Discriminating Congestion Losses from Wireless Losses using Inter-Arrival Times at the Receiver*

Saad Biaz† Nitin H. Vaidya
Department of Computer Science
Texas A&M University
College Station, TX 77843-3112, USA
E-mail: {saadb,vaidya}@cs.tamu.edu

Abstract

We present a simple scheme which enables a TCP receiver to distinguish congestion losses from corruption losses. The scheme works in the case where the last hop to the receiver is a wireless link and has the smallest bandwidth among all links on the connection path. We added our mechanism to TCP-Reno *to evaluate the performance improvement. We compared our scheme against* Ideal TCP-Reno *which is* TCP-Reno *that can perfectly (but artificially) distinguish between congestion losses and wireless transmission losses. Under favorable conditions, our scheme performs similar to* Ideal TCP-Reno *and can lead to significative throughput improvement.*

1 Introduction

TCP is a popular protocol for reliable data delivery in the Internet. TCP is robust in that it can adapt to disparate network conditions [11]. In recent years, wireless environments with transmission errors are becoming more common. Therefore, there is significant interest in using TCP over wireless links [15, 4, 8, 5, 4, 3, 9]. Previous work has shown that, unless the protocol is modified, TCP may perform poorly on paths that include a wireless link subject to transmission errors. The reason for this is the implicit assumption in TCP that all packet losses are due to congestion. Whenever a TCP sender detects a packet loss, it activates congestion control mechanisms [11] (these mechanisms reduce sender's window in response to the packet loss, reducing throughput temporarily). Taking congestion control actions may be appropriate when a packet loss is due to congestion, however, it can unnecessarily reduce throughput if packet losses happen to be due to wireless transmission errors.

Past proposals for improving performance of TCP over wireless require some cooperation from an intermediate node on the path from the sender to the receiver [3, 4, 5, 16, 2, 10]. Our interest is in mechanisms that do not require intermediate hosts (i.e., any host other than the sender or the receiver) to take any TCP-specific actions. Such mechanisms are particularly useful when the IP traffic is encrypted, or when incorporating TCP-awareness in intermediate nodes is not feasible.

Ideally, it would help if the sender could differentiate between packet losses due to congestion from the packet losses due to wireless transmission errors, using some end-to-end technique (that does not get any help from any intermediate host). Once a sender knows that the packet loss is due to congestion or corruption, it can respond appropriately. We previously [7] tried to adapt some well-known congestion avoidance schemes to enable the sender to distinguish between the two types of packet losses. That approach did not always yield good results. With the cumulative acknowledgement used by TCP, the sender does not know exactly which packets are lost. The receiver has a better view of the losses : it knows exactly which packets are lost. This observation led us to consider schemes which can enable the receiver to distinguish between congestion losses and transmission error losses.

In this paper, we present our scheme and measure its ability to distinguish congestion error losses from wireless transmission error losses. We use simulation to study this ability of discrimination. Then, we modify TCP-Reno to integrate our scheme and study the throughput enhancement induced. The *TCP-Reno* modified with our scheme will be called *TCP-Aware*. We compare the performance of our

*Research reported is supported in part by the Fulbright Program, the National Science Foundation grants MIP-9423735 and CDA-9529442, and the Texas Advanced Technology Program grants 010115-248 and 009741-052-C.

†On leave from the Ecole Superieure de Technologie Fes (MOROCCO).

scheme, *TCP-Aware*, with an *Ideal TCP-Reno* sender which can perfectly distinguish congestion losses from wireless transmission losses. The *Ideal TCP-Reno* sender is implemented artificially such that the TCP sender has a perfect knowledge of the real cause of a packet loss.

This paper is organized as follows. Section 2 presents the proposed scheme. Simulation model and simulation results on the ability of our scheme to distinguish losses are discussed in Section 3. Section 4 is dedicated to the performance study of TCP using our scheme. Conclusions and further work are presented in Section 5.

2 The Proposed Scheme

In this section, we first present the model of environment in which our scheme works well and then describe our scheme and show why such an environment is favorable.

2.1 The Model

TCP connections may traverse wireless links in several scenarios, as illustrated in Figure 1 - a solid line in this figure depicts a wired link, and a dashed line depicts a wireless link. In Figure 1(a), only the last hop is wireless; this scenario typically occurs when a mobile host communicates with a fixed host. In this case, the mobile host is connected with a *base station* via a wireless link. As shown in Figure 1(b), intermediate links may also be wireless. This situation can occur, for instance, when the path includes a satellite hop. Other scenarios shown in Figure 1 also occur in practice. In this paper, we consider only the scenario shown in Figure 1(a) where the sender is on the wired network and the receiver is connected via the wireless link. Moreover, we assume that the wireless link is the bottleneck for the connection. The assumption that the wireless link is the bottleneck is often valid in cellular environments. If the

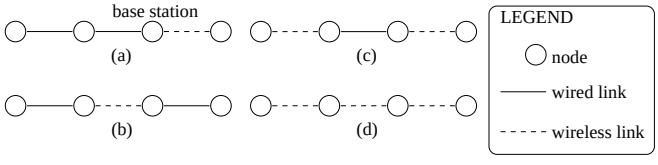

Figure 1. Typical scenarios

wireless link is the bottleneck for the connection, then the packets tend to queue up at the base station. Therefore, *most of the packets are sent back to back* on the wireless link. This last characteristic is the key to the simple heuristic we developed to distinguish packet losses due to transmission errors from congestion losses on the wired network. We now describe our heuristic.

2.2 Packet Inter-Arrival Time as the Discriminator

We describe in this section a scheme which allows the receiver to discriminate between transmission losses and congestion losses. For the sake of simplicity, we assume that the processing time at the base station and at the receiver is negligible. We also assume that:

- only the last link on the path is wireless.
- the wireless link is the bottleneck for the connection
- the sender performs a *bulk* data transfer

Because the wireless link is the bottleneck, the base station will typically buffer more than one packet that is destined to the receiver. Now consider three situations, as illustrated in Figure 2. The figures show three scenarios that may occur when the sender sends packets 1, 2 and 3 to the receiver. In

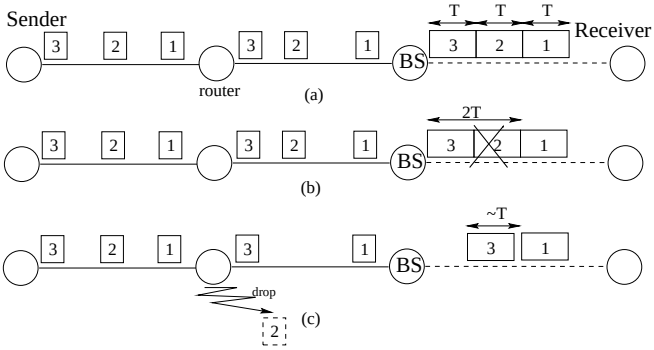

Figure 2. Inter-arrival gap

Figure 2(a), none of the packets are lost[1]. Therefore, the receiver receives all of the packets. In this case, the "packet inter-arrival gap", or the time between *arrival* of consecutive packets, is approximately equal to the time T required to transmit one packet on the wireless link - in practice, the inter-arrival gap can vary due to other factors, including queueing delay. Note that the time of arrival of a packet is the time when all bits belonging to the packet have been received by the receiver. Now, consider Figure 2(b) which shows that packet 2 is lost when transmitted over the wireless link. In this case, the time between the *arrival* of the two packets received by the receiver (i.e., packets 1 and 3) is $2T$, since packet 2 (lost due to transmission) uses the wireless link for T time units. Now consider the third possibility shown in Figure 2(c). Here, assume that packet 2 is lost due

[1] Note that the sender sends packets 1, 2 and 3 at irregular intervals. At the router, the delay between packets may be modified due to the cross traffic and the queueing policy of the router. But, at the base station, the packets are sent back to back because the wireless link is the slowest link along the path for the connection.

to queue overflow (congestion) at the intermediate router. In this case, the inter-arrival gap between packets 1 and 3 will be comparable to T (this holds true if packet 3 arrives at the base station either before, or just after, the base station has transmitted packet 1). When packet 2 is lost and packet 3 arrives at the receiver, packet 3 is called an *out-of-order packet*. Based on above observations, we have developed the following heuristic:

- Let T_{min} denote the minimum inter-arrival time observed so far by the receiver during the connection.
- Let P_o denote an out-of-order packet received by the receiver. Let P_i denote the last in-sequence packet received before P_o. Let T_g denote the time between arrivals of packets P_o and P_i. Finally, let the number of packets missing between P_i and P_o be n (assuming that all packets are of the same size).

  If $(n+1)T_{min} \leq T_g < (n+2)T_{min}$, then the n missing packets are assumed to be lost due to wireless transmission errors. Otherwise, the n missing packets are assumed to be lost due to congestion.

Note that the condition for identifying a packet loss as a wireless loss is quite restrictive. The reason is that it is preferable, for the sake of the network, to mistake a wireless loss for a congestion loss, rather than the opposite. Of course, there are many scenarios for which this heuristic will be defeated. Our preliminary measurements show that our heuristic works in the case when the wireless link has the lowest bandwidth.

To evaluate our scheme, we will measure the accuracy of discrimination of our heuristic. We use two simple and natural metrics :

A_c : accuracy of congestion loss discrimination

A_w : accuracy of wireless loss discrimination.

A_c is defined as the ratio of the number of congestion losses correctly identified over the total number of congestion losses. For instance, if 100 congestion losses occurred and our heuristic identified them correctly 75 times, then the accuracy A_c is 0.75. A_w is similarly defined, but for wireless transmission error losses.

3 Performance Evaluation

3.1 Simulation Model

We evaluate our scheme using the simulation tool *ns-2* (version 2.1b1) [1] from Berkeley. Figure 3 depicts the topology used. The topology is simple and yet serves our purpose. We have a TCP connection from the *fixed host* to the *wireless host*. We use the $Reno$ agent from *ns-2* for the TCP connection. This connection shares the link $R_1 \longleftrightarrow R_2$ with a cross traffic issued by four $Traffic/Expoo$ [1] agents. The traffic flows from the sources CBR_i to the sinks $SINK_i$ $(i = 0, 1, 2, 3)$. The $Traffic/Expoo$ agent from *ns-2* [1] is a constant-bit rate (CBR) source with idle time and busy time exponentially distributed with mean 0.1 sec. UDP is used for this type of source. All the links in Figure 3 are labeled with a *(bandwidth, propagation delay)* pair. Note that propagation delay does *not* include transmission time or queueing delays. The link between the router R_2 and the wireless host is wireless. This link has a transmission error rate r_w which will take the values from 1% to 5%. r_w is measured as a fraction (or percentage) of packets lost due to wireless transmission errors. In this paper, we set the propagation time from R_2 to the wireless host to 1 ms. Similarly, the four links from $R_2 \longrightarrow SINK_i$ have a propagation time of 1 ms. All other links are wired with a bandwidth bw_1 and a propagation delay δ. The propagation delay δ takes the values 1 ms, 8 ms and 18 ms such that the round trip propagation time varies from 6 ms to 74 ms.

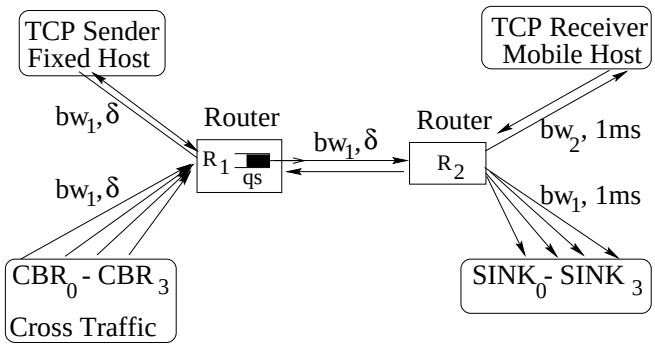

Figure 3. Simulation Model

3.2 Methodology

The objective is to set a TCP connection between a fixed host and a wireless host. This connection has to contend with a random cross traffic to share the link $R_1 \longrightarrow R_2$. This contention will generate congestion losses. Let r_c denote the congestion loss rate for the TCP connection. For each set of parameters r_w, r_c, bw_1, bw_2, and δ, the simulation has two phases. The first phase is used to determine the right rate pkr for the $Traffic/Expoo$ agent to produce a given congestion loss rate r_c. When the rate pkr is known, the second phase begins : we set the rate for the $Traffic/Expoo$ agent to pkr and run, 10 times, one long-lived TCP connection (300 to 1000 seconds). Note that each simulation starts with a warm-up period of 100 sec during which only the CBR sources are active. After the warm up

period, the TCP connection starts after a random period between 0 and 20 ms. For each run, we measure the accuracies of discrimination A_c and A_w.

3.3 Simulation Results

We conducted simulations varying these parameters :

- the congestion loss rate r_c takes values 1% to 5%
- the transmission error loss rate r_w takes values 1% to 5%
- the wired bandwidth bw_1 takes values 64 Kbits/s, 128 Kbits/s, 256 Kbits/s, 512 Kbits/s, 1 Mbits/s and 2 Mbits/s
- the wireless bandwidth bw_2 takes values 64 Kbits/s, 128 Kbits/s, 256 Kbits/s, 512 Kbits/s, 1 Mbits/s and 2 Mbits/s.
- the round trip propagation T_p time takes values 6ms, 34 ms and 74 ms.

For each set of parameters r_c, r_w, bw_1, bw_2, and T_p, we measured the accuracy A_c for congestion losses and the accuracy A_w for wireless transmission errors. Figures 4 and 5 present the accuracies A_c and A_w with $r_c = 1\%$ and $r_w = 1\%$. Figure 4 plots the accuracy A_c of congestion losses for 4 differents values (64 Kbits/s, 256 Kbits/s, 512 Kbits/s,and 2048 Kbits/s) for the bandwidth bw_2 on the wireless link. The four curves in Figure 4 corresponds to the four values of bw_2. The x-axis, with a logarithmic scale (base 2), represents the bandwidth bw_1 on the wired links. Figure 5 plots similarly the accuracy A_w of wireless transmission errors. Two factors determine the accuracies A_c

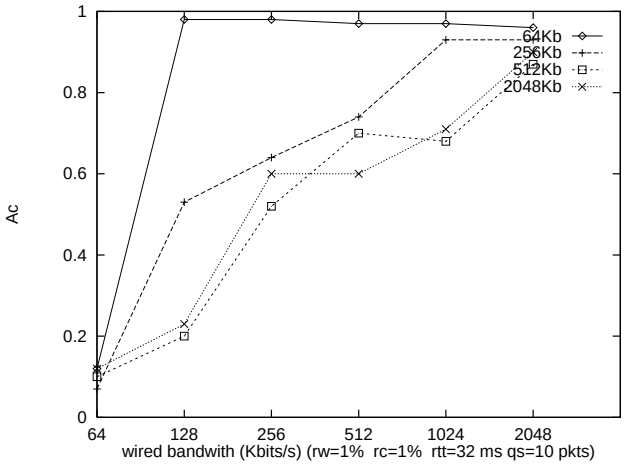

Figure 4. Accuracy A_c (r_c=1%, r_w=1%)

and A_w :

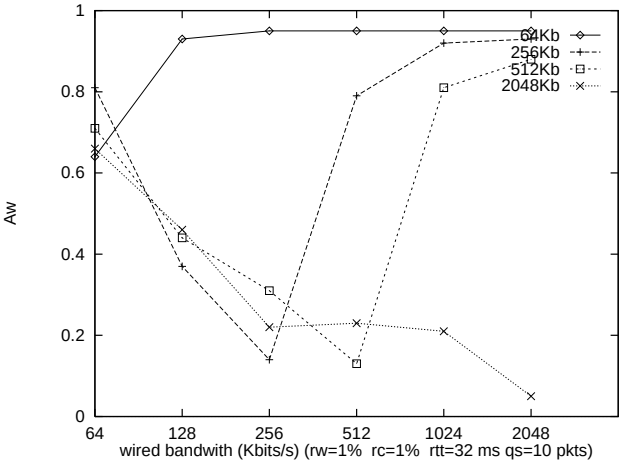

Figure 5. Accuracy A_w (r_c=1%, r_w=1%)

- the ratio, $\frac{bw_1}{bw_2}$,
- the overall loss rate $r_c + r_w$ when $\frac{bw_1}{bw_2} < 1$

$\frac{bw_1}{bw_2} > 1$: When $\frac{bw_1}{bw_2} > 1$, the accuracy of transmission losses A_w increases as the ratio $\frac{bw_1}{bw_2}$ increases. As we noted earlier, the larger is the ratio $\frac{bw_1}{bw_2}$ and larger will be the likelihood of packets buffering at the basestation. Therefore the scheme is more efficient as the ratio $\frac{bw_1}{bw_2}$ increases.

$\frac{bw_1}{bw_2} \simeq 1$: Now, if the ratio $\frac{bw_1}{bw_2}$ is close to one then the packets can be equally likely at the sender, at the router R_1, or at the base station R_2 output buffers. Therefore, it is rare that packets are sent back to back from the base station. In this case, the accuracy A_w losses is small.

$\frac{bw_1}{bw_2} < 1$: The packets are buffered at the sender and the router R_1 output buffers. The link $R_1 \longrightarrow R_2$ is the bottleneck. In this case, the minimum inter-arrival time is equal to the service time at router R_1. A wireless loss cannot be diagnosed reliably and accurately because the inter-arrival does not depend on the transmission time over the wireless link. Moreover, congestion losses may be mistakenly diagnosed as wireless losses. We show, in the following, that the smaller is the overall loss rate $r_c + r_w$, larger will the fraction of congestion losses mistakenly diagnosed as wireless loss.

In Figure 6, we consider an overflow situation at a router. Cross traffic packets are black, TCP packets are white and numbered 1, 2, and 3 (other "dashed" packets in the queue are from other traffic not of interest here). The TCP packet

number 2 is dropped because of queue overflow. Figures 6(a) and 6(b) differ only by the number of cross traffic packets between the two TCP packets 1 and 3. In Figure 6(a), there is only one cross traffic packet between TCP packets 1 and 3. In this case, our criteria may mistake this congestion loss for a wireless loss because the interarrival between packets 1 and 3 is more than twice the transmission time. If packets 2 and 3 are sent back to back, the loss will definitely be mistaken as a wireless loss. In Figure 6(b), there are two cross traffic packets between TCP packets 1 and 3. The interarrival time between packets 1 and 3 at the receiver is larger than three times the service time at router R_1. In this case, the loss of TCP packet 2 will be diagnosed correctly (This remains valid for 2 or more cross traffic packets between TCP packets 1 and 3). The case depicted in Figure 6(a) is more likely to happen than the second case depicted in Figure 6(b) if the cross traffic is lighter than the TCP traffic, i.e., if the congestion loss rate r_c is low or the overall loss rate $r_c + r_w$ is low. This is clearly illustrated by the plots in Figures 4 and 5. In Figures 4 and 5 ($r_c = 1\%$ and $r_w = 1\%$), we observe that, when the ratio $\frac{bw_1}{bw_2}$ gets smaller, A_w increases and A_c decreases. This is due to the fact that most congestion losses are mistakenly diagnosed as wireless losses.

Now, if a packet p is lost on the wireless link, this loss will be diagnosed as wireless provided that packets 1, 2 and 3 are sent back to back on the link $R_1 \longrightarrow R_2$. Therefore, most of the losses (due to congestion or transmission) are diagnosed as transmission losses.

When the congestion loss rate r_c increases or if the overall loss rate $r_c + r_w$ increases, the situation in Figure 6(b) is more likely to happen than the one in Figure 6(a). Therefore, most losses are diagnosed as congestion losses when the ratio $\frac{bw_1}{bw_2} < 1$. With $r_c = 5\%$ and $r_w = 1\%$, in Figures 7 and 8, the overall loss rate for the TCP connection is 6%. When the ratio $\frac{bw_1}{bw_2} < 1$, most losses are diagnosed as congestion losses. Therefore, A_c is high and A_w is low. When the ratio $\frac{bw_1}{bw_2} > 1$, the accuracy A_w for the wireless losses increases as the ratio $\frac{bw_1}{bw_2}$ increases. The accuracy

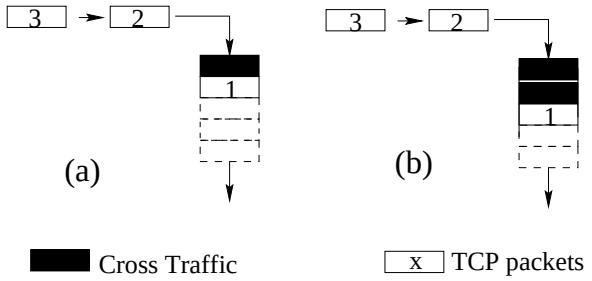

Figure 6. Congestion losses

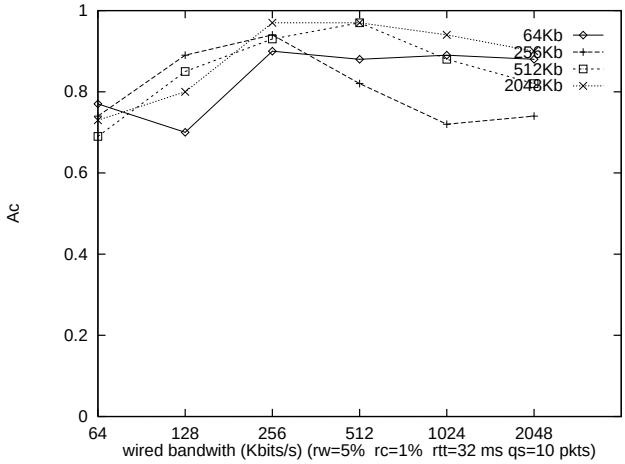

Figure 7. Accuracy A_c (r_c=5%, r_w=1%)

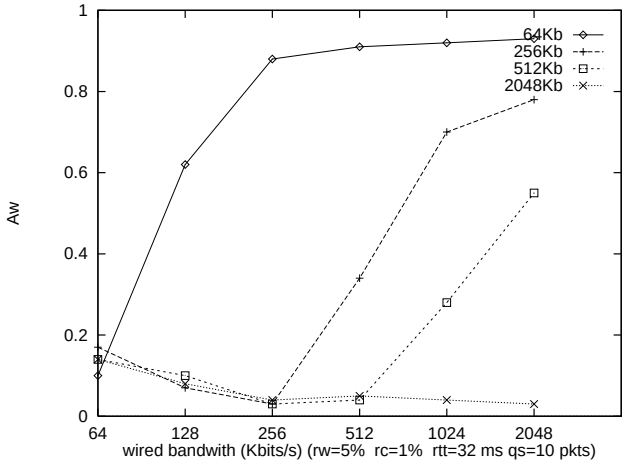

Figure 8. Accuracy A_w (r_c=5%, r_w=1%)

A_c is also high because the scheme, in these conditions, is efficient and very selective for wireless losses. When $r_c = 1\%$ and $r_w = 5\%$, the plots (which can be found in [6]) are similar to the plots in figures 7 and 8. This show that the ratio $\frac{bw_1}{bw_2}$ and the overall loss rate are determinant. When $r_c = 5\%$ and $r_w = 5\%$, the plots which can be found in [6] are similar to the plots in figure 7 and 8. We observe a slight decrease in A_w which can be explained by the small average congestion window size. The average congestion window size has a direct impact on the number of packets buffered at the basestation.

Finally, when $r_c = 3\%$ and $r_w = 3\%$, the overall loss rate is the same as the overall loss rate in figures 7 and 8. The plots, which can be found in [6], are very similar, confirming that the overall loss rate is determinant.

4 TCP Performance Using our Scheme

In this section, we evaluate through simulation the throughput improvement when using our scheme with TCP-Reno for long lived TCP connections. Hereafter, TCP-Reno modified by our scheme will be designated as *TCP-Aware*. We compare the performance of our scheme against the performance of an *ideal TCP-Reno* sender. The *ideal TCP-Reno* sender is a TCP sender which has artificially a perfect knowledge of the real cause of a packet loss.

4.1 Simulation model

We use a topology similar to the one in Figure 3. The only difference is that we have 4 fixed hosts and 4 wireless hosts as depicted in Figure 9. There is one TCP connection between each pair of fixed and wireless hosts. The 4 TCP connections share the link $R_1 \longleftrightarrow R_2$ with a cross traffic issued by four $Traffic/Expoo$ [1] agents which are exactly the same as in Section 3. The links between the router R_2 and the wireless hosts are wireless. These links have a bandwidth bw_2 which takes values 64 Kbits/s, 256 Kbits/s, 1 Mbits/s, 2 Mbits/s, and 8 Mbits/s. This link has a transmission error rate r_w which will take the values from 1%, 3%, and 5%. The bandwidth bw_1 takes values 64 Kbits/s, 256 Kbits/s, 1 Mbits/s, 2 Mbits/s, and 8 Mbits/s. The propagation delay δ takes the value 12 ms such that the round trip propagation time takes a value of 50 ms. This represents a typical value of round trip propagation time on WANs.

4.2 Methodology

The objective is to set 4 TCP connections between 4 fixed hosts and 4 wireless hosts. These connections have to contend against each other and against the random cross traffic to share the link $R_1 \longrightarrow R_2$. This contention will generate congestion losses.

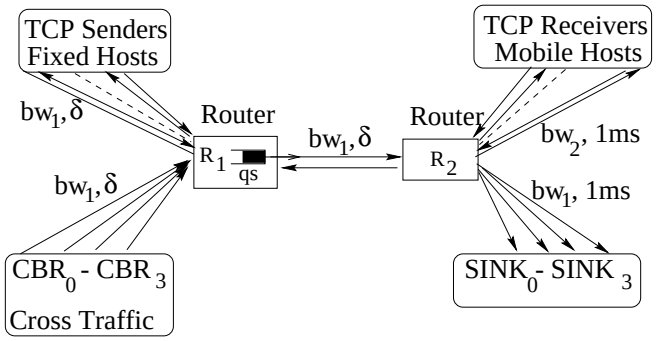

Figure 9. Simulation model

We study the performance of our scheme under light ($r_c = 1 - 2\%$), mild ($r_c = 3 - 4\%$) and heavy ($r_c \geq 5\%$) congestion. We set the level of congestion by choosing the rate pkr of the random sources simulated by the $Traffic/Expoo$ agents. We set the rate pkr of the random sources such that the overall random cross traffic represents 10% (light congestion), 60% (mild congestion), or 120% (heavy congestion) of the bandwidth of the bottleneck link $R_1 \longrightarrow R_2$. For each set of parameters r_w, bw_1, and bw_2, we start the random sources for a warm-up period of at least 100 seconds before opening the TCP connections. Each TCP connection starts independently and randomly between 0 to 20 milliseconds after the end of the warm-up period. Each TCP connection lasts between 300 to 1000 seconds depending on the values of the bandwidths bw_1 and bw_2. We perform 30 such simulations for TCP-Reno, *TCP-Aware*, and *Ideal TCP-Reno*. For each version of TCP (TCP-Reno, *TCP-Aware* and *Ideal TCP-Reno*), we compute the average throughput of all connections over all 30 simulations. In this paper, we report the ratio R_A of the average throughput obtained using *TCP-Aware* over the average throughput obtained with *TCP Reno*. This ratio measures the performance improvement using our scheme (if any). In order to evaluate this improvement, we compute also the ratio R_I of the average throughput obtained using *Ideal TCP-Reno* over the throughput obtained using *TCP Reno*. The *ideal TCP-Reno* sender has perfect knowledge of the cause of a loss. Therefore, the ratio R_I represents the highest improvement we can bring to *TCP-Reno* by perfectly distinguishing congestion losses from wireless transmission losses. The next section presents our results.

4.3 Simulation Results

Objective of our simulation experiments is two-fold: (a) determine the magnitudes of improvement achieved using

TCP-Aware and *Ideal TCP-Reno* through R_A and R_I, and (b) determine the variations in these metrics R_A and R_I as a function of network parameters (such as bw_1, bw_2, r_w and the level of congestion). For lack of space, we present only the set of results where the wireless transmission error rate r_w is held constant at 1%. We present two plots. For each plot, we held the congestion level constant: the congestion level may be mild ($r_c = 3-4\%$), or heavy ($r_c \geq 5\%$). Plots for light congestion ($r_c = 1-2\%$) are similar to those with mild congestion.

4.3.1 Wireless Transmission Error Loss $r_w = 1\%$

Figures 10 and 11 presents results for the case when the wireless transmission error rate r_w is held constant at 1%. For each graph, the x-axis represents the wired bandwidth bw_1. Each graph displays 4 plots :

- two plots for R_A (*TCP-Aware*) where bw_2 is held constant at 256 Kbits/s and 1 Mbits/s, respectively.
- two plots for R_I (*Ideal TCP-Reno*) where bw_2 is held constant at 256 Kbits/s and 1 Mbits/s, respectively.

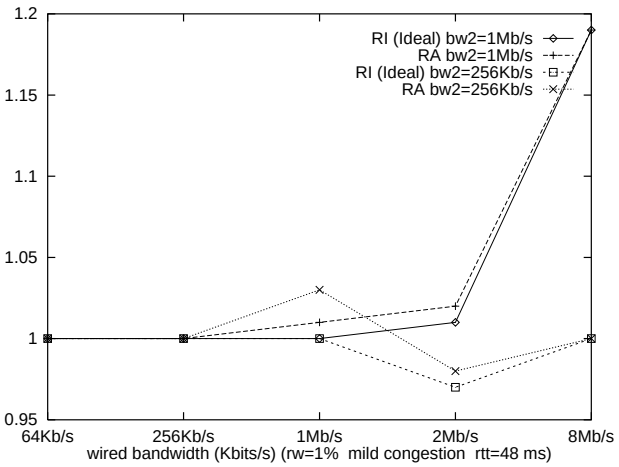

Figure 10. Performance improvement with $r_w = 1\%$ for mild congestion

The plots for light congestion, which can be found in [6], are similar to those for mild congestion.

We observe that the improvement is marginal for both *TCP-Aware* and *Ideal TCP-Reno* senders when we have light to mild congestion. There is no significant improvement when congestion is heavy.

Two factors are determinant in the performance improvement we can expect from the *Ideal TCP-Reno*. The first factor is the ratio $\frac{r_w}{r_c}$. The second factor is the bandwidth-delay product.

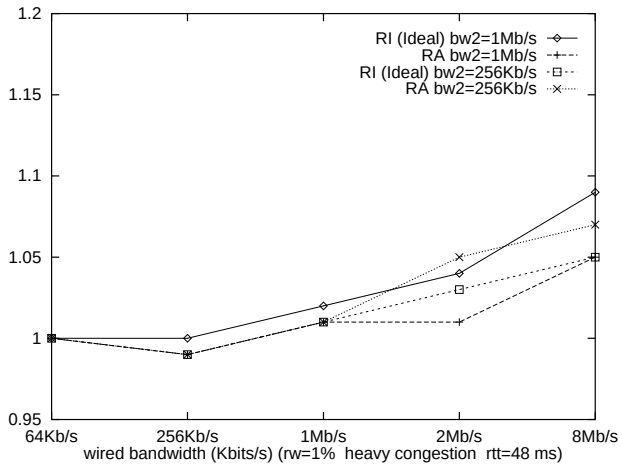

Figure 11. Performance improvement with $r_w = 1\%$ heavy congestion

An approximation of the long range throughput derived in [12, 14] can help explaining qualitatively the results and the general trends of our results. A simpler derivation of this approximation can be found in [13]. The long range throughput T can be approximated as

$$T = \frac{MSS}{RTT} \frac{C}{\sqrt{p}}$$

where MSS is the maximum segment size, RTT is the round trip time (assumed constant), p is a constant probability of random loss, and C is a constant. This approximation is based on the fact that the congestion window size is halved after every packet loss, neglecting the details of TCP data recovery and retransmission. If a TCP connection experiences congestion losses with rate r_c and wireless transmission losses with rate r_w, its long range throughput T_{TCP} may be approximated with $T_{TCP} = \frac{MSS}{RTT} \frac{C}{\sqrt{r_c + r_w}}$.

Now, the *Ideal TCP-Reno* halves its congestion window only when a congestion loss occurs. Therefore, its long range throughput T_{Ideal} may be approximated as $T_{Ideal} = \frac{MSS}{RTT} \frac{C}{\sqrt{r_c}}$. Therefore the ratio of improvement may be approximated as

$$\frac{T_{Ideal}}{T_{TCP}} = \sqrt{1 + \frac{r_w}{r_c}}$$

This approximation shows that the improvement increases with r_w and decreases with r_c. The ratio $\frac{r_w}{r_c}$ is a determinant factor of the improvement we can expect from an *Ideal TCP-Reno*. Recall that in Figures 10 and 11, the wireless transmission error loss rate r_w is held constant at 1%. As congestion increases, the improvement induced by *Ideal TCP-Reno* decreases. Moreover, when congestion is

low ($r_c = 1 - 2\%$), the approximation implies that the improvement should be between 1.22 and 1.4. The plots on Figure 10 and 11 for low congestion show that the improvement remains below these values. This is due to the fact that the bandwidth-delay product is low.

The second factor affecting performance is the bandwidth delay product. When the bandwidth-delay product is small, the *Ideal TCP-Reno* cannot achieve significant performance improvement. To illustrate this, let us consider the plots for which $bw_2 = 256Kbits/s$. For all the plots presented, the round trip propagation delay is 48 ms. Therefore, the bandwidth-delay product is at most 1536 bytes ($\simeq 1.5$ packet with packet size of 1000 bytes). There is no need of high window size to keep the pipe full. However, a high window size may increase the likelihood of triggering the fast-retransmit mechanism (avoiding time-outs). When $bw_2 = 256$ Kbits/s, we observe on all plots that the improvement due to *Ideal TCP-Reno* is low (less that 1.05). When $bw_2 = 1$ Mbits/s, the improvement in throughput may be significant (more than 3). Note that our scheme *TCP-Aware* performs similar to *Ideal TCP-Reno*. Sometimes, the performance of our protocol is actually better than *Ideal TCP-Reno* – this difference is due to two factors: (a) statistical variations, and (b) our protocol does not back off for all congestion losses when $A_c < 1$.

Results and discussions for $r_w = 3\%$ and $r_w = 5\%$ are similar to those with $r_w = 1\%$. They are omitted here for lack of space and can be found in [6].

5 Conclusion and further work

We studied a simple heuristic to distinguish between packet losses due to wireless transmission error from packet losses due to congestion. This heuristic works best when (i) last hop for the connection is wireless, (ii) the bandwidth of the wireless link is much smaller than the bandwidth of the wired link, and (iii) the overall packet loss rate is small. These conditions are often true in practice, particularly in cellular environments.

We added our scheme to *TCP-Reno* to distinguish the two types of losses. We compared through simulations the performance of our scheme with an unimplementable *Ideal TCP-Reno* which can perfectly, but artificially, diagnose the cause of a packet loss. The preliminary results with 4 TCP connections sharing a common bottleneck show that our scheme performs similarly to *Ideal TCP-Reno* except when the wireless error loss is high and congestion is not heavy. Now, we plan to use this heuristic with TCP and study the impact on the TCP performance under different scenarios (different round trip times, queue size, etc.). We aim to refine this heuristic to work with a shared wireless link with many connections competing for the wireless link as in Wavelan.

References

[1] VINT project U.C. berkeley/LBNL, ns2:network simulator. http://www-mash.cs.berkeley.edu/ns/.

[2] M. Allman and D. Glover. Enhancing TCP over satellite channels using standard mechanisms, May 1998. INTERNET DRAFT, http://gigahertz.lerc.nasa.gov/ mallman/papers.

[3] A. Bakre and B. Badrinath. I-TCP: Indirect TCP for mobile hosts. In *Proc. 15th International Conf. on Distributed Computing Systems (ICDCS)*, May 1995.

[4] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz. A comparison of mechanisms for improving TCP performance over wireless links. In *ACM SIGCOMM'96*, Aug. 1996.

[5] H. Balakrishnan, S. Seshan, and R. Katz. Improving reliable transport and handoff performance in cellular wireless networks. *ACM Wireless Networks*, 1(4), Dec. 1995.

[6] S. Biaz and N. H. Vaidya. Discriminating congestion losses from wireless losses using inter-arrival times at the receiver. Technical Report 98-014, CS Dept., Texas A&M University, June 1998. Revised August 1998.

[7] S. Biaz and N. H. Vaidya. Distinguishing congestion losses from wireless transmission losses : A negative result. In *IEEE 7th Int'l Conf. on Computer Communications and Networks*, Oct. 1998.

[8] R. Caceres and L. Iftode. Improving the performance of reliable transport protocols in mobile computing environments. *IEEE JSAC Special issue on Mobile Computing Networks*, 13(5), June 1995.

[9] A. DeSimone, M. Chuah, and O. Yue. Throughput performance of tranport-layer protocols over wireless lans. In *Proc. Globecom '93*, Dec. 1993.

[10] M. A. et. al. Ongoing TCP research related to satellites, Mar. 1998. INTERNET DRAFT.

[11] V. Jacobson. Congestion avoidance and control. In *ACM SIGCOMM'88*, pages 314–329, Aug. 1988.

[12] T. Lakshman and U. 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] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behavior of the tcp congestion avoidance algorithm. *Computer Communication Review*, 27(3), July 1997.

[14] T. J. Ott, J. Kemperman, and M. Mathis. The stationary behavior of ideal tcp congestion avoidance, Aug. 1996. In progress,Obtain via pub/tjo/TCPwindow.ps using anonymous ftp to ftp.bellcore.com.

[15] J. Postel. Transmission control protocol, Sept. 1988. RFC 793.

[16] R. Yavatkar and N. Bhagwat. Improving end-to-end performance of TCP over mobile internetworks. In *Workshop on Mobile Computing Systems and Applications*, Dec. 1994.