Effectiveness of Loss Labeling in Improving TCP Performance in Wired/Wireless Networks*

Dhiman Barman Ibrahim Matta
Computer Science Department
Boston University
Boston, MA 02215, USA
{dhiman, matta}@cs.bu.edu

Technical Report BUCS-TR-2002-016

Abstract

The current congestion-oriented design of TCP hinders its ability to perform well in hybrid wireless/wired networks. We propose a new improvement on TCP NewReno (NewReno-FF) using a new loss labeling technique to discriminate wireless from congestion losses. The proposed technique is based on the estimation of average and variance of the round trip time using a filter called Flip Flop filter that is augmented with history information. We show the comparative performance of TCP NewReno, NewReno-FF, and TCP Westwood through extensive simulations. We study the fundamental gains and limits using TCP NewReno with varying Loss Labeling accuracy (NewReno-LL) as a benchmark. Lastly our investigation opens up important research directions. First, there is a need for a finer grained classification of losses (even within congestion and wireless losses) for TCP in heterogeneous networks. Second, it is essential to develop an appropriate control strategy for recovery after the correct classification of a packet loss.

Keywords: TCP; Congestion Control; Error Control; Loss Labeling (Classification); Wireless Links; Simulation.

1 Introduction

The Transmission Control Protocol (TCP) has been the dominant transport mechanism for reliable data transfer over the Internet. While the Internet is growing in size and becoming increasingly heterogeneous, network designers are faced with the challenging question of how to empower TCP so it works well in such hybrid wired/wireless environment [23], where packets can be lost because of various reasons. Many studies have shown that TCP throughput can be improved if the cause of a packet loss is identified [3]. TCP was originally designed for a wired environment where packets are lost mainly due to congestion (i.e. buffer overflow), and the congestion control algorithms imbedded therein act accordingly. When a TCP connection extends over wireless links, packet losses over such links occur primarily due to channel errors or during handoff. By attributing a packet loss to wireless transmission errors, the TCP source can refrain from taking unnecessary "congestion" control measures. One set of solutions (e.g. I-TCP [2], Snoop [4], WTCP [19]) require support from the base station located at the interface between the wired infrastructure and the wireless access infrastructure. The base station can buffer data packets (or just their sequence

*This work was supported in part by NSF grant ANI-0095988.

numbers) as they are received from a wired source. This information can then be used by the base station to recognize if any of those packets are later lost over the wireless link on their way to destination (the mobile host). These solutions incur the cost of implementation at the base station and some violate the end-to-end semantics of TCP.

In this paper, we are primarily interested in *end-to-end* solutions, i.e. those which do not require any support from the network. Proposed end-to-end solutions differ in the measure(s) they use to infer the cause of a packet loss. These measures may be estimated at the sender without any support from the receiver (e.g. round-trip delay), or may require support from the receiver (e.g. one-way delay or delay variance) [1, 6, 18, 25].

Loss classification can be *implicit* in the congestion control of a protocol. TCP Westwood [9, 15, 24] is such a sender-side modification of TCP Reno which estimates the rate that a connection is getting based on the rate at which the sender receives the ACKs. TCP Westwood uses the estimated bandwidth in setting the congestion window and slow start threshold ($ssthresh$) parameters. This is in contrast to regular TCP implementations where the window size and $ssthresh$ are arbitrarily cut in half whenever a loss is detected [10]. This explicit bandwidth estimation scheme is shown to have a positive impact on the performance of TCP Westwood sources, especially in the presence of random, sporadic losses typical of wireless links or over paths with high bandwidth-delay product.

Many proposals tried to classify the losses *explicitly* through different estimation techniques. In the scheme of Biaz and Vaidya [6], the receiving host measures the interarrival times of packets. Assuming the last hop is wireless and the bottleneck, if the time between received packets is close to the minimum, then a lost packet in-between is assumed to have been lost due to wireless errors and not congestion. The receiving host in the Spike scheme [22] measures one-way delays, and switches to congested (wireless) state as the delay exceeds (drops below) a certain threshold. The ZigZag scheme [7] extends Spike to include the mean and deviation of measured one-way delays as well as number of losses in computing the delay thresholds. Intuitively, the higher the number of losses the higher the threshold beyond which congestion is assumed, i.e. the cause of the loss being wireless errors becomes more likely.

In [20], Samaraweera presents a method, called Non-Congestion Packet Loss Detection (NCPLD), to categorize the nature of the error. It uses the concept of the *knee point* of the throughput-load graph at which the network operates at optimum power. Before the knee point, no congestion is present so an increase in transmission rate causes a corresponding increase in throughput and the round trip delay remains relatively constant. After the knee point, packets need to be queued at the routers resulting in an increase in round trip delay. If the current (measured) round trip delay is less than the delay threshold at the knee point then the packet loss is assumed to be a wireless loss else it is assumed that congestion (buffer overflow) caused the error. It is important to note that NCPLD is conservative in its error categorization scheme, i.e., if slight congestion is present, it would more likely classify this loss as congestion error rather than a wireless loss.

In [8], the authors present an algorithm called Linear Increase/Multiplicative Decrease with History (LIMD/H). It uses explicit support from the receiver to send the loss rate back to the TCP source. Based on this loss rate, the sender estimates the goodput. If the current goodput is below a certain band around the mean, then the cause of a packet loss is assumed to be congestion, otherwise the cause of loss is attributed to wireless errors. LIMD/H backs off its transmission window less conservatively to wireless losses than to congestion-induced losses.

Our Contribution: In this paper, we study the fundamental gains and limits of *explicit* loss labeling techniques. Explicit loss labeling is advantageous as it provides a clean separation between the process of inferring the *cause* of a packet loss and the control (recovery) process that may make use of it. To this end, we consider a generic loss labeling technique for which we can vary its accuracy in correctly attributing

a packet loss to either congestion or wireless error. We refer to this generic technique as *NewReno-LL* since we empower the TCP NewReno version with loss labeling. We have only chosen NewReno as recent measurements show that the majority of TCP implementations are NewReno [17]. *NewReno-LL* is thus used to cover the spectrum of explicit loss labeling techniques described above. Note that each explicit loss labeling technique arrives in a different way at a particular loss labeling accuracy.

We also propose a new explicit loss labeling technique that would result in a certain loss labeling accuracy under a particular configuration. The motivation behind our proposal is that we assume the variation in round trip times (RTT) and the nature of loss are correlated. Therefore, a good estimation of RTT or observed delay can help in making TCP more optimistic, i.e. TCP could react less conservatively to wireless losses. In our estimation technique, we make use of a Flip-Flop filter [12] in estimating the average RTT. We use the 3-sigma rule [14] to account for the variance. Note that there are many techniques to estimate the mean and variance but we are interested in one that is simple and sufficiently accurate. The effectiveness of the Flip Flop filter in filtering out transients and capturing persistent conditions was shown in [12]. In this paper, we re-instantiate the filter to measure RTT and augment it with *history* information so as to distinguish between congestion and wireless losses, specifically to empower TCP NewReno in wired/wireless networks. We henceforth refer to our loss labeling technique as *NewReno-FF*.

We evaluate our loss predictor against a TCP NewReno variant that is equipped with perfect labeling (in short, *NewReno-PL*) to show the limits and gains of error classification/misclassification. Note that NewReno-PL is an instance of NewReno-LL, where the TCP NewReno variant knows *exactly* the cause of loss (wireless or congestion). It is tempting to believe that a TCP version that has perfect knowledge about the nature of loss would perform the best in terms of goodput. Surprisingly, our simulations show that this is not always the case. The reason is that misclassification sometimes makes a TCP version perform better by making it more aggressive. For example, transient (short-term) congestion losses are similar to random wireless losses. Therefore, if such losses are misclassified as wireless losses so TCP does not drastically back off its transmission rate, it turns out that goodput improves. We examine this dilemma by equipping TCP with loss predictors that possess different error classification accuracies.

The rest of the paper is organized as follows. In Section 2, we propose a loss prediction algorithm (NewReno-FF) to predict the cause of a packet loss. We also describe other schemes against which we compare NewReno-FF. In Section 3, we discuss the performance metrics to evaluate the various loss predictors. Section 4 presents an analytical model of our generalized loss labeling technique, NewReno-LL. Section 5 evaluates the performance of loss predictors using the *ns*-2 network simulator [21]. In Section 6, we discuss the performance of the protocols under correlated burst losses. Section 7 discusses directions for future research and concludes the paper.

2 Evaluated Schemes

2.1 TCP NewReno-LL

TCP NewReno inherently has no loss prediction ability; it considers all losses to be congestion losses. We denote by $P[C|C]$ ($P[W|W]$), the probability that a loss predictor classifies a packet loss as congestion (wireless) loss given that it is indeed caused by congestion (wireless) error [5, 7]. Thus for TCP NewReno, $P[C|C] = 1$ and $P[W|W] = 0$. Ideally we want to have a protocol with $P[C|C] \approx 1$ but also high $P[W|W]$ so as to react appropriately based on the type of loss. $P[C|C] \approx 1$ is highly desirable so as not to congest the network. High $P[W|W]$ enables the protocol to avoid taking unnecessary congestion control steps.

To evaluate the limits of TCP NewReno we evaluate it against TCP NewReno equipped with varying loss classification accuracies (NewReno-LL). In NewReno-LL, on a loss event, the sender reduces its congestion window with probability $P[C|C]$ if the loss is a congestion one and *ignores* the window adjustment with probability $P[W|W]$ if it is a wireless loss. In the perfect loss labeling version, NewReno-PL, we have

$P[C|C] = 1$ and $P[W|W] = 1$. We note that, although we assume fixed values for $P[C|C]$ and $P[W|W]$, in general these values depend on the parameters of the network as well as the loss classification algorithm. Furthermore, ignoring window adjustments in the case of wireless losses is not necessarily optimal. We only consider this control action for simplicity and so as to gain insight into the fundamental issues. We can imagine window adjustments that match particular levels of error.

2.2 TCP NewReno-FF

We propose a new loss labeling scheme for distinguishing wireless losses from congestion losses. In this scheme, using an adaptive Flip Flop filter [12], a parallel estimation of RTT is done on every new ACK received in NewReno. TCP usually uses one exponentially weighted moving average (EWMA) filter which is static. The Flip Flop filter uses two EWMA filters, one is stable and another is agile. An agile filter is one which gives more weight to recently observed samples unlike a stable filter. The underlying principle is to employ an agile filter whenever possible but switch to the stable one when the RTT samples vary drastically and become noisy. According to statistical quality control, control limits are defined around the current sample mean and when the samples exceed the control limits, the process is said to be out of control. To estimate the deviation, the filter uses a moving range which it estimates from the samples within the control limits. The control limits are defined as:

$$\overline{x} \pm 3\frac{\overline{MR}}{d_2} \tag{1}$$

where $\overline{x}$ is the sample mean, $\overline{MR}$ is the Moving Range which is the average of the differences between adjacent RTT samples, $|x_i - x_{i-1}|$, and d_2 estimates the standard deviation of a given sample given its range. When the range is from a sample of two, as for MR, $d_2 \approx 1.128$ [14].

The basic tenet of our approach is that if the packets are suffering congestion losses, the observed RTTs will vary but if packets are suffering random losses, the observed RTTs will not vary much. Using the Flip Flop filter, we define an upper control limit on RTT using (1). We then consider the much delayed packets, whose RTT exceeds the control limit, as "outliers." More than η outliers in the last l samples are used as congestion indication. η and l are tunable parameters.

On every ACK of a non-retransmitted packet, we compute the sample RTT as *s_rtt*, the estimated average RTT as *est_rtt*, and the moving range as $\overline{MR}$. To maintain a history, we maintain a bit vector of length l. Every time a sample RTT is calculated, the bit vector is shifted left and the least significant bit is set to 1 if the sample RTT is an outlier. The pseudo code of the loss labeling is as follows:

```
if (s_rtt > est_rtt+3 MR/1.128) then
   vector = vector << 1 // Left shift once
   vector = vector OR 0x01 // Bitwise OR
   est_rtt = 9/10 est_rtt + 1/10 s_rtt //stable RTT filter
else
   vector = vector << 1 // Left shift once
   vector = vector OR 0x00 // Bitwise OR
   est_rtt = 1/10 est_rtt + 9/10 s_rtt // agile RTT filter
   diff = |s_rtt − last_rtt|
   MR = 0.875 MR + 0.125 diff
end if
if (first ack) then
   vector = 0x00; // initialize bit vector of l bits
   est_rtt = s_rtt
   MR = est_rtt / 2
end if
```

Modified Recovery Strategy: When NewReno-FF detects a packet loss based on duplicate ACKs[1], it checks if the sender has received more than η outlier samples in the last l samples. If number of outliers is more than η, it continues with usual congestion control steps otherwise it ignores it assuming a wireless loss. The pseudo code of the modified recovery part of TCP NewReno is as follows:

```
if (error detected based on dup acks) then
  if (#bits set in vector ≤ η) then
    no change in congestion window and ssthresh // classified as wireless loss
  else
    do normal congestion window and ssthresh adjustment // classified as congestion loss
  end if
end if
```

We are still assuming that *all* timeouts indicate congestion, i.e. we do not attempt to classify timeout-detected losses as congestion versus wireless. It is known that timeouts caused by wireless losses can degrade the performance of regular TCP implementations, which may back off very conservatively [23]. We account for these effects through our TCP NewReno variant with perfect loss labeling.

2.3 TCP Westwood

We have also compared the performance of TCP Westwood which doesn't have any explicit loss labeling mechanism. However, it is claimed that the bandwidth estimation of TCP Westwood accounts for the wireless losses [15].

3 Performance Metrics

We have compared the schemes described in Section 2 based on the following metrics:

- Goodput: The rate of delivery of useful data. We measure the goodput at the receiver.
- Overhead: 1-$\frac{\text{Goodput}}{\text{Throughput}}$. Throughput is the rate of transferring the data to a receiver. Note that this measure reflects the end-to-end loss rate.
- $P[W|W]$: This metric indicates the accuracy in wireless loss classification. It defines the probability of identifying a loss as a wireless loss given that the loss is indeed a wireless loss.
- $P[C|C]$: This metric indicates the accuracy of congestion loss classification. It defines the probability of identifying a loss as a congestion loss given that the loss is indeed a congestion loss.
- Fairness: Reflects the fair share distribution across N various connections. It is defined as follows:

$$\text{Fairness Index} = \frac{(\sum_{i=1}^{N} T_i)^2}{N \sum_{i=0}^{N} T_i^2} \tag{2}$$

where T_i is the throughput of the ith connection [11].

[1]TCP detects loss either due to timeout or four consecutive duplicate acknowledgments.

4 Stochastic Model of NewReno-LL

In this section, we derive a simple fluid model for NewReno-LL to illustrate the effect of $P[C|C]$ and $P[W|W]$ on throughput. We follow the same lines of analysis as in [9]. For simplicity, we do not consider timeouts. We assume that the transmission window is reduced by half *only* if the packet loss is identified as congestion-induced.

For a window size of W (packets), it increases by $\frac{1}{W}$ on every ACK reception. This is the *additive increase* of TCP. We assume packet loss happens with a probability $p = 1-(1-p_c)(1-p_w)$, where p_c and p_w are the congestion and wireless drop rates, respectively. The window decreases by $\frac{W}{2}$ with probability $p_c P[C|C] + p_w P[C|W]$, i.e. whenever the packet loss is classified (or misclassified) as congestion-induced. To make the notation more readable, we denote $P[C|C]$ by $p_{c/c}$ and $P[C|W]$ by $p_{c/w}$.

The expected change in the window is given by:

$$\begin{aligned} E[\Delta W] &= \frac{(1-p_c)(1-p_w)}{W} - \frac{W(p_c p_{c/c} + p_w p_{c/w})}{2} \\ &= \frac{(1-p_c)(1-p_w)}{W} - \frac{W(p_c p_{c/c} + p_w(1-p_{w/w}))}{2} \end{aligned} \tag{3}$$

Note that $p_{c/w} + p_{w/w} = 1$.

Since W is updated at approximately every $\frac{RTT}{W}$, using Equation (3), the expected change in the sending rate (throughput) r per unit time is approximately:[2]

$$\frac{dr(t)}{dt} = \frac{(1-p_c)(1-p_w)}{RTT^2} - \frac{r^2(t)(p_c p_{c/c} + p_w p_{c/w})}{2} \tag{4}$$

By rearranging and integrating we have:

$$\int_0^{r(t)} \frac{dr(t)}{\frac{(1-p_c)(1-p_w)}{RTT^2} - \frac{r^2(t)(p_c p_{c/c} + p_w p_{c/w})}{2}} = \int_0^t dt \tag{5}$$

$$r(t) = \frac{1}{RTT}\sqrt{\frac{2(1-p_c)(1-p_w)}{p_c p_{c/c} + p_w(1-p_{w/w})}} \left\{ \frac{1+Ce^{-2at}}{1-Ce^{-2at}} \right\} \tag{6}$$

where a is given by $\sqrt{\frac{(1-p_c)(1-p_w)(p_c p_{c/c} + p_w(1-p_{w/w}))}{2\,RTT^2}}$ and C depends on initial conditions.

The steady state throughput of NewReno-LL is thus given by:

$$r = \lim_{t\to\infty} r(t) = \frac{1}{RTT}\sqrt{\frac{2(1-p_c)(1-p_w)}{p_c p_{c/c} + p_w(1-p_{w/w})}} \tag{7}$$

From Equation (7), we can observe the significance of $p_{c/c}$ and $p_{w/w}$ on the throughput of a connection. A higher value of accurate congestion loss classification, $p_{c/c}$, reduces throughput. On the other hand, a higher value of accurate wireless loss classification, $p_{w/w}$, increases throughput (and hence goodput). Note, however, that we are assuming fixed drop rates and classification probabilities. In reality, these loss and classification probabilities would depend on the number of competing connections, network conditions and the behavior of the protocol. In particular, p_c, assumed fixed in traditional analytical studies [16], depends on $p_{c/c}$, $p_{w/w}$ and r. The throughput r in turn depends on p_c, $p_{c/c}$ and $p_{w/w}$. In general, it is very difficult to solve analytically for closed-form expressions for throughput and other performance measures. In the following, we resort to simulations to evaluate the effectiveness of loss labeling techniques.

[2] Note that $r(t) = W(t)/RTT$.

5 Simulation Model and Methodology

We use the network simulator *ns*-2 (version 2.1b8a) [21]. The network topology used in the simulation is shown in Figure 1.

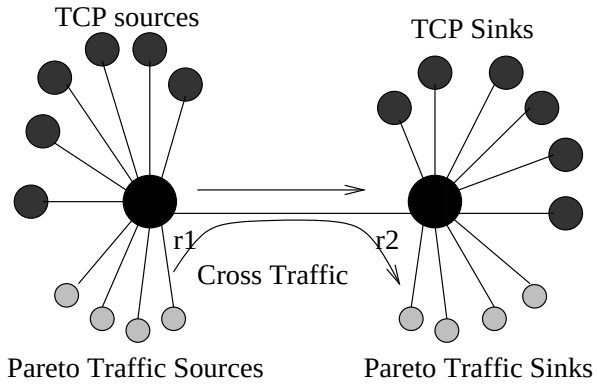


Figure 1: Wireless Last Hop Network Topology setup

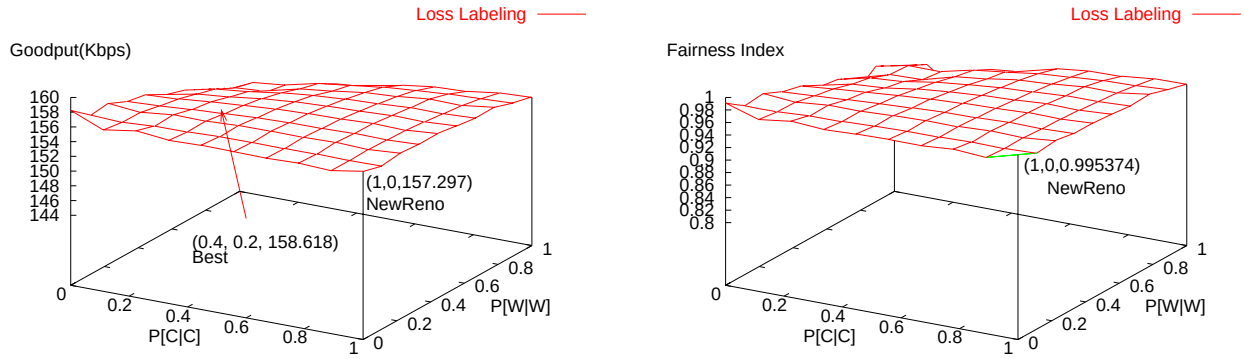
We have a number of TCP traffic source-destination pairs. The link from $r2$ to each TCP traffic sink has been assigned 2Mbps bandwidth and 0.01ms propagation delay. These links represent wireless links with transmission errors. All other links are error free with 10Mbps bandwidth and 1ms propagation delay except the shared (bottleneck) wired link $r1 \leftrightarrow r2$ whose bandwidth is 10Mbps and delay is 50ms. The buffer size at $r1 \leftrightarrow r2$ is equal to the bandwidth-delay product (127 packets) and all other buffer sizes are set to default value of 50 packets. All the TCP sources and the cross traffic on-off sources are started at 0 sec and the simulations are run till 210 sec. For each cross connection, the on and off periods are Pareto distributed with average duration of 100ms each and shape parameter of 2.5. We calculate the performance measures within 95% confidence intervals.

5.1 Simulation Results

Configuration 1: Figure 2 shows the goodput and fairness index and Figure 3 shows the overhead of NewReno-LL against $P[W|W]$ and $P[C|C]$ both varying from 0 to 1.0. The experiments were done under 5% wireless loss as it represents low to medium range wireless errors. We have confirmed that the trend is similar for 1% and 10% wireless errors. We have 20 TCP sources. The rate of each of the 20 background cross traffic sources during the on period is set at 0.6782 Mbps to keep the congestion loss at the shared link for NewReno at $\sim 1\%$. For our loss labeling technique, the values for l and η are 8 and 6, respectively.[3] Table 1 summarizes the results for NewReno, NewReno-FF, TCP Westwood and NewReno-LL. Note that NewReno-PL, the perfect loss labeling version, is a special case of NewReno-LL with $P[C|C] = P[W|W] = 1$.

Configuration 2: Figure 4 shows the goodput and fairness index and Figure 5 shows the overhead of NewReno-LL against $P[W|W]$ and $P[C|C]$ under a different configuration. In this configuration, the number of TCP sources is reduced to 10 and the number of background traffic sources is increased to 30 so as to make the contribution of cross traffic significant. The rate of each of the background cross traffic sources during the on period is set at 0.557 Mbps to keep the congestion loss at the shared link for

[3]Recall that l and η are tunable parameters in our FF-based loss classification technique. We show later the effect of these parameters on performance—a short history is found to improve goodput. In general, l and η depend on the network configuration and may be adjusted dynamically. We leave this as future work.



(a) Goodput

(b) Fairness Index

Figure 2: Goodput and Fairness Index of NewReno-LL as a function of $P[W|W]$ and $P[C|C]$ (configuration 1)

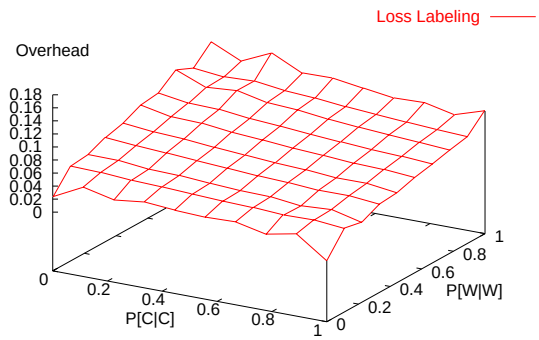


Figure 3: Overhead of NewReno-LL as a function of $P[W|W]$ and $P[C|C]$ (configuration 1)

NewReno at $\sim 1\%$. For our loss labeling technique, we set $l = 8$ and $\eta = 4$. Table 2 summarizes the results for NewReno, NewReno-FF, TCP Westwood and NewReno-LL.

Effect of History on $P[C|C]$ and $P[W|W]$: The length of history in our loss labeling technique affects the protocol behavior. Figure 6 shows the loss classification accuracy metrics and Figure 7 shows the goodput of NewReno-FF against l and η for configuration 1.

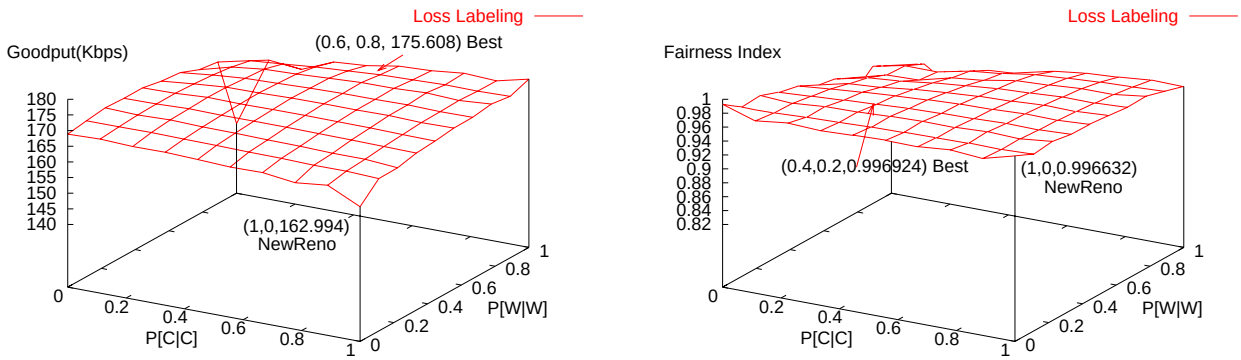
5.2 Observations and Discussion

- For both configurations, NewReno-FF achieves a goodput higher than that of NewReno and NewReno-LL for all values of $P[C|C]$ and $P[W|W]$. Recall that NewReno corresponds to $P[C|C] = 1$ and $P[W|W] = 0$. NewReno-FF achieves higher goodput without losing much in terms of fairness index. However, NewReno-FF has higher overhead than NewReno, which has the least overhead.

- The goodput spectrum in Figure 2 quantifies that NewReno-LL can achieve highest goodput at $P[C|C] = 0.4$ and $P[W|W] = 0.2$. The goodput of NewReno-FF is more than this highest goodput

Protocol	Goodput(bps)	Fairness	Overhead
NewReno	157297	0.995374	0.03347
NewReno-FF	163196	0.985023	0.0412812
Westwood	142516	0.710854	0.0835633
NewReno-LL Best($P[C\|C], P[W\|W]$)	158618 (0.4,0.2)	0.995374 (1,0)	0.03347 (1,0)
NewReno-LL Worst($P[C\|C], P[W\|W]$)	132314 (0,1)	0.803211 (0.2,1)	0.169904 (0.2,1)

Table 1: Comparison of performance metrics for different schemes (configuration 1)



(a) Goodput

(b) Fairness Index

Figure 4: Goodput and Fairness Index of NewReno-LL as a function of $P[W|W]$ and $P[C|C]$ (configuration 2)

value as shown in Table 1. The least goodput is observed at $P[C|C] = 0$ and $P[W|W] = 1$ as at this point, the protocol is most aggressive and the window is never adjusted on a congestion loss. We note that the fairness index is relatively comparable except for $P[C|C] \approx 0$ and $P[W|W] = 1$ where the excessive aggressive nature of the protocol overloads the network.

The average values of $P[C|C]$ and $P[W|W]$ for NewReno-FF are found to be 0.415401 and 0.602154, respectively. This means that $P[W|C] \approx 0.6$. This high value of congestion misclassification probability makes NewReno-FF more aggressive and consequently has the highest goodput. Note that the loss misclassification in NewReno-FF can sometimes be viewed as *finer* classification—for example, misclassifying *short-term* congestion as wireless and thus avoiding unnecessary window backoff may be beneficial.

Similar observations can be made for configuration 2 where the average values of $P[C|C]$ and $P[W|W]$ for NewReno-FF are found to be 0.222449 and 0.292744, respectively.

- NewReno with Perfect Labeling (NewReno-PL) is not necessarily optimal in terms of goodput. Although it seems reasonable to take a control action (such as ignoring window adjustment in wireless loss cases) based on the *exact* nature of a *past* loss (as in NewReno-PL), such control action may not always be correct or may not yield better performance globally. This is because of the delayed feedback. By the time the feedback reaches the senders, the *actual* network state might have changed.

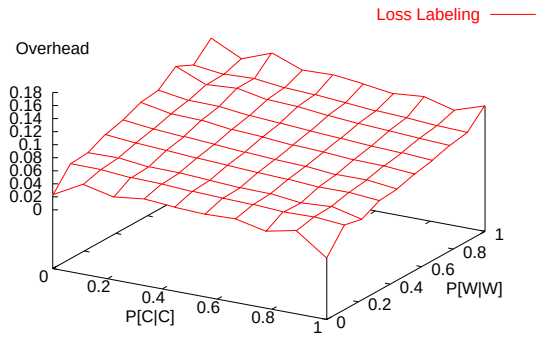


Figure 5: Overhead of NewReno-LL as a function of $P[W|W]$ and $P[C|C]$ (configuration 2)

Protocol	Goodput(bps)	Fairness	Overhead
NewReno	162994	0.996632	0.03426
NewReno-FF	182614	0.992191	0.0431594
Westwood	150901	0.72508	0.0838718
NewReno-LL Best($P[C\|C], P[W\|W]$)	175608 (0.6,0.8)	0.996924 (0.4,0.2)	0.03426 (1,0)
NewReno-LL Worst($P[C\|C], P[W\|W]$)	142222 (0,1)	0.827704 (0.2,0.1)	0.160594 (0.2,1)

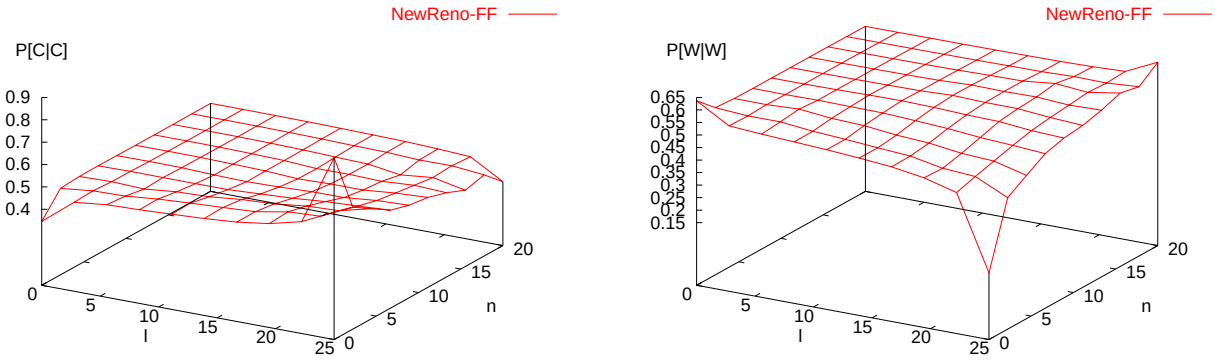
Table 2: Comparison of performance metrics for different schemes (configuration 2)

In NewReno-LL, on a loss event, the senders reduce their congestion window with probability $P[C|C]$ if the loss is a congestion one, and ignore window adjustments with probability $P[W|W]$ if it is wireless loss. Such randomization of the control actions may improve goodput by making the system more robust—for example, if the congestion loss was transient, then having some sources misclassifying it as wireless and thus not backing off their transmission rate may be beneficial.

- Although NewReno-FF adapts its $P[C|C]$ and $P[W|W]$ values according to the network conditions, the average values of $P[C|C]$ and $P[W|W]$ in both configurations are observed to be lower than one would expect.

  To justify the lower values of $P[C|C]$ and $P[W|W]$, we need to reconsider the definitions of the two metrics. The values of $P[C|C]$ and $P[W|W]$ much depend on the agility and stability of the filter. In our experiments, the Flip Flop filter is stable enough to filter out low frequency transient changes. The random wireless losses and even short-term congestion losses fall into this category. Therefore, the Flip Flop filter classifies many short-term congestion losses as wireless losses, thus the transmission window is not reduced. Although this "misclassification" lowers $P[C|C]$, this may in fact be a more appropriate control strategy. The loss classification process for a TCP flow is illustrated in Figure 8 for configuration 2. We don't have timeout cases over the shown time interval.[4] We have plotted the instantaneous queue length of the shared wired link, the instances of queue drops, wireless drops, and duplicate ACKs. The sample RTTs and upper control limit of estimated RTT

[4]Recall that in this paper, timeouts do not contribute to our loss classification process, only duplicate ACKs.



(a) $P[C|C]$ (b) $P[W|W]$

Figure 6: $P[W|W]$ and $P[C|C]$ of NewReno-FF as a function of l and η (configuration 1)

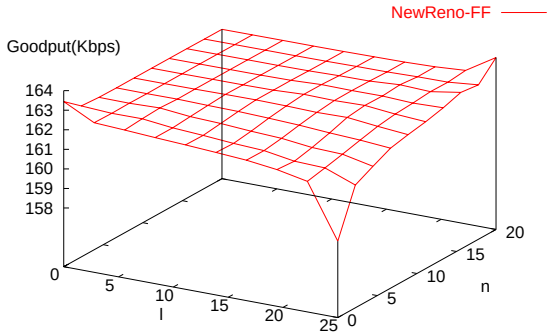


Figure 7: Goodput of NewReno-FF as a function of l and η (configuration 1)

are scaled by a factor of 900 on the y-axis.

When the queue drop $C1$ happens at 145.706 sec, the last l samples are below the threshold η of our loss labeling technique, therefore window adjustment is not done on the duplicate ACK, $D1$ at 146.001 sec. Observe that this action seems appropriate as the queue length drops indicating that congestion has subsided. In effect, the filter classifies this to be a case of "transient" congestion loss whereas NewReno would have simply reduced the *ssthresh* and congestion window. The error detection event based on duplicate ACKs, $D2$ at 146.43 is caused by a wireless drop, $W1$ but since the history of not enough outliers is there, this wireless loss will be correctly classified as a wireless loss and the congestion window is not reduced in this case.

As congestion builds up, many samples may be detected as outliers and if the congestion is persistent, the number of samples arriving at the sender will reduce. Therefore, outliers appearing in the growing phase of the bottleneck queue will retain the bad history for long if the values of l and η are chosen properly. On the other hand, if the congestion is transient, more samples will arrive subsequently and the bad history can soon be shaken off. The queue drop, $C2$ in Figure 8, is preceded by more than η outlier samples, therefore, the filter classifies the loss as congestion at $D3$, thus the transmission window is reduced. From the buffer occupancy, we can see that it is appropriate to be conservative

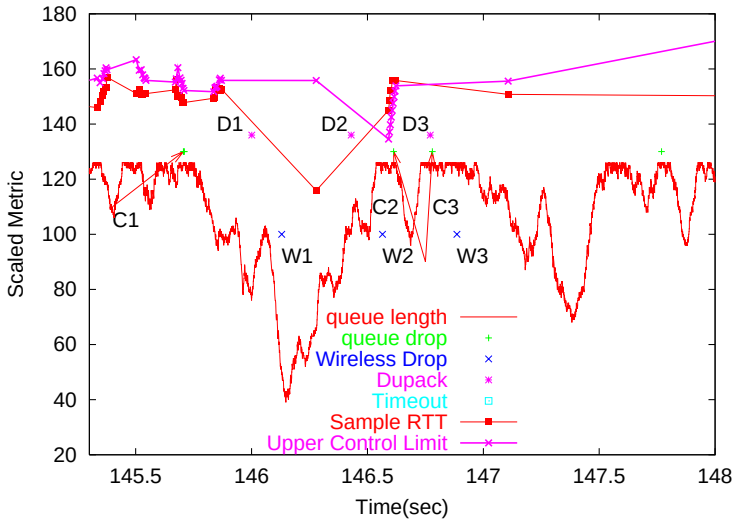


Figure 8: TCP dynamics with different event traces (configuration 2)

on $D3$.

- From Figures 6 and 7 we observe that the highest value of $P[C|C]$, 0.872928 is obtained at $l = 25, \eta = 0$ and highest $P[W|W]$, 0.636781 is observed at $l = 25, \eta = 20$. The goodput is also highest at $l = 25, \eta = 20$. This shows that increasing history using higher values of l and η generally increases goodput of NewReno-FF. Exactly how much history is dependent on the network configuration and is the subject of further research.

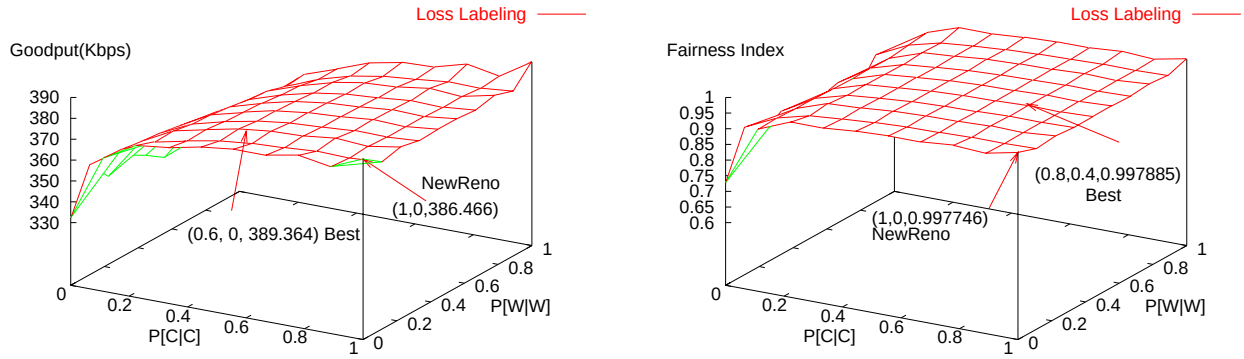
6 Burst Error Model

To study the performance of the protocols in the presence of correlated errors, we use the 2-state Markov model following [13] [15]. In such models burst errors occur at high rate due to a variety of conditions such as terminal mobility and fading. The wireless link is assumed to be in one of two states: Good or Bad. For the simulation experiments we keep the error rate of the Bad state at 5% (low-medium) while keeping 0% error rate in the Good state. In the error model, unless otherwise specified, the mean sojourn time of the bad state is 50ms and that of the good state is 500ms. For our loss classification technique, we keep $l = 20$ and $\eta = 5$ and we use configuration 2. For each cross connection, the on and off periods are Pareto distributed with average duration of 10ms and 30ms, respectively, and shape parameter of 2.5. The rate of each of the cross traffic sources during the on period is fixed at 0.85Mbps to produce a congestion loss $\sim 1\%$ for NewReno.

6.1 Simulation Results

Figure 9 shows the goodput and fairness index and Figure 10 shows the overhead of NewReno-LL against $P[W|W]$ and $P[C|C]$. The average $P[C|C]$ and $P[W|W]$ values for NewReno-FF are observed to be 0.320472 and 0.237183, respectively. Table 3 summarizes the results for NewReno, NewReno-FF, TCP Westwood and NewReno-LL for configuration 2.

Effect of History on $P[C|C]$ and $P[W|W]$: As we observed earlier, the length of history in our loss labeling technique affects the protocol behavior. Figure 12 shows the loss classification accuracy metrics



(a) Goodput

(b) Fairness Index

Figure 9: Goodput and Fairness Index of NewReno-LL as a function of $P[W|W]$ and $P[C|C]$ (configuration 2 with burst errors)

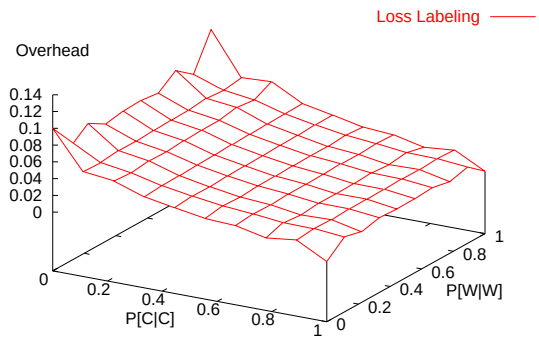


Figure 10: Overhead of NewReno-LL as a function of $P[W|W]$ and $P[C|C]$ (configuration 2 with burst errors)

and Figure 13 shows the goodput of NewReno-FF against l and η for configuration 2.

6.2 Observations and Discussion

- We observe that performance is mainly determined by $P[C|C]$; a high $P[C|C]$ value significantly improves performace. Thus, in the presence of bursty wireless errors, a more conservative transmission control strategy is desirable. Results are consistent with those obtained for random wireless losses.

- Figure 11 shows the behavior of NewReno-FF for one TCP flow in the presence of burst/correlated losses. As we emphasized earlier on the stability of the filter, the congestion losses C1 and C2 are very much like random drops. From the history of outliers, the Flip Flop filter (mis)classifies them as wireless losses at instances D2 (151.613 sec) and D3 (151.861 sec), respectively, when loss is detected through duplicate ACKs. Thus, the control action is that of maintaining the same transmission window. Observing the underlying buffer occupancy, we confirm that such action is indeed appropriate as the queue length drops indicating short-term congestion.

Protocol	Goodput(bps)	Fairness	Overhead
NewReno	386466	0.997746	0.01182
NewReno-FF	391110	0.99576	0.0269928
Westwood	351813	0.996096	0.0577565
NewReno-LL Best($P[C\|C], P[W\|W]$)	389364 (0.6,0)	0.997885 (0.8,0.4)	0.01182 (1,0)
NewReno-LL Worst($P[C\|C], P[W\|W]$)	169370 (0,1)	0.639093 (0,0.8)	0.123368 (0,0.8)

Table 3: Comparison of performance metrics for different schemes (configuration 2 with burst errors)

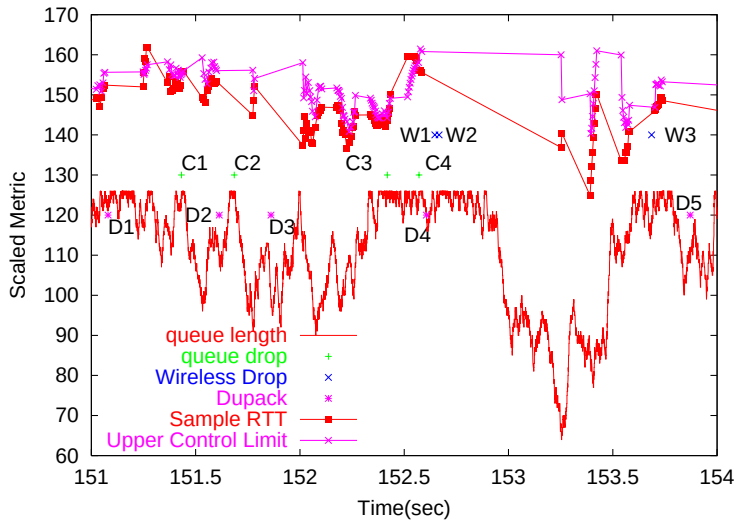
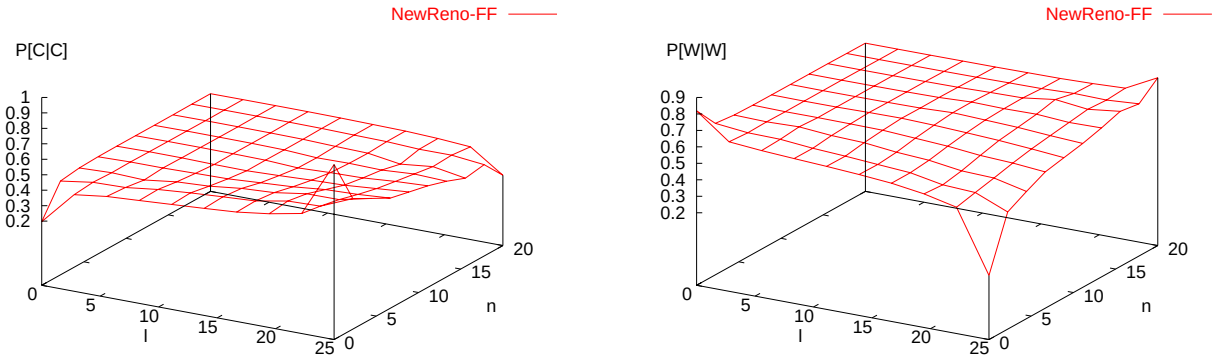


Figure 11: TCP dynamics with different event traces (configuration 2 with burst errors)

We next observe the congestion losses C3 and C4 at 152.419 sec and 152.571 sec, respectively. These losses are followed by consecutive wireless losses W1 and W2 at 152.647 sec and 152.667 sec, respectively. We also observe that the bottleneck buffer is full for an extended duration. Consequently, the wireless losses are overlapping with relatively persistent congestion losses. At D4 (at time 152.606 sec), as a result of congestion loss C4, the history of our loss classification technique has more than η outlier samples. Thus, the loss is classified (correctly) as congestion error and the sender window is appropriately reduced. Lastly, at D5 (at time 153.871 sec) which is due to wireless drop W3, NewReno-FF appropriately classifies it as wireless loss since the history does not contain any outliers.

We should point out that NewReno-FF may occasionally misclassify a packet loss as wireless loss when the network is congested. Such misclassification makes NewReno-FF unjustifiably aggressive as it keeps its transmission window unchanged. This aggressive behavior is evident from the increased overhead over regular NewReno.

- From Figures 12 and 13 we observe that the highest value of $P[C|C]$, 0.916505 is obtained at $l = 25, \eta = 0$ and highest $P[W|W]$, 0.787625 is observed at $l = 20, \eta = 15$. The goodput is highest at $l = 5, \eta = 4$ (392.561 Kbps). Thus, to achieve good performance in terms of goodput, a reasonable history is necessary.



(a) $P[C|C]$ (b) $P[W|W]$

Figure 12: $P[W|W]$ and $P[C|C]$ of NewReno-FF as a function of l and η (configuration 2 with burst errors)

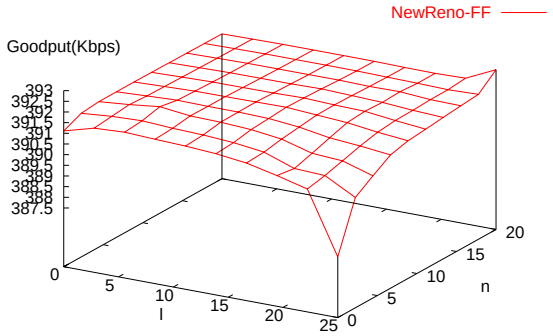


Figure 13: Goodput of NewReno-FF as a function of l and η (configuration 2 with burst errors)

7 Conclusions and Future Work

In this paper, we attempted to improve our understanding of the fundamental gains and limitations of loss classification techniques employed to empower TCP in hybrid wired/wireless networks. To this end, we abstract the general approach of various proposals by controlling the loss labeling accuracies, $P[C|C]$ and $P[W|W]$. These two measures quantify the probability that an event is identified correctly. In all proposals, this event is only one of two possibilites: congestion-induced loss or wireless loss. Such loss classification can be used to empower the recovery strategy of TCP. For example, the usual congestion control measure of backing off the transmission rate is taken if the packet loss is attributed to congestion. However, if the packet loss is attributed to wireless, a different control action is taken— in this paper, we assume an aggressive strategy whereby the transmission rate is kept unchanged.

We also introduced a new loss labeling technique that uses a Flip Flop filter to estimate RTT and use it to differentiate between congestion and wireless loss. Our technique uses *history* to examine the number of "outlier" RTT samples, i.e. those samples that exceed a control limit beyond which delay values are considered high. A packet loss is classified as congestion-induced if enough outliers are observed. Note that by using history, we overcome a limitation common to existing loss labeling techniques. This

limitation stems from the need to receive enough samples, which becomes difficult when the network is highly congested [5]. Maintaining a history of outliers enables our technique to overcome this challenge.

Through extensive simulations, we show that our loss labeling technique outperforms regular TCP and TCP Westwood in terms of goodput. Furthermore, its fairness is competitive to regular TCP and its overhead is lower than that of TCP Westwood. We also observe the low values of $P[C|C]$ and $P[W|W]$ under our Flip Flop based technique. This does *not* mean that our FF-based classification is inaccurate. Rather, the low value of $P[C|C]$ implies a higher value of $P[W|C]$.[5] This is the case when short-term congestion losses are treated as random wireless losses or more generally, *transient* losses. This is indeed an appropriate (in effect, finer grained) loss classification. Based on these observations, we believe that there is a need for such fine grained classification that goes beyond a binary classification of congestion versus wireless. Furthermore, this opens up the important research question of what kind of recovery actions should a protocol like TCP implement given the correct classification of packet losses.

We are currently analyzing our FF-based loss classification technique mathematically. We are also generalizing this analysis to any technique as a function of $P[C|C]$ and $P[W|W]$, or finer classification accuracies. We again note that the values of $P[C|C]$ and $P[W|W]$ depend on the parameters of the network as well as the loss classification algorithm. The ultimate goal is to develop a fine grained loss classification technique with an associated *adaptive* recovery strategy which enhances goodput and fairness while maintaining low overhead over hybrid wireless/wired networks. An ideal recovery strategy would match the level and density of error to appropriate transmission window adjustments. Lowering overhead is especially important for battery-operated devices.

References

[1] Anna Calveras Auge, Jaume Linares Magnet, and Josep Paradells Aspas. "Window Prediction Mechanism for Improving TCP in Wireless Asymmetric Links". In *Proceedings of Globecom'98*, Sydney, Australia, November 1998.

[2] A. Bakre and B. Badrinath. I-TCP: Indirect TCP for Mobile Hosts. In *IEEE ICDCS 1995: Proceedings of the 15th International Conference on Distributed Computing Systems*, pages 136–143, Vancouver, British Columbia, Canada, May 1995.

[3] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R.H. Katz. "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links". In *ACM SIGCOMM Conference*, Stanford University, CA, August 1996.

[4] H. Balakrishnan, S. Seshan, and R. Katz. "Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks". *IEEE/ACM Wireless Networks*, 1(4), December 1995.

[5] S. Biaz and N.H. Vaidya. Distinguishing Congestion Losses from Wireless Transmission Losses: A Negative Result. In *IC3N: Seventh International Conference on Computer Communications and Networks*, New Orleans, October 1998.

[6] Saad Biaz and Nitin H. Vaidya. Distinguishing Congestion Losses from Wireless Losses using Inter-Arrival Times at the Receiver. In *ASSET 1999: IEEE Symposium on Application-Specific Systems and Software Engineering & Technology*, Richardson, TX, March 1999.

[7] Song Cen, Pamela C. Cosman, and Geoffrey M. Voelker. "End-to-End Differentiation of Congestion and Wireless Losses". In *MMCN2002: SPIE Multimedia Computing and Networking*, San Jose, CA, January 2002.

[8] Tae eun Kim, Songwu Lu, and Vaduvur Bharghavan. "Improving Congestion Control Performance Through Loss Differentiation". In *ICCCN 1999: Eighth International Conference on Computer Communications and Networks*, Boston, MA, October 1999.

[5] Note that $P[C|C] + P[W|C] = 1$.

[9] L. A. Grieco and S. Mascolo. "TCP Westwood and Easy RED to Improve Fairness in High-Speed Networks". In *Seventh International Workshop on Protocols For High-Speed Networks (PfHSN'2002)*, Berlin, Germany, April 2002.

[10] V. Jacobson. Congestion Avoidance and Control. In *ACM SIGCOMM Conference*, Stanford, CA, August 1988.

[11] Raj Jain, D-M. Chiu, and W. Hawe. A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems. Technical report, DEC Research Report TR-301, September 1984.

[12] Minkyong Kim and Brian Noble. "Mobile Network Estimation". In *Mobicom 2001: The Seventh Annual International Conference on Mobile Computing and Networking*, Rome, Italy, July 2001.

[13] S. Q. Li and C. Hwang. "Link Capacity Allocation and Network Control by Filtered Input Rate in High Speed Networks". *IEEE/ACM Transactions on Networking*, 3(1):10–15, February 1995.

[14] Douglas C. Montgomery. *"Introduction to Statistical Quality Control"*. John Wiley & Sons, New York, 1985. (1st edition, 1985, 2nd edition, 1991).

[15] Saverio Morris and Claudia Casetti. "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links". In *MobiCom 2001: The Seventh Annual International Conference on Mobile Computing and Networking*, Rome, Italy, July 2001.

[16] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. In *ACM SIGCOMM Conference*, Vancouver, British Columbia, September 1998.

[17] J. Padhye and S. Floyd. "On Inferring TCP Behavior". In *ACM SIGCOMM Conference*, San Diego, CA, August 2001.

[18] C. Parsa and J.J. Garcia-Luna-Aceves. "Differentiating Congestion vs. Random Loss: A Method for Improving TCP Performance over Wireless Links". In *WCNC 2000: IEEE Wireless Communications and Networking Conference*, pages 23–28, Chicago, IL, September 2000.

[19] Karunaharan Ratnam and Ibrahim Matta. "WTCP: An Efficient Mechanism for Improving TCP Performance over Wireless Links". In *ISCC 1998: Proceedings of the Third IEEE Symposium on Computers and Communications*, Athens, Greece, July 1998.

[20] N.K.G. Samaraweera. "Non-congestion Packet Loss Detection for TCP Error Recovery using Wireless Links". In *IEE Proceedings Communications*, volume 146 (4), pages 222–230, August 1999.

[21] The Network Simulator – ns-2. http://www.isi.edu/nsnam/ns.

[22] Yoshito Tobe, Hiroto Aida, Yosuke Tamura, and Hideyuki Tokuda. "Detection of Congestion Signals from Relative One-Way Delay". *IPSJ Journal*, 42(12), 2001.

[23] Vassilis Tsaoussidis and Ibrahim Matta. Open Issues on TCP for Mobile Computing. *Journal of Wireless Communications and Mobile Computing – Special Issue on Reliable Transport Protocols for Mobile Computing*, 2(1), February 2002.

[24] R. Wang, M. Valla, M.Y. Sanadidi, B. K. F. Ng, and M. Gerla. "Efficiency/Friendliness Tradeoffs in TCP Westwood". In *ISCC 2002: Seventh IEEE Symposium on Computers and Communications*, Taormina, Italy, July 2002.

[25] C. Zhang and V. Tsaoussidis. TCP-Real: Improving Real-time Capabilities of TCP over Heterogeneous Networks. In *NOSSDAV 2001: The 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, Port Jefferson, NY, June 2001.