSN messages before the source timer expires, then there will be no timeouts at the source. However, the main disadvantage of this approach is that it requires TCP code modification at the source.

• **Fast Retransmission**

The fast-retransmission approach [10] does not address the issue of wireless link reliability, but reduces the effect of mobile host hand-off. During a mobile host hand-off from one base station to another, TCP segments can be lost or delayed, and the source can timeout. Because of the typical coarse granularity of the TCP clock, the timeout period is much higher than hand-off time, and the mobile host has to unnecessarily wait for a long duration to receive a retransmission of a lost TCP segment from the source. In the fast retransmit approach, immediately after completing the hand-off, the IP in the mobile host triggers TCP to generate a certain number of duplicate acknowledgments. Since most of TCP implementations now have fast-retransmit, these duplicate acknowledgments cause the source to retransmit the lost segment without waiting for the timeout period to expire. This requires modification to the TCP code at the mobile host.

Link level proposals

• **Snoop**

In this scheme [4], the base station sniffs the link interface for any TCP segments destined for the mobile host, and buffers them if buffer space is available. Segments are forwarded to the mobile host only if the base station deems it necessary. Source retransmitted segments that have already been acknowledged by the mobile host are not forwarded by the base station. The base station also sniffs into the acknowledgments from the mobile host. If the base station sees a duplicate acknowledgment, it detects a segment loss and locally retransmits the lost segment if it is buffered and starts a timer. If the retransmitted segment is not acknowledged within twice the round trip time of the wireless link, the segment is again retransmitted. Unlike I-TCP, Snoop does not completely shield the wireless link losses from the fixed network, and source timeout is still possible. In particular, if acknowledgments are lost on the wireless link, a base station retransmission cannot occur as there are no duplicate acknowledgments, and the source can timeout and source retransmission takes place. The transmission over the wireless link resumes only after the arrival of the source retransmitted segment. Hence, in the presence of burst losses the throughput will be poor compared to I-TCP.

• **Channel State Dependent Packet Scheduling (CSDP)**

This is a purely link layer approach, where it has been shown that FIFO scheduling of data over links with bursty losses yields poor utilization of the channel [5]. A set of algorithms are provided to enhance the channel utilization based on the arrival time of the packet, whether it is a new packet or retransmission, and the length of each output queue.

III. WTCP Architecture

We propose a new scheme, we call WTCP, where the base station is involved in the TCP connection. The conceptual view of the transport link is shown in Figure 1. WTCP requires no modification to the TCP code that runs in the mobile host or the fixed host. This is an important advantage at both ends. In the fixed network, one cannot expect the millions of fixed hosts to install a modified TCP implementation to improve throughput for connections involving wireless links. On the mobile host side, by letting the mobile host choose any available TCP implementation, the market share for service providers is increased.

WTCP receives data segment from source

```
wtcp_rcv_data()
   if (pkt_seqno is for next pkt
       expected from src)
       increment seqno of next pkt
       expected from src by pkt_size;
       update receive array;
       buffer pkt;
       store arrival time;

   if (pkt_seqno larger than seqno of
        next pkt expected from src)
       update receive array;
       buffer pkt;
       store arrival time;

   if (pkt_seqno less than seqno of
       next pkt expected from src)
       ignore_packet;
```

The network layer protocol (viz. M-IP [11]) running in the base station detects any TCP segment that arrives for a mobile host and sends it to the WTCP input buffer. If the segment is the next segment expected from the fixed host, it is stored in the WTCP buffer along with its arrival time, and the receive array is updated. The receive array maintains the sequence numbers of the segments received by WTCP. As explained later in Section III-A, the arrival time is stored by the base station in order to

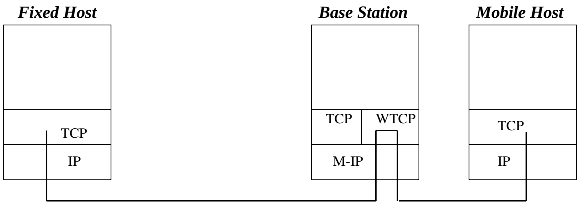

Fig. 1. Conceptual view of the transport connection.

exclude the time spent by the segment in the WTCP buffer and thus hide local recovery time from TCP. The sequence number for the next segment expected from the fixed host is increased by the number of bytes received (`pkt_size`). If the newly arriving segment has a larger sequence number than what is expected, the segment is buffered, the arrival time is recorded and the receive array is updated, but the sequence number of the next packet expected is not changed. If the sequence number of the segment (`pkt_seqno`) is smaller than what is expected, the segment is ignored. Lines 1-11 above show the steps taken when the base station receives a data segment (`pkt`) from a source (`src`).

WTCP sends data segments to mobile host

On the wireless link, WTCP tries to send the segments that are stored in its buffer. WTCP independently performs flow control for the wireless connection. It also maintains state information for the wireless connection, such as transmission window, sequence number of last acknowledgment received from the mobile host, and the sequence number of the last segment that was sent to the mobile host. WTCP transmits from its buffer all the segments that fall within the wireless link transmission window (`t_window`). Each time a segment is sent to the mobile host (including a retransmission), the timestamp of the segment is incremented by the amount of time that segment spent in the WTCP buffer (`residence_time`). The reason for doing this is discussed in detail in Section III-A. When a segment is sent to the mobile host, the base station schedules a new timeout if there is no other timeout pending. Lines 12-18 shows the steps taken to send data segments to a mobile host (`MH`).

```
wtcp_send_data()
12 while (seqno of next pkt to
     send to MH is within t_window)
13   retrieve pkt from buffer;
14   set timestamp to original
     timestamp plus residence_time;
15   if (no timeout pending)
        schedule new timeout;
16   if (no rtt estimation pending)
        // rtt: round trip time
        mark pkt for new rtt
        estimate;
17   send(pkt);
18   increment seqno of next pkt
     to send to MH by pkt_size;
```

WTCP receives acknowledgement from mobile host

The base station acknowledges a segment to the fixed host *only after* the mobile host actually receives and acknowledges that segment. Hence, TCP end-to-end semantics is maintained throughout the lifetime of the connection. Also, the round trip time seen by the source is the actual round trip time taken by a segment to reach and return from the mobile host, *i.e.* it does not include residence time at the base station needed for local recovery. The corresponding pseudo code is given below in lines 19-30.

```
wtcp_rcv_ack(ack)
19 if (seqno of ack is larger
      than last_ack)
20    set t_window to receiver
      advertised window;
       // fully open window
21    set last_ack to seqno of ack;
22    set dupack counter to zero;
      // initialize dup ack counter
23    if (seqno of ack covers a
         segment marked for rtt
         estimate)
24       compute new rtt;
25    forward ack to src;
26    delete acked segments from
      buffer;
27 if (seqno of ack same as
       last_ack)
      // duplicate ack
28    increment dupack counter;
29    if (first dupack)
30       send lost pkt to MH;
```

Based on duplicate acknowledgment or timeout, the base station locally retransmits lost segments. In case of timeout, the transmission window for the wireless connection is reduced to just *one* segment assuming a typical burst loss on the wireless link is going to follow. If only one segment is lost and the following segment(s) pass(es) through, the loss would likely be indicated by a duplicate acknowledgment (see below). In case of timeout, by quickly reducing the transmission window, potentially wasteful wireless transmission is avoided and the interference with other channels is reduced.

The opening of the transmission window in regular TCP is based on whether the connection is in the slow start or congestion avoidance phase. In the slow start phase, each time an acknowledgment is received, the congestion window (cwnd) is incremented by one segment size. In the congestion avoidance phase, each time an acknowledgment is received, the congestion window is incremented by a factor of 1/cwnd. WTCP however opens the wireless transmission window completely assuming that an acknowledgement indicates that the wireless link is in good state. That is, when an acknowledgment is received, the transmission window size is set to the window size advertised by the receiver.

TCP-reno assumes a duplicate acknowledgment is caused by a TCP segment being lost due to congestion and the connection is put in the congestion avoidance phase. In other words, after a duplicate acknowledgment, the transmission window is halved and then incremented only by one segment per round trip time. However, for duplicate acknowledgment, WTCP does *not* alter the transmission window assuming that the reception of the duplicate acknowledgement is an indication that the wireless link is in good state, and immediately retransmits the lost segment. The base station continues to transmit the remaining segments that are within the transmission window if they have not already been transmitted. However, until the mobile host receives the lost segment, the reception of each out-of-order segment will generate a duplicate acknowledgment. The number of these additional duplicate acknowledgments can be determined by the base station, which ceases to retransmit during their reception. By avoiding more than one duplicate acknowledgment based retransmission for a segment, WTCP improves the utilization of the wireless channel.

Clock granularity

In most multi-hop networks, the round trip time is typically in the order of few hundred milliseconds. Because of this reason, many TCP implementations use a coarse clock granularity. For example, 4.4BSD-Lite uses a 500ms clock [19]. On the other hand, the base station–mobile host link is single-hop, and the round trip time is typically in the order of tens of milliseconds. Hence, in order to accurately estimate round trip time and to trigger timeouts in the millisecond range, WTCP uses a 10 ms clock granularity.

A. *Effect of Local Retransmission on RTT Estimation*

In the older version of TCP specified in RFC 1185 [8], the round trip time (RTT) is computed once per transmission window, and none when there is a timeout. Since RTT is not frequently updated, the RTT estimate may not be accurate. Inaccurate RTT estimation can cause unnecessary retransmissions by triggering timeouts too soon or can degrade the throughput by triggering timeouts too late. The newer implementation of TCP specified in RFC 1323 [7] uses the timestamp option in the TCP header to better estimate the round trip time. The sender writes the current timestamp in the header, and the receiver echoes back this timestamp without modifying it. Since each acknowledgment contains the timestamp of the segment that generated it, the round trip time can be computed for every acknowledgment, even in the presence of timeout in a transmission window. However, when an intermediate host (base station) buffers the segments and locally tries to retransmit them, the RTT computation can be affected. For example, if a segment is successfully retransmitted after one or more timeouts at the base station, the RTT value computed at the source for this segment will overshoot by the residence time of this segment at the base station buffer. If the burst error duration of the wireless link is significantly long, the residence time of the segment in the WTCP buffer may span a few source clock ticks and affect the RTT computation, especially its variance. Even after successful transmission resumes on the wireless link and acknowledgments are sent back to the source, the RTT computation at the source may take a long time to settle down to the actual value. This problem exists in all the proposals that use local retransmission by the base station except I-TCP [1]. To hide the wireless link errors from the source, ideally the residence time of the segment at the WTCP buffer could be added to the timestamp value in the TCP header before the acknowledgment is forwarded back to the source. This has the effect of subtracting the residence time at the base station from the RTT value computed by the source.[4]

This approach requires the base station to know the clock granularity of the source since the timestamp field in the TCP header contains a clock tick value, not the real-time clock value. In particular, when a system is booted, the clock tick is set to 0 and is incremented by one every 500 ms in 4.4BSD-Lite. To get around the problem of not knowing the clock granularity, an *approximation* to subtracting the exact residence time at the base station is as follows: when a segment is sent to the mobile host the base station can replace the timestamp of that segment with the timestamp of the *most recent segment* received from the source.[5] When the acknowledgment comes back from the mobile host, it is forwarded to the source without any modification. When there are no source timeouts, the base station receives TCP segments from the source at least once every RTT interval. Hence the maximum error in the RTT computation using the most recent timestamp is bounded by one RTT (see Figure 2) when there are no source timeouts.

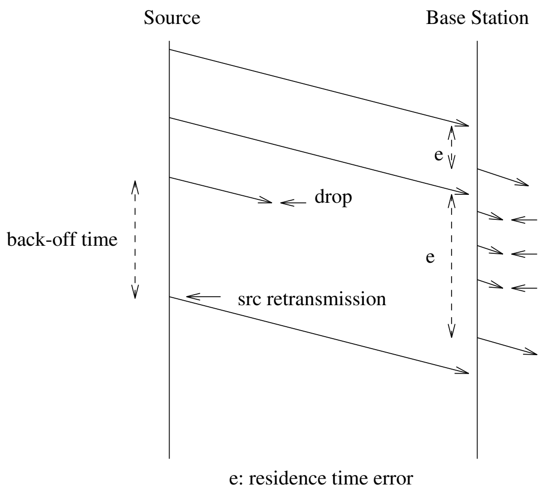

Fig. 2. Diagram showing the error in RTT using the most recent timestamp.

The above approach works fine as long as the base station frequently receives TCP segments from the source. However, if the source timeouts due to a prolong burst error state of the wireless link or due to congestion in the wired portion of the network, the inter-arrival time between source TCP segments at the base station can be large. During this phase, the error in excluding residence time using the timestamp of the most recent segment received by the base station can be as large as the source back-off time (see Figure 2). Any acknowledgment sent to the source using this timestamp will obviously affect the RTT estimate maintained at the source by a large factor. To avoid this problem, before sending a segment over the wireless link,

[4] Let t_s be the original timestamp value in the packet header, i.e. the time the packet was sent by the source. Let t_d be the residence time (spent at the base station), and t_r be the time the acknowledgment is received by the source. Then by modifying the timestamp in the header to $t_s + t_d$, the RTT computed by the source for this packet is $t_r - (t_s + t_d) = (t_r - t_s) - t_d$. Thus the RTT computation excludes t_d.

[5] A trivial case is when the WTCP buffer is empty. When a TCP segment arrives, it is obviously the most recent segment received at the base station. Thus, in this case, the new timestamp will be the same as the original timestamp in this segment's header yielding an RTT value that indeed does not include any residence time.

the base station can remove its timestamp option field if there has been any retransmission from the base station *after* receiving the most recent segment from the source. Timestamps are again used only after receiving a TCP segment from the source.

IV. Simulation Model and Performance Measures

In our evaluations, we used the network simulator **ns** v.1.2a3 [13], a public domain software from Lawrence Berkeley National Lab. We extended ns to incorporate wireless link characteristics and to simulate WTCP, I-TCP, and Snoop. For all three protocols, the clock resolution of the base station is changed to 10 ms, and the resolution of the fixed host is maintained at 100 ms. No changes are made to the fixed network hosts or the mobile hosts. However, the base station node is modified to receive, buffer and transmit packets to the mobile host.

In the case of I-TCP, the incoming segment from the fixed host is accepted and buffered, and an acknowledgment is *immediately* sent back to the source. A separate TCP connection is created for the mobile host, and all segments in the base station buffer are transmitted over this wireless connection.

In order to avoid buffer overflow at the base station, the maximum window size of the TCP connection on both sides of the base station is chosen to be smaller than its maximum buffer size. However, in the case of I-TCP, buffer overflow can easily happen due to the split nature of the connection. TCP handles the receiver buffer overflow situation by using receiver window size advertisements. Thus for the I-TCP simulation, we modified the ns code at the base station to advertise available buffer size to the source as the window size in every acknowledgement. When there is no buffer space available, the base station advertises a window size of zero, and the source refrains from transmitting any more data until the window size is increased. The base station will increase the window size only if it can accept at least one full TCP segment [16]. For WTCP and Snoop, the base station buffer cannot become full as the transport connection is *end-to-end*. For an end-to-end connection, the number of unacknowledged segments sent out by the source is bounded by the transmission window. Since the maximum window size is chosen to be smaller than the buffer size at the base station, the base station buffer can never overflow.

In WTCP, when the base station receives a segment from the fixed host, the segment is saved in the buffer. Whenever the base station is ready to transmit that segment, a new header is created, and every header field from the buffered segment is copied to the new header except the timestamp value and the checksum. As discussed earlier in Section III-A, we modify the timestamp in two different ways to hide the effect of wireless link errors on the RTT estimation at the source. The *first method* assumes an ideal scenario where the clock granularity of the source is known to the base station, and the base station increments the timestamp by the actual (exact) residence time of the segment at the base station. The *second method* makes no assumption about the source clock granularity, and uses the timestamp of the most recent segment received from the source as an approximation to excluding residence time at the base station from the RTT estimate. These two methods are simulated separately and the results are compared in Section V.

WTCP also measures the base station–mobile host RTT. If there is no RTT estimate pending, WTCP marks the next segment sent to the mobile host for RTT estimate and stores the sending time of that segment. When the first acknowledgement that covers the sequence number of this marked segment arrives at the base station, the base station computes the RTT by subtracting its sending time from its arrival time. Since only one hop is involved in the base station–mobile host link, the round trip time variation is small. Hence, the smooth round trip time estimation used in regular TCP is not employed in WTCP. When there is no base station timeout, the timeout value is set to 1.5 times the wireless link RTT. WTCP uses a linear back-off policy, that is, the timeout value is always $1.5 \times$ RTT. The value of 1.5 is chosen so that any minor variation in the RTT does not trigger unnecessary timeouts. The selection of the back-off value depends on the nature of the wireless link. The first retransmission occurs at $1.5 \times$ RTT. If the exact duration of the burst error (bad) state of the wireless link is known, the second retransmission can be made to occur immediately after the link returns to a good state. However, since the exact burst error duration is not known, the second retransmission can be made to occur after the mean burst duration by selecting a suitable back-off period. For example, if the mean burst duration is around $4 \times$ RTT, then the back-off duration for the second retransmission can be set to $3 \times$ RTT so that the total time for the second retransmission to take place is ($1.5 \times$ RTT + $3 \times$ RTT =) $4.5 \times$ RTT, which covers the mean bad state duration.

The round trip delay in the wireless portion of the network is usually small (except in the case of satellite links). Hence, for better utilization of the wireless link, the window size of WTCP for the wireless portion of the connection should be neither very large nor very small. The optimal window size is given by the delay-bandwidth product of the wireless link. Hence we chose a WTCP window size of 3 in the simulation. Note here that, the maximum window size of the end-to-end TCP connection is not changed (it is given in Table I).

We also simulated the Snoop protocol based on [4] for comparison purposes. Snoop uses negative acknowledgment at the mobile host to improve performance. We did not implement negative acknowledgments because our intention is not to modify TCP at the mobile host.

Wireless channel error model

Unlike the wired link, wireless links are characterized by high bit error rates. The errors are generally bursty in nature with varying burst duration. We model the wireless link using a two-state Markov channel [5], where the channel is in either good state or bad state. In the good state the bit error rate (BER) of the channel is low, and in the bad state BER is high. We simulated the wireless link with bit error rates of 10^{-6} and 10^{-2} in good and bad states, respectively. The time spent in each state is modeled by an exponential distribution with mean good state duration fixed at 1 second. The mean bad state duration is varied from 10 ms to 100 ms.

A. Performance Measures

Our intention is to improve the throughput performance of a transport connection. The *throughput* is measured as the average number of bits successfully transmitted to the mobile host in a second.

Improving throughput may result in poor utilization of the wireless channel. For example, if the wireless channel is in bad state, aggressive retransmission by the base station will likely result in loss of segments. Though aggressive retransmission improves throughput, it also increases the interference with other wireless links. To capture this tradeoff, we measure *goodput* [3] defined as the ratio of the number of new (header + data) bytes transmitted over the wireless link and the total number of bytes transmitted (new and retransmissions).

V. RESULTS AND DISCUSSION

We have simulated the network shown in Figure 3 with error free wired link and erroneous wireless link. The wired link bandwidth is 10 Mbps with 100 ms latency. The wireless side has an 800 Kbps link with 10 ms latency. Table I shows the parameters used in the simulation. The simulation is run for 200 seconds and 95% confidence intervals were computed for the performance measures.

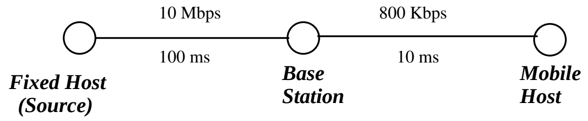

Fig. 3. Simulated network.

Parameter	Value
Maximum window size	10 Kb
Buffer size at base station	20 Kb
Segment size (including header)	1 Kb
BER in good state	10^{-6}
BER in bad state	10^{-2}
Mean good period	1 sec

TABLE I
SIMULATION PARAMETERS.

Figure 4 shows the trace of data transfer on the fixed link (top graph) and wireless link (bottom graph) using our WTCP protocol. The x-axis shows the time when the segment is put on the output queue of the fixed or wireless link, and the y-axis shows the segment's sequence number modulo 100. The plus sign indicates the dropped segments. It can be seen from the fixed link trace that source retransmissions occur only at around 7 seconds. However, from the wireless link trace, it can be seen that the corresponding wireless burst error (bad) duration is around 1 second, which is around four times the RTT seen by the source.[6] If the burst duration is in the range of few hundred milliseconds as in typical RF wireless environments [5] there are no source timeouts.

The throughput results of I-TCP, Snoop, WTCP and TCP-tahoe[7] are shown in Figure 5. For a good quality wireless link, the throughput of I-TCP is better than other protocols. However, when the wireless link quality degrades WTCP yields better throughput mainly because of its aggressive retransmission policy over the wireless link. WTCP achieves throughput values 4-8 times higher than TCP-tahoe.

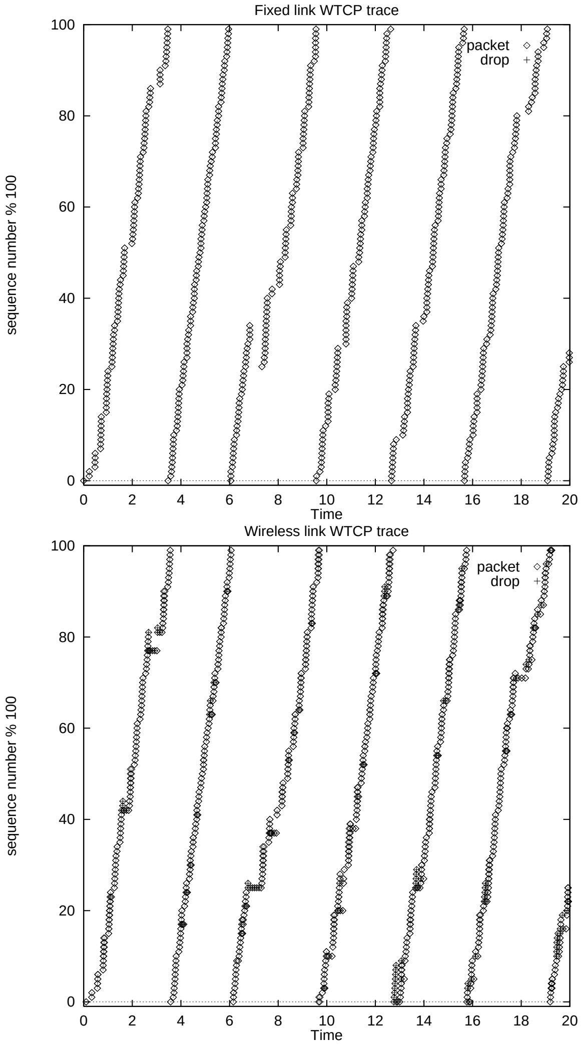

Fig. 4. WTCP trace.

Though Snoop achieves throughput values comparable to I-TCP and WTCP at low loss situations, the throughput of Snoop is poor when the wireless link is very bursty (in bad error state) for long durations. The reason is that Snoop triggers retransmission only after the base station receives a duplicate acknowledgment. However, when the wireless link is in bad error state, it is likely that acknowledgments be lost. In this situation, the base station fails to trigger duplicate acknowledgment based retransmission, and the throughput degrades.

WTCP's aggressive retransmission policy slightly reduces the goodput of the wireless link as shown in Figure 6. However, the throughput gain (4-8 times higher than TCP-tahoe) outweighs the small loss in goodput (around 2%). There is a clear tradeoff between throughput and goodput. One can improve goodput by

[6] The round trip time from the source to mobile host with 1 Kbyte data segment and 40-byte acknowledgment for the network topology shown in Figure 3 is approximately 230 ms.

[7] We do not show results for TCP-reno as it was found to be less robust than TCP-tahoe in wireless applications [12].

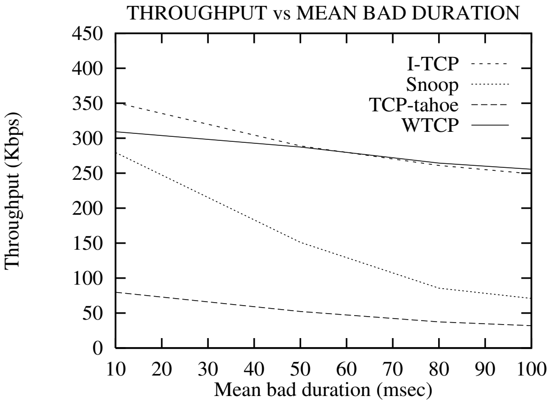

Fig. 5. Throughput of different protocols.

using a less aggressive retransmission policy over the wireless link at the expense of decreased throughput. However, the gain in goodput will be minimal compared to the loss in throughput.

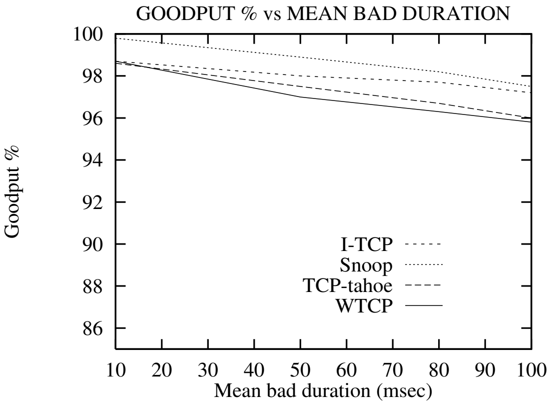

Fig. 6. Goodput of different protocols.

WTCP: On Hiding the Effect of Local Retransmission on RTT Estimation of TCP

By storing TCP segments at the base station and locally retransmitting them, WTCP and other proposals shield most of the wireless link errors from the source and improve throughput for reliable bulk data transfer. But, the amount of time a segment spends in the base station buffer *is* an indication of the wireless link state, and should be hidden from the source as much as possible. Otherwise, the ability of the source to effectively detect congestion in the fixed wireline network will be affected. Depending on the nature of the wireless link, the amount of time a segment spends in the base station buffer (residence time) can span a few source clock ticks and offset the RTT estimate maintained at the source by a large value.

In Figure 7 we plot the source timeout values, sampled every second, as a function of time, when WTCP is employed at the base station. The plotted timeout value is the initial timeout computed by the TCP source, which is close to the smooth (average) RTT between the fixed host and mobile host when the variance is small. The granularity of the source clock is 100 ms, the mean bad period of the wireless link is 100 ms, and there are no congestion losses in the wired network. Figure 7 shows the effect of (a) not hiding the residence time; (b) excluding actual residence time assuming the granularity of the source clock is known to the base station; and (c) using the timestamp of the most recent segment received from the source with timestamp removal option (discussed earlier in Section III-A). When the residence time is not excluded, the timeout value reaches as high as 2.4 seconds, a five fold jump from the timeout value when the residence time is excluded.

In the latter case, the timeout value varies between 500–700 ms. However, when the exact value of the residence time is excluded, the timeout value varies more slowly compared to when most recent timestamp is used. When the exact value of residence time is used, the small variations in the timeout value are due to minimal queuing delays. In particular, if there is a retransmission at the base station, WTCP closes the wireless transmission window to just one segment. However, when an acknowledgment is received, the transmission window is opened fully, and the packets are queued on the wireless link for transmission. For an 800 Kbps link, the maximum queuing delay is 90 ms if ten 1 Kbyte packets are queued.

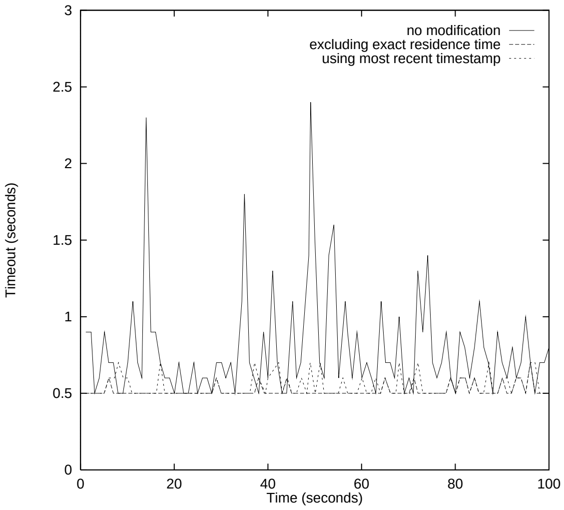

Fig. 7. Plot of sample timeout values with 100 ms clock.

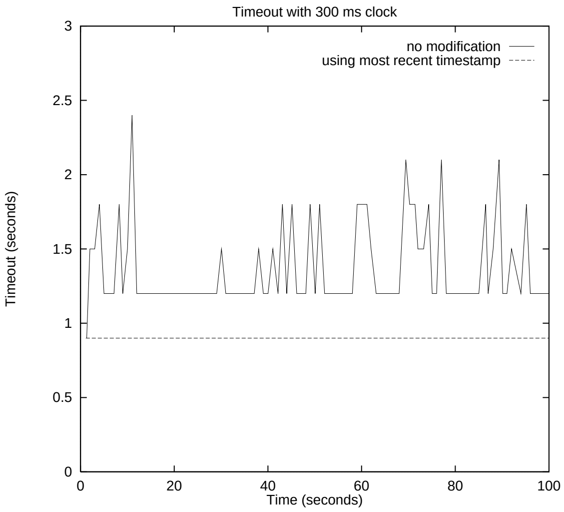

Fig. 8. Plot of sample timeout values with 300 ms clock.

If, as in most TCP implementations, a coarser clock granularity is used at the source, the queuing effect is indiscernible. In Figure 8 we show the plot of timeout values when the source clock granularity is 300 ms. When the timestamp of the most recent segment is used or the exact residence time is excluded, the timeout value is always 900 ms (i.e. 3 clock ticks). However, if the timestamp is not modified (i.e. residence time is not excluded) the timeout value reaches as high as 2.4 seconds.

In summary, if the timestamp is not modified to exclude the residence time in the base station buffer, the timeout value maintained at the source will be high. Figures 7 and 8 shows source timeout values when the wired network causes no congestion loss. However, in the presence of congestion losses, the difference in source timeout values when including and excluding residence time is even more pronounced than in the no congestion case as shown in Figure 9.

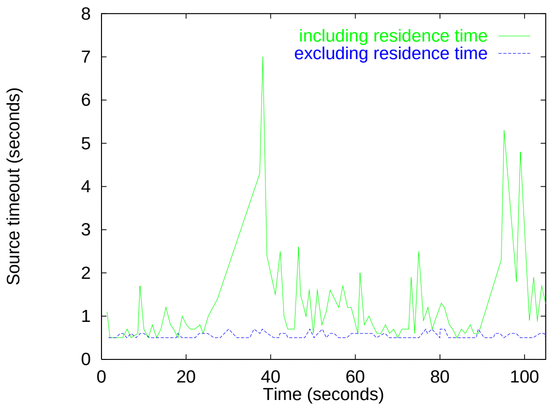

Fig. 9. Plot of sample timeout values with 100 ms clock and congestion loss probability 0.01.

As pointed out earlier, this affects the TCP source's ability to effectively detect any congestion losses in the wired network portion of the connection. To see this effect we modeled a lossy wired link which randomly looses data packets with probability *P*. Figure 10 shows the throughput for WTCP without modifying the timestamp and with modifying the timestamp to exclude the residence time (using the timestamp of the most recent segment), as well as the throughput for Snoop and TCP-tahoe. The wireless link is modeled by a two-state error channel as before, and the mean good period is 1 second. The effect of residence time on the TCP source's ability to detect congestion losses in the wired link can be clearly seen in Figure 10(a) where the wired loss probability is 0.01. The advantage of hiding the residence time at the base station becomes even more pronounced for higher wired (congestion) loss probabilities. However, when there are only few wired link losses (Figure 10(b)), as expected, the number of data segments lost is low, and hence number of retransmissions is low and excluding the residence time does not provide significant improvement in throughput.

Effect of background load

We have looked at the performance of several wireless TCP approaches by artificially introducing congestion losses in the wireline network. Here we add background traffic to the wireline network to cause congestion losses. We add 20 TCP-tahoe

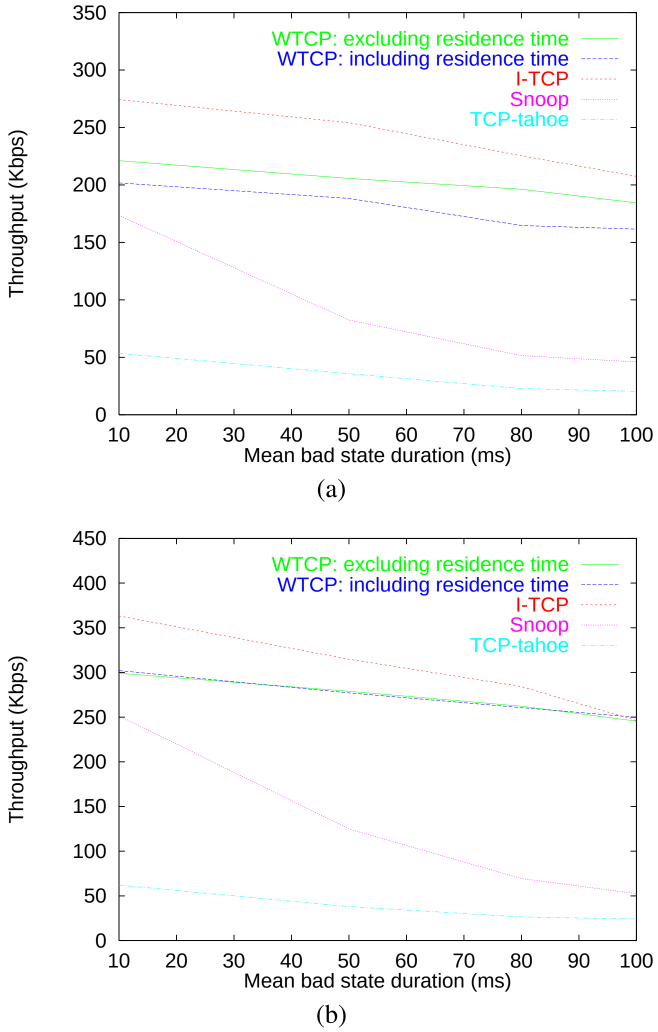

Fig. 10. WTCP throughput with wired (congestion) link loss probability (a) $P = 0.01$; (b) $P = 0.001$.

connections from the fixed host to the base station.

Figure 11 shows the throughput performance in the presence of background traffic. The average packet dropping rate over the wired link is observed to be 1%, similar to Figure 10(a) of no background traffic but artificial (wired) drops. I-TCP yields better throughput in the presence of background traffic than in the absence of it. The reason being, the I-TCP traffic itself now experiences a loss rate smaller than 1%. The throughput of Snoop and WTCP suffers when background traffic is added. Both of these are end-to-end transport connections unlike I-TCP which splits the connection at the base station. TCP is known to be unfair to connections with large delay-bandwidth product [12]. Thus the throughput of WTCP and Snoop suffers due to the longer paths and additional delays introduced by the background load. This is also why the difference in WTCP throughput between including and excluding residence time is more pronounced in the presence of background load than in the absence of it.

VI. Handoff

Two main causes of data errors in both fixed wireless environments and mobile cellular environments are multi-path fading

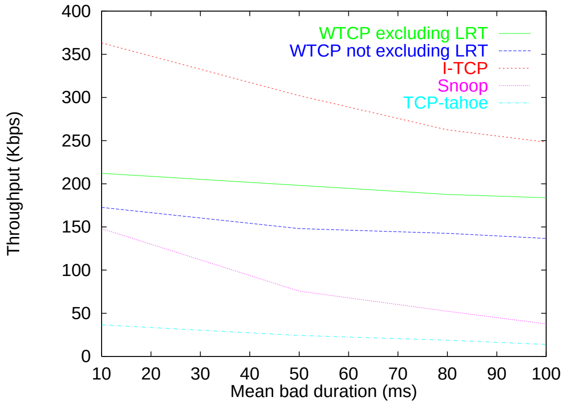

Fig. 11. Throughput performance with background traffic.

and rain fading. In fixed wireless networks, where the wireless terminals are stationary, man-made structures and terrain contribute to fading. The user mobility also causes fading in networks such as cellular, Mobile-IP, and wireless local area networks. Another cause of data error in such networks is handoff of the data connection from the current base station (or access point) to another triggered by the mobile user moving away from the current cell into the other cell. In the previous section we have compared the performance of transport layer protocols in the presence of wireless channel errors that arise due to fading in fixed wireless and mobile cellular environments. In this section, we compare the transport layer protocol performance in the presence of both handoff as well as fading related errors.

The cellular network provides circuit switched[8] data connections, and delivers error free data service to higher layers using radio link protocols. In such environment, handoff does not cause packet drops. But, in packet switched networks, such as Mobile-IP and wireless LAN, handoff may cause consecutive packet drops. These packet drops and handoff delay may cause the TCP sender to timeout, where the sender shrinks the congestion window to just one segment, and the transmission is put into slow start phase. Even if the delay is short enough to not trigger a timeout, these consecutive packet drops cause the receiver to generate duplicate acknowledgements, which may cause the sender to go into the congestion avoidance phase (TCP-reno) or slow start phase (TCP-tahoe). In effect, these packet drops and delay diminish TCP's throughput performance.

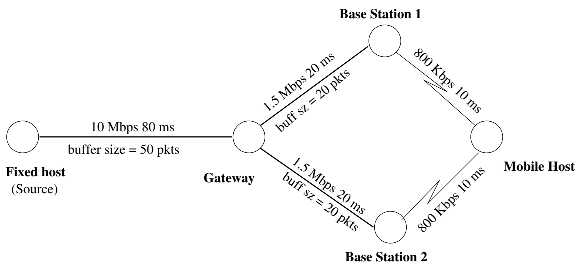

Fig. 12. Conceptual view of the transport connection.

Let's look at the throughput performance of several wireless TCP protocols in the presence of handoff and wireless channel errors. We simulated a transport connection using different protocols on the network shown in Figure 12. The amount of time the mobile host spends in a base station cell is assumed to be exponentially distributed with mean 10 seconds. The background traffic in the network is modeled by 20 TCP-tahoe connections from the source node to the gateway node. The wireline network uses drop-tail links with buffer sizes as shown in the figure. We observed that, this network setup causes an average packet drop rate of about 1% on the source to gateway link.

We simulated a WTCP, I-TCP, Snoop, and TCP-tahoe connection from the fixed host to the mobile host. The local retransmission algorithms are run on the gateway node for WTCP, I-TCP, and Snoop. The handoff delay (i.e. time between breaking connection with the old base station and making connection to the new base station) is 50 msec. When the mobile host moves from the old base station and registers with a new base station, the latter sends a routing update message to the gateway. Until the new location is updated in the gateway node's routing table, it sends data packets destined to the mobile to the old base station, where the packets will be discarded.

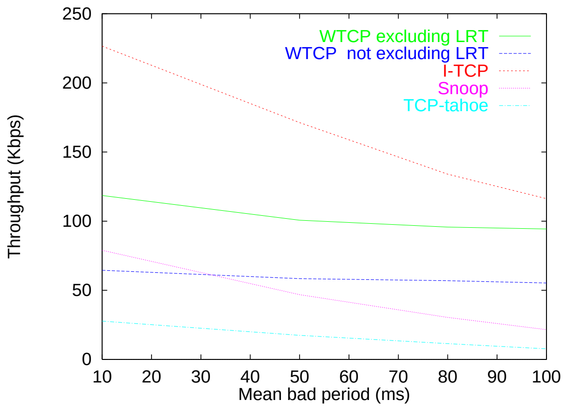

Fig. 13. Throughput performance with mobility and background traffic.

Figure 13 shows the corresponding results for different wireless TCP schemes. As expected, I-TCP detects and mitigates wireless losses more efficiently than other schemes. But, among those protocols that do not violate end-to-end TCP semantics, WTCP is more throughput efficient. Excluding the local recovery time almost doubles WTCP throughput. The reason is, since the WTCP algorithm is run on the gateway node, excluding the local recovery time excludes the handoff delay as well.

There are many proposals for reducing the effect of handoff on the data throughput. One of the first such proposals for TCP was *fast-retransmit* [10], where, after the handoff is completed, the mobile host (i.e. the receiver) sends a number of duplicate acknowledgements to the sender irrespective of receiving any new TCP packets. This enables the sender to detect packet losses immediately after the completion of handoff, and triggers retransmission of lost packets.

Another approach is to identify the base station(s) to which the mobile host is potentially going to move, and forward IP packets (i.e. TCP and UDP) in advance to those base station(s) so that when the mobile host changes its base station, it continues to receive IP packets from the new base station without large delay. This multicasting approach needs to address many issues;

[8]Pure packet switched data service is provided in the second and third generation cellular data standards [18][17].

how to predict the potential base station to which the mobile host may move, when to start advance forwarding, how many advance forwarded packets a base station buffers, how long they need to be buffered, what types of packets to advance forward, and more importantly how to prevent duplicate packets. A next cell prediction scheme for wireless LAN networks is presented in [6]. It maintains the profile of the cells and mobile users, and uses it to predict when and to which base station cell the mobile host is going to move, and reserves resources for the mobile host in advance. To explain it using a simple example, the profile of the base station cell could be what type of area it covers, such as office room, conference room, hallway, etc. If the base station covers a conference room, for instance, then the algorithm could use the conference room reservation schedule in predicting when the mobile hosts are going to move. One can imagine many different profiles to base the prediction depending on the deployment scenario, and the solution needs to be tailor made for a particular deployment for better results. A simpler and more generic approach is presented in [2]. It provides a probabilistic guarantee of loss free handoff by initiating the advance forwarding at a fraction (stagger) of the average residency time, which is computed based on the velocity of the mobile and the size of the cell.

Which node is the best node to be the root for multicasting the IP packets? A natural choice would be the gateway router, through which the traffic from and to the wireless network flows. Thus we let the gateway node (cf. Figure 12) perform both multicasting/advance forwarding to deal with handoffs and retransmission to deal with locally detecting and mitigating wireless impairments.

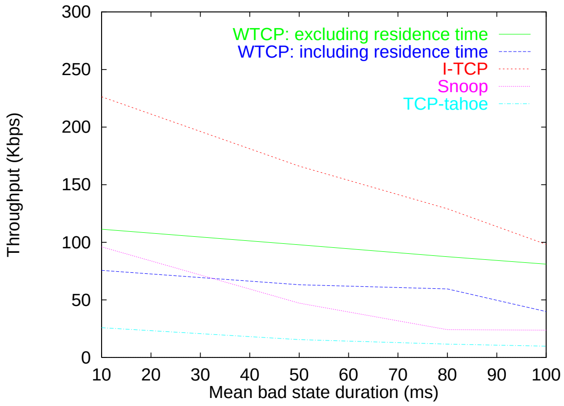

Fig. 14. Throughput performance using advance forwarding in the presence of mobility and background traffic.

We expect that TCP throughput cannot be significantly improved by advance forwarding because of TCP's flow control which limits the potential number of advance forwarded packets. Advance forwarding is initiated at 70% of the residence time. The results are shown in Figure 14. Comparing the result with that of no advance forwarding (Figure 13), it is clear that, as expected, advance forwarding does not significantly improve the throughput of any of the wireless TCP protocols. However, advance forwarding may improve data throughput of other types of transport services that do not employ flow control, such as UDP.

VII. FAIRNESS OF WTCP AND OTHER TCP CROSS TRAFFIC

In this section we investigate the effect of including/excluding the local recovery time (LRT) in WTCP on other TCP cross traffic in the network. In the absence of packet loss in the network, including or excluding LRT does not significantly impact WTCP throughput (cf. Figure 10(b)). Hence, to investigate the fairness of WTCP, we vary the bandwidth and buffer size of the wired network link to vary the (congestion) loss rate. Figure 15 shows the network configuration used in the analysis. We refer to the TCP connection from the fixed host to the gateway as 'tcp' connection, and we refer to the WTCP-supported connection from the fixed host to the mobile host as 'wtcp' connection.

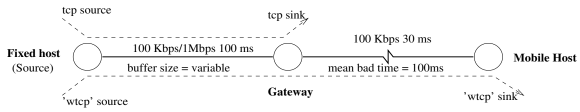

Fig. 15. Network topology used in analyzing fairness.

We first present the throughput of each connection in the absence of congestion losses, which is achieved by having sufficient buffers in the links. Here we consider the effect of tcp window size on wtcp throughput. The wired link (fixed host to gateway) bandwidth is set to 100 Kbps, and buffer size is 50 packets, which ensures that there is no congestion related losses. Table II presents three sets of results. In the first set, the maximum window size of the tcp connection is set to 10 packets. In the second and third cases, the maximum window size of the tcp connection is increased to 20 and 40 packets, respectively, while keeping the rest of the parameters the same as in the first set. For all three sets, the (end-to-end) maximum window size of the wtcp connection is 10 packets. The results clearly show that, a tcp connection with a larger window size grabs a larger share of the bandwidth. The last set of results in Table II shows the throughput of two tcp connections, one from the fixed host to the gateway, and the other from the fixed host to the mobile host. The window sizes of both tcp connections are set to 10 packets. In this case, the link between the gateway and mobile host is replaced by a wired link, i.e. no wireless losses. As expected, the throughput of the short tcp connection is higher than that of the long tcp connection. Observe that the throughput of the wtcp connection in setup 1 is very close to that of the long tcp connection. This confirms the effectiveness of WTCP in hiding wireless losses *without* unfairly grabing bandwidth from other regular TCP connections.

Next we introduced congestion in the network by limiting the wired link buffer size. There is no buffer overflow at the base station. The (end-to-end) window sizes of connections are fixed at 10 packets. The wired link bandwidth is set to 1 Mbps, and we vary its buffer size to vary the congestion loss rate. In Table III, we compare the results of wtcp throughput and the average throughput of 20 tcp connections which are all from the fixed host to the gateway. The results show that, by excluding LRT, wtcp achieves throughput gain of up to 265%, with insignificant loss of tcp throughput. That is, excluding LRT in WTCP connections does not unfairly reduce the throughput of other TCP connections in the network.

Experiment	Connection	Throughput (Kbps)	
		Excluding LRT	including LRT
1	tcp (win = 10)	57.08	56.68
	wtcp	42.24	42.68
2	tcp (win = 20)	69.32	69.48
	wtcp	30.28	30.12
3	tcp (win = 40)	80.76	80.64
	wtcp	18.80	19.40
4[a]	tcp (Src-GW)	52.36	
	tcp (Src-MH)	47.36	

[a]The wireless link GW-MH is replaced by a drop-tail link. i.e. no wireless losses

TABLE II
THROUGHPUT COMPARISON IN CONGESTION FREE NETWORK.

Experiment	Connection	Throughput (Kbps)	
		Excluding LRT	including LRT
1	tcp	48.68	49.36
buffer = 50 pkts	wtcp	16.44	6.12
		Excluding LRT	including LRT
2	tcp	48.48	49.20
buffer = 100 pkts	wtcp	21.20	9.28
		Excluding LRT	including LRT
3	tcp	47.96	47.92
buffer = 800	wtcp	36.96	37.36

TABLE III
THROUGHPUT COMPARISON IN CONGESTED NETWORK - 20 TCP CONNECTIONS.

VIII. CONCLUSION

We presented a new reliable transport layer mechanism, called WTCP, for a network with wireless links. While maintaining end-to-end TCP semantics and requiring no modification to TCP running on the fixed or mobile host, WTCP runs on the base station (or some intermediate gateway) and achieves very high throughput compared to TCP-tahoe. We have simulated other schemes (I-TCP and Snoop) for comparison purposes. WTCP performs better or as well as other schemes in terms of throughput with only a slight loss in goodput. WTCP effectively hides the time spent by the base station to locally recover so as not to hinder the source's ability to effectively detect congestion in the fixed wireline network.

The effectiveness and superiority of WTCP is confirmed in the presence of background load and mobility related handoff of the mobile host. We also show that the technique of hiding the local recovery time significantly improves the performance of WTCP-enabled connections without exceeding their fair share of the bottleneck bandwidth.

The goal of WTCP is to hide wireless losses from TCP sources. However, some wireless networks employ radio link protocols (RLP) and forward error correction to improve the reliability of the wireless links. RLP ensures reliable transmission at the link layer by detecting lost data and retransmitting them. If the wireless link layer is completely reliable, then WTCP and other similar schemes offer no improvement. However, many wireless networks that carry multi-media traffic do not employ an RLP, or if they use it, limit the number of retransmissions to reduce latency. For example, GSM and IS95 (North American CDMA) limit the latency by allowing only a few retransmissions. GSM even has the option of turning off the RLP. In those networks, WTCP can be used to significantly improve the performance of TCP connections.

REFERENCES

[1] Ajay Bakre and B. R. Badrinath. Handoff and systems support for indirect TCP/IP. In *Proceedings of the second USENIX Symposium on Mobile and Location-Independent Computing, pages 11–24, April 10–11, 1995, Ann Arbor, Michigan, USA*.

[2] Bikram Bakshi, P. Krishna, D. K. Pradhan, and N. H. Vaidya. Providing seamless communication in mobile wireless networks. Technical Report 96-015, Texas A & M University, 1996.

[3] Bikram S. Bakshi, P. Krishna, N. H. Vaidya, and D. K. Pradhan. Improving performance of TCP over wireless networks. Technical Report 96-014, Texas A & M University, 1996.

[4] Hari Balakrishnan, Venkata N. Padmanabhan Srinivasan Seshan, and Randy H. Katz. A comparison of mechanisms for improving TCP performance in wireless networks. In *ACM SIGCOMM Symposium on Communication, Architectures and Protocols*, August 1996.

[5] Pravin Bhagwat, Partha Bhattacharya, Arvind Krishna, and Satish K. Tripathi. Enchancing throughput over wireless LANs using channel state dependent packet scheduling. In *Proc. IEEE INFOCOM'96*, pages 1133–40, March 1996.

[6] Vaduvur Bharghavan and Jayanth Mysore. Profile-based next-cell prediction in indoor wireless LANs. IEEE Singapore International Conference on Networking, 1997.

[7] D. Borman, R. Braden, and V. Jacobson. RFC 1323: TCP extensions for high performance, May 1992. Obsoletes RFC1185.

[8] R. Braden, V. Jacobson, and L. Zhang. RFC 1185: TCP extensions for high-speed paths, October 1990. Obsoleted by RFC1323 [7].

[9] Kevin Brown and Suresh Singh. A network architecture for mobile computing. In *Proc. IEEE INFOCOM'96*, pages 1388–1396, March 1996.

[10] Ramon Caceres and Liviu Iftode. Improving the performance of reliable transport protocol in mobile computing environment. *IEEE Journal of Selected Areas in Communications*, 13(5), June 1995.

[11] IP Mobility Working Group. IP mobility support. In Charles E. Perkins, editor, *Internet Draft*, April 1996.

[12] T.V. Lakshman and Upamanyu Madhow. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. *IEEE/ACM Transactions on Networking*, 5(3), June 1997.

[13] Steven McCanne, Sally Floyd, and Kevin Fall. Network simulator. Public domain software, http://www-nrg.ee.lbl.gov/ns/.

[14] Karunaharan Ratnam and Ibrahim Matta. Effect of local retransmission at wireless access points on the round trip time estimation of TCP. *IEEE 31st Annual Simulation Symposium '98*, April 1998. http://www.cs.bu.edu/fac/matta/Papers/ss98.ps

[15] Karunaharan Ratnam and Ibrahim Matta. WTCP: An efficient mechanism for improving TCP performance over wireless links. *Third IEEE Symposium on Computer and Communications (ISCC '98)*, June 29 - July 2 1998. http://www.cs.bu.edu/fac/matta/Papers/iscc98.ps

[16] W. Richard Stevens. *TCP/IP Illustrated* The Protocols. Addison-Wesley, Reading, MA, USA, 1994.

[17] TIA. The cdma2000 itu-r rtt candidate submission, June 1998.

[18] TIA/EIA-95B. Mobile station-base station compatibility requirements for dual-mode wideband spread spectrum cellular systems, July 1997.

[19] Gary Wright and W. Richard Stevens. *TCP/IP Illustrated: The Implementation*. Addison-Wesley, Reading, MA, USA, 1995.