“De-randomizing” Congestion Losses To Improve TCP Performance over Wired-Wireless Networks

Saâd Biaz and Nitin H. Vaidya

Abstract

Currently, a TCP sender considers all losses as congestion signals and reacts to them by throttling its sending rate. With Internet becoming more heterogeneous with more and more wireless error-prone links, a TCP connection may unduly throttle its sending rate and experience poor performance over paths experiencing random losses unrelated to congestion. The problem of distinguishing congestion losses from random losses is particularly hard when congestion is light: congestion losses themselves appear to be random. The key idea is to “de-randomize” congestion losses. This paper proposes a simple biased queue management scheme that “de-randomizes” congestion losses and enables a TCP receiver to diagnose accurately the cause of a loss and inform the TCP sender to react appropriately. Bounds on the accuracy of distinguishing wireless losses and congestion losses are analytically established and validated through simulations. Congestion losses are identified with an accuracy higher than 95% while wireless losses are identified with an accuracy higher than 75%. A closed form is derived for the achievable improvement by TCP endowed with a discriminator with a given accuracy. Simulations confirm this closed form. TCP-Casablanca, a TCP-Newreno endowed with the proposed discriminator at the receiver, yields through simulations an improvement of more than 100% on paths with low levels of congestion and about 1% random wireless packet loss rates. TCP-Ifrane, a sender-based TCP-Casablanca yields encouraging performance improvement.

1 Introduction

Widespread use of wireless networks is becoming a reality. Companies and homes increasingly access the Internet through wireless links. Such a trend urges the community to adapt and enhance TCP for environments prone to random losses, unrelated to congestion: wireless networks, hybrid coaxial fiber broadband access networks [11], or very high speed links. Despite efficient link layer techniques (using automatic repeat request (ARQ) and/or forward error correction (FEC)), some losses unrelated to congestion may still occur. The transport layer must deal with these “residual” errors not recovered at the link layer.

TCP is the most popular transport protocol used over the Internet: it carries more than 90% [12] of all Internet traffic. The problem of TCP over links prone to random losses is well known [2, 4]: random transmission losses unnecessarily trigger TCP’s congestion control mechanisms. When a random loss occurs, TCP reacts as if the loss was due to congestion: TCP halves its congestion window, throttling unduly its sending rate, leading to poor performance over Internet paths that include non congested links. To avoid such a performance degradation, TCP could be enabled to distinguish congestion losses from random losses and react appropriately. A discriminator is any technique which distinguishes congestion losses from random losses unrelated to congestion. To this date, many end-to-end discriminators [5, 23, 15, 22, 9, 17] were proposed, but none is satisfactory enough to be widely deployed.

The design of an efficient discriminator is hard because congestion losses themselves often appear to be random. Researchers, unsuccessfully so far, have attempted to characterize a fingerprint of congestion losses that could help in distinguishing them from random losses. The problem of distinguishing congestion losses from random losses unrelated to congestion is particularly hard when congestion is light: congestion losses appear to be random because such losses are not always preceded by events such as an increase in the round trip time or variations in the packet interarrival times. To solve this problem, a promising approach is to “de-randomize” congestion losses such that the distribution of congestion losses differs from that of wireless error losses. With such an approach, when losses occur, one could ask the question: “Is this pattern of loss random?”.

If the answer is negative then losses are diagnosed as congestion losses. A discriminator, dubbed Casablanca, is built upon two components:

1. Send segments of a TCP connection using packets with different discard priorities
2. Establish on the routers a biased queue management that “de-randomizes” congestion losses.

Since congestion impacts differently packets with different dropping priorities, the probability distribution of congestion losses will be different for the two types of packets. Different patterns of losses may well exhibit a fingerprint of congestion dropping because random losses do not treat differently packets based on their discard priorities. The key idea of Casablanca is to send, within the same TCP connection, packets with different dropping preferences: a small proportion of selected packets are marked to be
Currently, a TCP sender considers all losses as congestion signals and reacts to them by throttling its sending rate. With Internet becoming more heterogeneous with more and more wireless error-prone links, a TCP connection may unduly throttle its sending rate and experience poor performance over paths experiencing random losses unrelated to congestion. The problem of distinguishing congestion losses from random losses is particularly hard when congestion is light: congestion losses themselves appear to be random. The key idea is to "de-randomize" congestion losses. This paper proposes a simple biased queue management scheme that "de-randomizes" congestion losses and enables a TCP receiver to diagnose accurately the cause of a loss and inform the TCP sender to react appropriately. Bounds on the accuracy of distinguishing wireless losses and congestion losses are analytically established and validated through simulations. Congestion losses are identified with an accuracy higher than 95% while wireless losses are identified with an accuracy higher than 75%. A closed form is derived for the achievable improvement by TCP endowed with a discriminator with a given accuracy. Simulations confirm this closed form. TCP-Casablanca, a TCP-Newreno endowed with the proposed discriminator at the receiver, yields through simulations an improvement of more than 100% on paths with low levels of congestion and about 1% random wireless packet loss rates. TCP-Ifrane, a sender-based TCP-Casablanca yields encouraging performance improvement.
dropped first by the routers when congestion occurs. A service similar to RIO (RED In/Out) [10] is expected from the network provider. In RIO packets are marked in or out by an edge router such that core routers preferentially drop packets marked out in case of congestion. However, we emphasize here three differences between Casablanca and RIO: in Casablanca scheme, (1) packets are marked in or out by the TCP sender (endpoint), (2) packets of the same TCP connection get different discard priorities, and (3) packets marked out are dropped first (with probability 1) before any packet marked in is. With RIO, (1) packets are tagged by edge routers, (2) packets of the same TCP connection are marked the same, and (3) a packet marked in may well get dropped while a packet marked out is not (it is only on average that packets marked out are dropped more often). The difference in treatment of packets marked in or out is limited to buffer management only. The discard priority based pattern of losses allows the development of a robust and accurate method to distinguish congestion losses from random losses.

This paper is organized as follows: Related work is presented in Section 2. Section 3 discusses the rationale of Casablanca discriminator. A closed form expression is proposed in Section 4 for the improvement one may expect from TCP if endowed with some discriminator that achieves a given accuracy in distinguishing congestion losses from random losses. The appropriate action that TCP should take when random losses unrelated to congestion are detected is discussed in Section 5. The techniques developed in this work are evaluated through simulations. Section 6 describes the experiments. The results of these experiments are presented and discussed in Section 7. Section 8 presents TCP-Ifrane, a sender-based TCP-Casablanca that does not require any change at the receiver. Section 9 briefly addresses issues related to Casablanca implementation. Section 10 is dedicated to conclusions and future work.

2 Related work

This work is interested only in end-to-end solutions to enhance TCP over networks where random losses may be unrelated to congestion. It is assumed that no intermediate node returns any explicit information [16] regarding congestion or random losses. There are mainly two approaches toward end-to-end solutions for enhancing TCP dealing with losses unrelated to congestion. The first approach consists in distinguishing random losses from congestion losses and make TCP react appropriately to each kind of loss. The second approach does not use losses as a congestion measure to control the sending rate. Instead, it attempts to estimate the available bottleneck link capacity and adopts an adequate sending rate that optimally exploits this available capacity. This approach does not require the discrimination of losses because losses are not used to control the sending rate.

For the first approach, Biaz and Vaidya [5] studied the ability of three loss discriminators based on congestion predictors: Jain’s delay based congestion predictor [14], Wang’s throughput based predictor [25], and Vegas predictor [8]. These congestion predictors rely on round trip time and/or throughput changes in response to congestion window size changes to infer the level of congestion on a path. These discriminators were shown to perform as poorly as a random coin tossing predictor. Samaraweera proposed a non-congestion packet loss detection (NCPLD) technique (based on Jain’s predictor). Kim et al. proposed the technique LIMD/H [15] that exploits the history of packet loss and the evolution of transmission rates (similar to Wang’s predictor) to infer the cause of loss. Parsa and Garcia-Luna-Aceves [22] proposed TCP Santa Cruz that updates a 3-state machine based on round trip time changes in response to congestion window size changes (very similar to the Vegas predictor).

The accuracy of classifying losses is not evaluated for NCPLD, LIMD/H, or TCP Santa Cruz. The authors only report TCP performance improvement when they enable TCP to distinguish congestion losses from random wireless losses. Note that some performance improvement is usually obtained even with weak loss discriminators. This observation is supported by Barman and Matta [3] who studied the effectiveness of poor loss discriminators in improving TCP performance. Barman and Matta showed that despite a low accuracy in diagnosing congestion losses or wireless losses, TCP performance can still be significantly improved.

Cen et al. proposed a hybrid loss discriminator ZBS [9] which uses three loss discriminators: ZigZag [9], Biaz [6], and Spike [24]. ZBS is used at the receiver. ZigZag and Biaz are based on the interarrival of the packets at the receiver and the number of losses detected. Spike is based on the relative one way trip time. ZBS dynamically switches between the three loss discriminators according to observed network conditions. The accuracy of ZBS discriminator depends on the number of flows sharing the bottleneck and yields high wireless and congestion misclassifications.

Liu et al. proposed an approach [17] that integrates two techniques: loss pairs (LP) and Hidden Markov Modeling (HMM). LP is based on sending two back to back packets. When one packet is lost, the round trip time for the second packet of the pair appears to bear some information about the cause of the loss. An HMM is trained over the observed RTTs to infer the cause of loss. All techniques above do not achieve high accuracy in classifying both congestion and wireless losses. The authors are not aware of any technique that achieves high accuracy for both types of losses.

In the second approach, losses are not used to update the congestion window. Therefore, there is no need to distinguish the losses. TCP Vegas illustrates such an approach. Biaz and Vaidya [5] and Liu et al. [17] showed that TCP-Vegas is a poor congestion predictor. This is surprising given the well known TCP-Vegas improvement. But, TCP-Vegas [8] introduces multiple modifications to TCP. Hengartner et al. [13] studied the individual impact of every modification proposed in TCP-Vegas. They showed that the Vegas congestion control scheme based on the round
trip time has a marginal (sometimes negative) impact on the performance gain of TCP Vegas. Mascolo et al. proposed TCP-Westwood [18] that estimates the available link capacity to adjust the size of the slow start threshold and the congestion window. Remarkable improvement is reported. However, as shown in this work, TCP-Westwood fails to accurately estimate the bottleneck link capacity in presence of non TCP traffic, particularly on the reverse path.

In this paper, a simple biased queue management scheme is proposed to be added to any active queue management (AQM) technique to distinguish congestion losses from random wireless losses. When TCP traffic is predominant (higher than 85%), simulation results show that Casablanca discriminator consistently achieves high accuracy (more than 95%) in identifying congestion losses and high accuracy in identifying wireless losses (more than 75%).

3 Rationale of the Casablanca Discriminator

The problem of distinguishing congestion losses from random wireless losses is particularly hard when congestion is light: congestion losses themselves appear to be random. A biased queue management that first drops specifically marked packets will “de-randomize” congestion losses such that when losses occur, one could ask: “Is this pattern of loss random?” If the answer is affirmative then losses are diagnosed as wireless, otherwise they are diagnosed as congestion. The Casablanca discriminator is implemented at the TCP receiver because the receiver has a better “view” of the losses than the sender. Consider Figure 1 that illustrates a pattern of losses that would likely result from a biased dropping as opposed from a wireless random dropping. Figure 1 presents a sender and a receiver with a path that includes a bottleneck link implementing a biased queue management and a wireless link. The sender sends 9 packets (small rectangles) numbered from 1 to 9. The sender marks packets numbered 2 and 7 as out (black rectangles) and marks all other packets as in (white rectangles). In case of congestion, the router at the bottleneck link must first drop packets marked out before dropping any packet marked in. Therefore, the black packets numbered 2 and 7 marked out are dropped first by the biased queue management at the bottleneck. If there is no congestion, the pattern of dropping will likely be different because, as shown in Figure 2, the wireless link does not distinguish between out and in packets. Since there is a high proportion of packets marked in, it is expected that these will likely be dropped. As illustrated in Figure 2, packets numbered 4 and 8 may get dropped. The probability that both packets marked out (packets 2 and 7) be randomly dropped on the wireless link among 9 packets is equal to \( \frac{1}{28} \) (hypergeometric distribution): it is unlikely that both packets marked out (2 and 7) be randomly dropped by a wireless link. From this observation, the receiver will diagnose a pattern of losses as biased if a high proportion of packets marked out are lost. If the pattern appears to be biased, the receiver concludes that the losses are due to congestion. In the following, it is formally shown that a biased queue management enables the design of a very accurate Casablanca loss discriminator.

Let us denote all data packets a TCP sender sends as \( P^M \): \( P^M_i \) is the \( i \)-th packet, and it is marked with \( M \) (in or out). A retransmitted packet keeps the same index \( i \) but is always marked in to avoid double jeopardy for packets initially marked out. It is assumed that routers first drop packets marked out, and start dropping packets marked in only if there are no more packets marked out in the queue. TCP sender marks the packets such that one packet out of \( k \) (\( k \geq 2 \)) packets is marked out (all other packets are marked in). The marking pattern is:

\[
P^{in}_1, P^{in}_2, \ldots, P^{in}_{k-1}, P^{out}_k, P^{in}_{k+1}, \ldots, P^{in}_{2k-1}, P^{out}_{2k}, \ldots
\]

Figures 1 and 2 illustrate two cases of congestion and wireless losses. Whenever an out-of-order packet is received, a loss is assumed to have occurred. TCP receiver considers the pattern of losses between the next expected packet \( P_{nxt} \) and the packet \( P_{hi} \) with the highest sequence number seen so far. For simplicity, the convention of \( ns=2 \) is adopted where packets are of fixed size and numbered (real TCP implementations number each byte). Let \( S = hi - nxt + 1 \) be the number of packets in the ordered set \( \{ P_{nxt}, \ldots, P_{hi} \} \). Suppose there are \( r \) losses among the \( S \) packets. The phenomenon of packet loss may be considered as a random sampling of \( r \) elements among a population of \( S \) elements which contains \( n \) defective elements (here, a defective element means a packet marked out). The probability of drawing (loosing) \( x \) packets marked out in a random sample of size \( r \) among a population of \( S \) packets follows a hypergeometric distribution. Let \( X \) be a random variable representing the number of packets marked out in a sample of \( r \) packets taken among \( S \) packets (\( r \) packets among \( S \) are lost). Denote \( y = \left\lfloor \frac{S}{X} \right\rfloor \) the number of packets marked out within the \( S \) packets, then

\[
P(X = x) = \frac{\binom{y}{x} \binom{S-y}{r-x}}{\binom{S}{r}}
\]

If the pattern of the \( r \) losses appears to be the result of a random sampling, then losses are wireless. Randomness
of sampling is tested as follows: if a loss pattern has a very low probability to occur, then the sampling is not random, but rather biased. In case of doubt, it is safe to conclude that losses are due to congestion.

As an example, suppose that one packet out of 8 \((k = 8)\) is labeled \textit{out} and that \(S = 24\). Consider two extreme cases. The first case is when 3 packets marked \textit{out} are the only missing packets within \(S = 24\) packets. Since \(P(X = 3) = \frac{1}{120}\), it is quite unlikely that the sampling was random: the sampling is rather biased. Since the sampling is probably not random, then the three losses are probably not wireless. Therefore, such a pattern with a loss of three packets marked \textit{out} is a very strong indication of congestion losses. The second extreme case is when 3 packets marked \textit{in} are lost. The probability that these losses are wireless is equal to \(P(X = 0) \approx 0.65\). This means that three packets marked \textit{in} may very well have been corrupted. But, what makes this pattern a strong indication of wireless losses is the following: in case of congestion, packets marked \textit{out} are dropped first. It is very unlikely that NO packet marked \textit{out} is dropped during congestion while three packets marked \textit{in} are. From these observations, a very simple function is developed to “summarize” the pattern of the loss.

\[
F(x, r, k) = 1 - \left[ k \cdot \frac{x}{r} \right]
\]  

(1)

where \(x\) is the number of lost packets marked \textit{out} and \(r\) is the number of losses within the ordered set \( \{P_{nat}, \ldots, P_{hi}\} \). The function \(F(x, r, k)\) is built upon the following rationale: if losses are wireless (random), it is expected to have \(\frac{x}{r} \approx 1\) if the losses are uniformly distributed, and \(F(x, r, k)\) will be equal to 0. Recall that \(\frac{x}{r}\) represents the proportion of lost packets marked \textit{out} (i.e., \(x\)) over the total number of losses (i.e., \(r\)). If the losses are due to congestion, it is expected that the proportion \(\frac{x}{r}\) of packets marked \textit{out} be higher than \(\frac{1}{r}\), making \(F(x, r, k)\) negative. Smaller is \(F(x, r, k)\), higher is the likelihood of congestion losses. Note that \(1 - k \leq F(x, r, k) \leq 1\). The function \(F(x, r, k)\) does not capture all the information that could be extracted from the pattern of losses. However, the function \(F(x, r, k)\) yields a simple, robust, and accurate discriminator. This discriminator is called \textit{Casablanca}. Whenever \(r\) losses occur with \(x\) packets marked \textit{out}, they are diagnosed as follows: if \(F(x, r, k) \geq 0\), then the losses are diagnosed as wireless losses, otherwise (i.e., \(F(x, r, k) < 0\)) they are diagnosed as congestion losses.

Figure 3 presents how \textit{Casablanca} is implemented within TCP-Newreno at the server (sender) and client (receiver) sides. TCP-Newreno is the basis to implement Casablanca. TCP-Newreno recovers from multiple losses within the same window by halving the congestion window in response to only ONE of the losses. The diagnosis of a loss is made by the TCP receiver when it generates a duplicate ack. As the TCP receiver receives further out of order packets, it keeps diagnosing losses and sending back its diagnosis with generated dupacks. If a wireless loss is diagnosed, the duplicate ack is marked with "ELN" (explicit loss notification). The sender uses the marking on the third dupack to decide whether to halve or not the congestion window.

To measure the accuracy of \textit{Casablanca} discriminator, two metrics are used: the congestion accuracy \(A_c\) and the wireless accuracy \(A_w\). \(A_c\) is the ratio of the number of congestion losses correctly diagnosed over the total number of congestion losses. \(A_w\) is similarly defined for wireless losses. Our objective is to achieve a congestion accuracy \(A_c \approx 1\) because multiple congestion losses mistaken for wireless losses may significantly degrade the network performance, since TCP senders would not decrease their sending rate when they should. On the other hand, a lower accuracy \(A_w\) for diagnosing wireless losses is acceptable: a wireless loss mistaken for congestion will only degrade the performance of this TCP connection that unduly throttles its sending rate. As of today, TCP has an accuracy \(A_c\) of 1 (all losses are considered congestion losses) and an accuracy \(A_w\) of 0. Our objective is to reach \(A_c \geq 0.95\) and \(A_w \geq 0.75\). The value of \(k\) is critical to the accuracy of \textit{Casablanca} discriminator. Relationships between \(k\), the accuracy \(A_w\), the accuracy \(A_c\), and the congestion packet loss rate \(p_c\) are developed in the following. Upperbounds on \(A_w\), \(A_c\), and \(k\) are established below. \textit{Casablanca} scheme is shown to be fully efficient only if \(k \leq \frac{\sqrt{p_c}}{2}\). Moreover, the impact of the bottleneck buffer size on \(A_c\) is discussed.

**Upperbound on \(A_w\):** Consider the set \(W\) of wireless losses: the expected fraction of packets marked \textit{out} in \(W\) is \(\frac{1}{2}\) because: (1) wireless losses are assumed random, (2) the fraction of packets marked \textit{out} is \(\frac{1}{r}\), and (3) wireless loss phenomenon does not discriminate between packets marked \textit{in} and \textit{out}. If losses do not occur in batches then each wireless loss of a packet marked \textit{out} will be incorrectly diagnosed as a congestion loss. In summary, of all wireless losses, a fraction \(\frac{x}{r}\) will be incorrectly diagnosed. So, the

![Figure 3: TCP-Casablanca Flow Chart](image-url)
accuracy $A_w$ cannot be higher than $1 - \frac{1}{k}$. Then,

$$A_w \leq 1 - \frac{1}{k} \quad (2)$$

This bound assumes that a proportion of $\frac{1}{k}$ packets marked \textit{out} reach the wireless link: this assumption is only true when there is no congestion, or congestion is very low. Such a bound suggests to take a large value for $k$. However, $k$ cannot be large without limit.

\textbf{Upperbound on $A_c$:} Consider a flow with a total number of 100 packets with one packet out of 10 marked \textit{out} (i.e., $k=10$). A favorable case is when the congestion loss rate is less than 10\% (i.e., $p_c \leq \frac{1}{k}$). In such a favorable case, less than 10 packets on average will be dropped. Since there are 10 packets marked \textit{out}, it is highly probable that BQM will "find" in the queue packets marked \textit{out} to drop and no packet marked \textit{in} would be dropped. Therefore, congestion losses will be correctly diagnosed. Under such favorable conditions, $A_c$ can reach 1.

Now, suppose that the congestion loss rate is higher than 10\% (i.e., $p_c \geq \frac{1}{k}$). Let $p_c$ be for example 15\%. Then 15 packets will be dropped. Since there are only 10 packet marked \textit{out}, then 5 packets marked \textit{in} will be dropped and will be incorrectly diagnosed as wireless losses.

In summary, the congestion loss rate $p_c$ must be less than $\frac{1}{k}$ for $A_c$ to reach 1. Otherwise, (i.e., if $p_c \geq \frac{1}{k}$), then $A_c$ is upperbounded by $1 - (p_c - \frac{1}{k})$. Note that $(p_c - \frac{1}{k})$ is the fraction of packets marked \textit{in} that would be dropped on average by BQM and incorrectly diagnosed as wireless losses.

$$A_c \leq 1 - (p_c - \frac{1}{k}) \quad \text{when} \quad p_c \geq \frac{1}{k}$$

\textbf{Upperbound on $k$:} Let $p_c$ be the congestion packet loss probability experienced by a TCP connection at the bottleneck. If $p_c > \frac{1}{k}$, then there are more packets dropped due to congestion than the number of arrivals (at the bottleneck) of \textit{out} packets. Therefore, the router is forced to drop packets marked \textit{in}. When a router starts dropping packets marked \textit{in}, the dropping is not anymore biased, but rather random. Then, given that a loss is due to congestion, the probability that it was a packet marked \textit{in} is equal to $1 - \frac{1}{k}$. If a packet marked \textit{in} is lost due to congestion, $F(x, r, k) = F(0, 1, k) = 1$ wrongly implying that it is a wireless loss. Therefore,

$$A_c \leq \frac{1}{k} \quad \text{when} \quad p_c \geq \frac{1}{k}$$

In other words, if $k$ is too large, $A_c$ will be very small. $k$ should be chosen such that $k \leq \frac{1}{p_c}$. For a congestion dropping probability of 10\% ($p_c = 0.10$), $k$ must be smaller or equal to 10.

\textbf{Impact of the Bottleneck Buffer Size:} if the buffer size at the bottleneck is too small, the biased queue management will not work well because there will not be many packets marked \textit{out}. With a small buffer, packets marked \textit{out} quickly get exhausted, forcing the drop of packets marked \textit{in}. The impact of the buffer size on the accuracies $A_w$ and $A_c$ is investigated later.

\textbf{TCP-Casablanca over Networks Without Wireless Losses:} TCP-Casablanca is beneficial in environments with congestion losses exclusively. Measurements were made on wired networks. These measurements show that biased queue management (BQM) has a positive impact on TCP performance over wired networks WITHOUT any change to TCP code. BQM spreads out congestion losses in comparison to drop-tail that drops consecutive packets. With BQM, congestion losses are:

1. spread \textit{out} over time within each flow: BQM drops first packets marked \textit{out}. As long as there are packets marked \textit{out} in the router, drops will be spread out over time for the same flow.
2. spread \textit{out} among different flows. When looking for packets to drop, there is a high probability that packets marked \textit{out} will be dropped from different flows as long as there are different flows in the router.

The spreading out over time within the same flows increases the probability to recover without timeouts while the spreading among flows results in more fairness.

TCP-Casablanca is not an end-to-end scheme because it requires a specific behavior from the network layer. TCP-Casablanca will not work if routers implement any buffer queue management other than BQM. On the other end, TCP-Casablanca does not require any explicit feedback from any network element.

An important question is whether 	extit{Casablanca} discriminator can improve TCP throughput. Section 4 establishes an expression of the expected improvement of TCP throughput when TCP is endowed with a discriminator with accuracies $A_c$ and $A_w$.

\section{Expected Improvement}

Now, an approximate analysis of the expected improvement using the proposed scheme is proposed. This analysis is validated later through simulation results. Padhye et al. established \cite{20} a closed form expression for TCP throughput as a function of loss rate $p$, average round trip time \textit{RTT}, duration of the first timeout $T_0$, and number of packets acknowledged per Ack $b$. Assuming that there is no limit on the flow control window size (no buffer constraints at the end points), the expression of the throughput $Thgt(p)$ (in packets per second) is

$$Thgt(p) = \frac{1}{RTT \sqrt{\frac{2bp}{3}} + T_0 \min \left(1, 3 \sqrt{\frac{3bp}{8}}\right) p(1 + 32p^2)}$$

(3)

Consider a TCP connection that experiences congestion and wireless losses with loss rates $p_c$ and $p_w$, respectively. The throughput of such a TCP connection is $Thgt(p_c + p_w)$ because TCP does not distinguish the different types of losses. Suppose that some discriminator achieves the
accuracies $A_c$ and $A_w$. With such a discriminator, a TCP sender would not halve its congestion window size when it diagnoses wireless losses. Along the lines of Padhye et al.’s [20] work, the authors derived an approximation of the throughput that could be achieved if a discriminator with accuracies $A_c$ and $A_w$ is used with TCP. The expression of this throughput $Thgt(p_c, p_w, A_c, A_w)$ depends on $p_c$, $p_w$, $A_c$, and $A_w$:

$$\frac{1}{RTT} \sqrt{\frac{2b}{3}} + T_0 \min \left( 1, 3 \sqrt{\frac{3p_w}{8}} \right) p_{cw} (1 + 32p_{cw}^2)$$

where $p'_c = A_c \cdot p_c + (1 - A_w)p_w$ and $p_{cw} = p_c + p_w$.

When timeouts are not taken into account and TCP recovers from a packet loss only through fast-recovery and fast retransmit with halving the congestion window, Mathis et. al [19] and others have independently established that TCP throughput depends only on the term $A = RTT \sqrt{\frac{2b}{3}}$. When taking into account timeouts, Padhye et. al showed that for a given packet loss rate $p$, timeouts impact the throughput with the term $(B = T_0 \min \left( 1, 3 \sqrt{\frac{3p_w}{8}} \right) p(1 + 32(p^2))$).

Now, if we endow TCP with a loss discriminator, this discriminator is invoked only for losses recovered through fast retransmit and fast recovery. A discriminator is not invoked when a timeout occurs. For losses detected through timeouts, TCP backs off regardless of the reason of a loss. Therefore, the term $B$ resulting from a loss rate $p_{cw} = p_c + p_w$ is independent from the accuracies $A_c$ and $A_w$. One of the authors discussed this with Padhye who found the argument correct.

Now, a discriminator is invoked during fast retransmit and fast recovery. Therefore, the term $A$ is impacted by accuracies $A_c$ and $A_w$. TCP endowed with a discriminator backs off only for losses that it diagnoses as congestion losses: a discriminator diagnoses congestion losses with rate $A_c \cdot p_c$ and mistakes wireless losses for congestion losses with rate $(1 - A_w)p_w$. Therefore, TCP performance is similar to a TCP agent which experiences a congestion loss rate of $p'_c = A_c \cdot p_c + (1 - A_w)p_w$.

Based on our discussion above about the term $B$, the accuracies $A_c$ and $A_w$ do not play any role for the term related to timeouts (i.e., $T_0 \min \left( 1, 3 \sqrt{\frac{3p_w}{8}} \right) p_{cw}(1 + 32p_{cw}^2)$) because the nature of the loss has no impact on performance degradation due to timeouts. If a discriminator with accuracies $A_c$ and $A_w$ is used for a TCP connection that experiences congestion and wireless losses with packet loss rates $p_c$ and $p_w$ respectively, the percentage of improvement $Imp(p_c, p_w, A_c, A_w)$ in throughput is:

$$Imp(p_c, p_w, A_c, A_w) = 100 \times \frac{Thgt(p_c, p_w, A_c, A_w)}{Thgt(p_c + p_w)} - 1$$  \hspace{1cm} (4)

Note however that the expression for $Imp(p_c, p_w, A_c, A_w)$ does not capture the fact that, when using a loss discriminator, the congestion packet loss rate $p_c$ is higher than when using a normal TCP that considers all losses as congestion losses. The proposed expression $Imp(p_c, p_w, A_c, A_w)$ in Equation (4) assumes that the congestion packet loss rate $p_c$ observed with TCP (without discriminator) remains the same as with TCP using a discriminator. This is not accurate. Let $p_c$ be the congestion packet loss rate when using a normal TCP without discriminator. If TCP is endowed with a discriminator, ceteris paribus, then the connection experiences a congestion packet loss rate higher than $p_c$. This is due to the fact that TCP endowed with a discriminator backs off less often, leading to more aggressive TCP flows that provoke more congestion losses.

With a perfect discriminator ($A_c = 1$, $A_w = 1$), $p_c$ is higher than with an imperfect discriminator ($A_c = 1$, $A_w < 1$). Figures 4 and 5 plot the improvement $Imp(p_c, p_w, A_c, A_w)$ (based on Equation (4)) that could be expected from a discriminator with accuracies $A_c$ and $A_w$ when used with TCP. The horizontal axis represents, on a logarithmic scale, the wireless packet loss rate $p_w$ on the wireless medium. The vertical axis represents the improvement $Imp$. Figures 4 and 5 plot four curves corresponding to four packet congestion loss rates $p_c$: 0.0001, 0.001, 0.01, and 0.1. First, note that the curves related to $p_c = 0.1$ (10%) are barely visible because with high congestion packet loss rate, there is no improvement even if the discriminator is perfect. Figure 4(a) plots the results for a perfect discriminator with accuracies $A_c = 1$ and $A_w = 1$.

Second, observe that if the packet congestion loss rate $p_c$ is equal to 0.01 (1%), the expected improvement is still marginal (at most 20%). Third, when congestion is low, the improvement can be quite significant. When $p_w$ is very small, the improvement is small in comparison to the maximal improvement: the appropriate reaction of TCP to wireless losses has a marginal impact on TCP performance because wireless losses are so few. On the other hand, if wireless packet loss rate is high (0.1), the improvement is also small in comparison with the maximal improvement. With a high wireless packet loss rate, the pipe dries out too often even with a high congestion window size. Moreover, with a high packet loss rate, losses are typically detected through timeouts: when time out occurs, TCP sets the congestion window size to the initial value regardless of the reason of the loss. With a perfect discriminator (please see Figure 4(a)), TCP would yield impressive im-

![Figure 4: Imp(p_c, p_w, A_c, A_w) versus p_w](image)
Improvement of up to 250% when congestion packet loss rate $p_c = 0.0001 = 0.01\%$, and the wireless packet loss rate $p_w \approx 0.005 = 0.5\%$ conditions. Figures 4(b) and 5(b) illustrate that the improvement is quite sensitive to accuracy $A_w$. $A_w$ has a strong impact on the TCP performance improvement, especially for low congestion packet loss rate. A discriminator with accuracies $A_c = 1$ and $A_w = 0.80$ yields a much smaller improvement than a perfect discriminator. Figure 5(a) illustrates how much the improvement is sensitive to the accuracy $A_c$. $A_c$ has much less impact on the TCP performance improvement than $A_w$, especially for low congestion packet loss rate. Figure 5(a) shows that with a lower accuracy $A_c = 0.80$ (and $A_w = 1$), the expected improvement is slightly higher than with $(A_c = 1)$ because the TCP sender (mistakenly) back off less often. In reality, such a behavior of the TCP sender will lead to more congestion losses and will yield a poor performance improvement. The objective is to get a very high accuracy $A_c \approx 1$ and an accuracy $A_w$ as high as possible. Figure 6 presents $\text{Improvement} = 0.070s, T_0 = 1.25s, b = 2$ (a) $(A_c = 0.80, A_w = 1)$ (b) $(A_c = 0.80, A_w = 0.80)$

Figure 5: $\text{Imp}(p_c, p_w, A_c, A_w)$ versus $p_w$

$\text{RTT} = 0.005s, T_0 = 1.25s, b = 2$ (a) $(A_c = 1, A_w = 1)$ (b) $(A_c = 0.98, A_w = 0.80)$

Figure 6: $\text{Imp}(p_c, p_w, A_c, A_w)$ versus $\text{RTT}$

the expected improvement $\text{Imp}(A_c, A_w, p_c, p_w)$ (based on Equation (4)) versus the round trip time $\text{RTT}$ with a constant wireless packet loss rate of $p_w = 0.005$. Figure 6 plots four curves corresponding to four packet loss rates $p_c$ due to congestion: 0.0001, 0.001, 0.01, and 0.1. The curves related to $p_c = 0.1$ (10%) are not visible even for high $\text{RTT}$. Figure 6(a) plots the expected improvement $\text{Imp}(1.0, 1.0, p_c, 0.005)$ (based on Equation (4)) from TCP endowed with a perfect discriminator versus the round trip time $\text{RTT}$. Figure 6(a) shows that when congestion is light ($p_c = 0.0001$), the improvement exceeds 500% for $\text{RTTs}$ over 500ms if TCP is endowed with a perfect discriminator. The improvement increases with $\text{RTT}$. This is due to the fact that when a TCP sender halves the congestion window, it takes more time to recover the initial value if the $\text{RTT}$ is higher. Since TCP connections with longer $\text{RTTs}$ suffer more from congestion window size halving, it is normal that they benefit more from a TCP endowed with a perfect discriminator. Figure 6(b) plots the expected improvement $\text{Imp}(p_c, 0.005, 0.98, 0.80)$ (based on Equation (4)) from TCP endowed with an imperfect discriminator ($A_c = 0.98$ and $A_w = 0.80$) versus the round trip time $\text{RTT}$. The values ($A_c = 0.98$ and $A_w = 0.80$) are chosen because Casablanca discriminator achieves such accuracies. Figure 6(b) illustrates the tremendous impact of the accuracy $A_w$ as the expected improvement drops to 100% for a discriminator with accuracy $A_w = 0.80$. However, the expected improvement is significant and is encouraging to pursue the design of a discriminator.

Padhye et al’s Expression 3 of the throughput assumes that losses are independent. This assumption is valid on high bandwidth channels where a TCP connection represents a small fraction of the overall traffic. When computing the improvement in throughput, it is assumed that congestion losses are independent from each other and from the behavior of a TCP sender that responds appropriately to each type of losses. This last assumption is not accurate. Let $p_c$ be the congestion loss rate for TCP-Newreno. When TCP-Newreno responds appropriately to losses, its throughput increases and the congestion loss rate increases too to some value $p'_c$ with $p'_c \geq p_c$. By using the same value $p_c$ to compute the throughput $T_g$ of TCP-Casablanca, the throughput $T_g$ is overestimated. The increase of $p_c$ depends on many factors: bandwidth at the bottleneck, the round trip time, the number of TCP connections sharing the bottleneck link. Therefore a closed form for the new value $p'_c$ as a function of $p_c$ could not be established. Expression 4 for the improvement is an upperbound on the throughput improvement.

5 Appropriate Reaction to Wireless Losses

Past papers typically advocate that TCP should not halve the congestion window size or adjust the slow start threshold when a packet loss is known to be due to wireless errors. The authors advocate in the following that, even with a perfect loss discriminator, TCP should somewhat throttle its sending rate for wireless losses. TCP should decrease the congestion window size because a wireless loss signals to some extent a temporary decrease of the link layer goodput of the wireless link. Higher the wireless packet loss rate, lower will be the goodput of a wireless link. This decrease may lead to queue build up and congestion drops. Therefore, TCP should at least decrease by one packet its congestion window size for each wireless loss detected (additive decrease). Let us now consider a loss discriminator that is not perfect (i.e., $A_c < 1$ and $A_w < 1$). Let $n$ be
the number of packets sent by a TCP sender. For simplicity, retransmissions are neglected. The average number of congestion losses is \( n p_c \). The estimated number \( M_c \) of misclassified congestion losses (congestion losses diagnosed as wireless losses) is equal to \( n p_c(1 - A_c) \). Therefore, TCP wrongly does not halve its congestion window size for \( M_c \) losses. On the other hand, the estimated number \( M_w \) of misclassified wireless losses is equal to \( n p_w(1 - A_w) \). Here, TCP wrongly halves its congestion window size for \( M_w \) losses. Optimistically, the wireless and congestion loss misclassifications statistically will cancel each other. However, it is better to have \( M_w \geq M_c \), i.e., the TCP sender should wrongly throttle its sending rate more often than wrongly maintain its sending rate. This requires to know \( \frac{M_w}{M_c} \). Since 

\[
\frac{M_w}{M_c} = \frac{p_w}{p_c} \frac{1 - A_w}{1 - A_c}
\]

depends on \( p_w \) and \( p_c \) that are unknown, \( \frac{M_w}{M_c} \) must be estimated. Suppose that \( n_c \) losses are classified as congestion losses, then a good estimate \( \widehat{n_c} \) of the real number of congestion losses is \( \frac{A_c}{A_w} n_c \). Similarly the real number of wireless losses can be estimated as \( \widehat{n_w} = \frac{A_w}{A_c} n_c \). By simple algebraic manipulation, 

\[
\frac{M_w}{M_c} = \frac{A_w}{A_c} \frac{1 - A_w}{1 - A_c}
\]

now, suppose that \( \frac{M_w}{M_c} < 1 \) (there are more misclassified congestion losses than misclassified wireless losses). In such a case, TCP congestion control stability may be jeopardized. To avoid this, the only way is to decrease the congestion window also for losses diagnosed as wireless losses. Further study is needed to determine how to decrease the window size for losses diagnosed as wireless losses such that the effects of misclassifying congestion losses and wireless losses cancel each other statistically. In all simulation experiments of this work, a TCP sender does not back off when a wireless loss is detected.

6 Experiments

Accuracies \((A_c, A_w)\) of Casablanca discriminator and the improvement \( \text{Imp}(p_c, p_w, A_c, A_w) \) are measured using ns-2 simulations. The impact of the value \( k \), the round-trip time, and the buffer size on the accuracies \( A_c \) and \( A_w \) is investigated. This section presents the topology used for the simulations, the packet loss model, and the method used to collect the data. Figure 7 shows the topology used. There are three types of pairs sender-receiver: TCP connections over type A sender-receiver pair experience the longest propagation delay path with a wireless last hop. There are five routers \( R_i \). The dashed lines show the TCP transfers between senders and receivers. With this topology, a competing TCP traffic with different round trip times is maintained. Bit rates on all links are set such that the bottleneck is the link \( R_3-R_4 \). All senders are TCP senders. Sources are fed with FTP traffic. The congestion level is controlled by varying the bit rate \( B_w \) on the bottleneck \( R_3-R_4 \) and the number of senders \( N_1, N_2, \) and \( N_3 \). Every link is marked with a pair (bit rate, propagation time). While simulations are run with different bit rates, the results presented in this paper have a bit rate \( B_w \) on the wired bottleneck of 45 Mbps and a bit rate \( B_{ws} \) on the wireless link of 10 Mbps. Different experiments were run with \( N_1, N_2, \) and \( N_3 \) varying from 0 to 32. For some experiments, only senders of type A were used to get low congestion packet loss rates.

For the wireless packet loss model, a two-state Markov model is used. In each state, the time between successive losses is exponentially distributed with a mean that depends on the state: in the good state, the mean time is much higher than the mean time for the bad state. The transition probability from the good (resp. bad) state to the bad (resp. good) state is 0.10 (resp. 0.90).

Two discriminators were added to TCP sink in ns-2: Casablanca discriminator and a Perfect discriminator. The perfect discriminator is implemented in a brute force manner: whenever a wireless loss occurs, its sequence number and TCP flow information are logged. Whenever a receiver detects a loss, the perfect discriminator diagnoses it by checking whether this loss was logged. The Perfect discriminator achieves the accuracies \( A_c = 1 \) and \( A_w = 1 \) and establishes an upperbound on the accuracy. The perfect discriminator cannot be implemented on real networks: it assumes perfect knowledge of all wireless losses. From now on, TCP-Newreno endowed with the Casablanca discriminator will be referred as (TCP-Casablanca), and to TCP-Newreno with a perfect discriminator as (TCP-Perfect). For these experiments, TCP connections are started at random times and they last until the end of the experiment. Each experiment lasts 180 seconds. Starting times of the TCP connections are randomly scheduled within a period equal to the round trip propagation time. Accuracies \( A_c \) and \( A_w \), and the throughput \( \text{Thgt} \) are collected for each experiment. Note that the same starting times are used to conduct the experiment with TCP-Newreno, TCP-Casablanca, and TCP-Perfect. 30 runs of the same experiment are run with changing only the starting times. The results reported here for the accuracies and the improvement in throughput are the average over the 30 runs.

\footnote{With Casablanca scheme, the values \( A_c \) and \( A_w \) remain quite constant within 5\%.}

3The two-state packet loss model provided with ns-2 (version 2.1b9a) was not used. This model is known to be flawed: time stops increasing when a link is idle.
7 Results and Discussion

7.1 Accuracies $A_c$ and $A_w$

It is shown in Section 3 that the accuracies $A_c$ and $A_w$ depend on the value $k$ (one packet marked \textit{out} every $k$ packets, others being marked \textit{in}), the congestion packet loss rate $p_c$, and the buffer size at the bottleneck. In this section, the relationship between the accuracies $A_c$ and $A_w$ and the three parameters $k$, $p_c$, and bottleneck buffer size is verified through simulations.

Impact of $k$ on $A_c$ and $A_w$: Figure 8(a) plots the \textit{measured} accuracy $A_c$ versus $k$ for different values of the wireless packet loss rate $p_w$: $0.0001$, $0.001$, and $0.01$. Similarly, Figure 8(b) plots the \textit{measured} accuracy $A_w$ versus $k$ for the following values of the wireless packet loss rate $p_w$: $0.0001$, $0.001$, $0.01$, and $0.1$. Additionally, Figure 8(b) plots the upperbound on $A_w$ (i.e., $1 - \frac{1}{\pi}$) from Equation (2). The horizontal axis represents $k$ taking the values from $2$ to $50$. The key observation is that accuracy $A_c$ decreases sharply at $k = 8$. It was shown that a large $k$ decreases the expected number of packets marked \textit{out} in the queue at the bottleneck. When congestion occurs, packets marked \textit{out} get quickly exhausted for large values of $k$. Simulations results on Figure 8(b) confirm the upperbound $1 - \frac{1}{\pi}$ and show that \textit{Casablanca} discriminator closely achieves the upperbound. $A_w$ increases as $k$ increases. When $k$ is large, packets marked \textit{out} are rare and rarely get dropped on the wireless medium. Therefore, most losses appear to be random, leading to a high accuracy $A_w$. Figures 8(a) and 8(b) suggest that $k$ should be chosen around $8$ to achieve high values for $A_c$ and $A_w$ both.

Impact of $p_c$ on $A_c$ and $A_w$: It was shown in Section 3 that $k$ should be chosen such that $k \leq \frac{1}{p_c}$. If there are more packet losses at the router than the number of arrivals of packets marked \textit{out}, the router will be forced to drop packets marked \textit{in}. If this occurs, congestion losses will appear random and will be misinterpreted as wireless losses by the discriminator. Thus, for a given $k$, the accuracy $A_c$ will decrease as $p_c$ increases. In general, $p_c$ is unknown and may vary. If $k$ does not meet the requirement ($k \leq \frac{1}{p_c}$), the performance of the discriminator will be poor. In order to evaluate the impact of $p_c$ on $A_c$ and $A_w$, a non-TCP traffic is needed in order to get congestion packet loss rates higher than $5\% (p_c \geq 0.05)$. When there is only TCP traffic, the TCP connections adapt to the conditions, avoid congestion losses, and yield congestion packet loss rates lower than $5\%$. To get large congestion packet loss rates, $8$ TCP connections share the bottleneck link with a cross traffic that is on or off for periods of time that are exponentially distributed with means of $2$ ms and $1$ ms, respectively. The peak rate of the cross traffic was adjusted to a given fraction of the bottleneck bit rate. This fraction was varied from $0$ to $120\%$. Figures 9(a) and 9(b) summarize our results. The horizontal axis represents the congestion packet loss rate $p_c$ varying from $0.003$ ($0.3\%$) to $0.35$ ($35\%$). Figures 9(a) and 9(b), respectively, plot five curves each for $A_c$ and $A_w$. Each curve corresponds to a different value of $k$, with $k$ taking the values $4$, $8$, $12$, $16$, and $20$. Figures 9(a) plots the accuracy $A_c$ versus the congestion packet loss rate $p_c$.

It was shown that the upperbound on $A_c$ varies linearly as a function of $p_c$. When $p_c$ gets higher than $\frac{1}{k}$, $A_c$ was shown to be bounded by $1 - (p_c - \frac{1}{k})$. Based on this upperbound, $A_c$ accuracy will decrease smoothly because only the proportion $(p_c - \frac{1}{k})$ is incorrectly diagnosed.

Figures 9(b) plots $A_w$. For a given $k$, $A_w$ increases as $p_c$ increases. Note, however, that $A_w$ settles at some value when $p_c$ becomes significant. The reason of such a phenomenon is that the function $F(x, r, k)$ is positive or null (condition to diagnose a wireless loss) only when the ratio $\frac{r}{k}$ of the number $x$ of packets marked \textit{out} to the total number of losses $r$ is less than $\frac{1}{k}$. This happens rarely when congestion packet loss rate is too high (more than $20\%$): packets marked \textit{out} rarely get through the bottleneck due to the biased queue management.

Impact of the Bottleneck Buffer Size on $A_c$ and $A_w$: It was shown in Section 3 that the accuracy $A_c$ will decrease as the buffer size at the bottleneck decreases. With a smaller buffer, there will be fewer packets marked \textit{out} in the buffer. When congestion occurs, the packets marked \textit{out} will get exhausted faster, eventually forcing the bottleneck to drop packets marked \textit{in}. The queue management policy will not be sufficiently biased anymore, making congestion losses appear as random losses. Therefore, these losses will be mistaken as wireless losses, decreasing the accuracy $A_c$ of our discriminator. Figures 10(a) and 10(b) plot four curves each for $A_c$ and $A_w$, respectively. Each curve cor-
The horizontal axis represents the buffer size at two curves for TCP-Perfect: the expected improvement in the stream: with high that will be identified as wireless losses, leading to higher accuracy. The derived expression remains the same for TCP-Perfect and TCP-Newreno: the formula $Imp(p_c, p_w, 1.0, 1.0)$ assumes that $p_c$ remains the same for TCP-Newreno and TCP-Perfect. When $p_w > 0.01$, the throughput is mainly driven by wireless losses: TCP-Newreno backs off for all of them ($A_w = 0$) leading to very few congestion losses. The formula $Imp(p_c, p_w, 1.0, 1.0)$ assumes that $p_c$ remains the same: TCP-Perfect can theoretically keep increasing its window size without getting congestion losses, leading to a very high throughput. This assumption is wrong.

Figure 11(b) plots two curves for TCP-Casablanca: improvement $Imp(p_c, p_w, 0.99, 0.83)$ based on Equation (4) and the measured improvement obtained through simulations. For these experiments $p_c$ is approximated as $p_c(p_w) = 0.0005 e^{-0.483 \log(p_w)}$. The measured improvement for TCP-Casablanca matches quite well $Imp(p_c, p_w, 0.99, 0.83)$.

### 7.3 Throughput Improvement versus rtt

Figures 12(a) plots two curves: improvement $Imp(rtt)$ versus the round trip time for TCP-Perfect measured using simulations and the expected improvement $Imp(p_c, p_w, A_c, A_w)$. The discriminator is perfect, so $A_c = 1$ and $A_w = 1$. The measurements presented are for $p_w = 0.01$. The congestion packet loss rate $p_c$ decreases as the round trip time increases. Based on the measurements of $p_c$, $p_c$ is approximated as $p_c = 0.0007 + 0.020 \times e^{-5127 \times rtt^2}$. The horizontal axis represents the round trip time in seconds (s) from 0.012 s to 0.380 s. The vertical axis represents the percentage of improvement $Imp$. Observe that the derived expression for the improvement $Imp(p_c, p_w, A_c, A_w)$ is quite close to the simulation measurements. The improvement increases with rtt because it is well known that the recovery time for TCP from losses increases as the round trip time increases. The impact of a perfect discriminator is marginal for small round trip times. As the round trip time increases, a normal TCP is more penalized with halving its congestion window size in response to wireless losses. For high round trip time, the improvement seems to flatten around 220%
for TCP-Perfect and 105% for TCP Casablanca. The reason of the flattening can be justified by the closed form expression for the improvement:

\[
\lim_{RTT \to \infty} Imp = 100 \left( 1 + \frac{p_c + p_w}{A_c p_c + (1 - A_w) p_w} - 1 \right)
\]

### 7.4 TCP-Casablanca, TCP-Westwood, and TCP-NewReno (+ECN)

TCP-Casablanca requires changes at the endpoints and the intermediary nodes. Two TCP variants that do not require such changes are considered: TCP-Westwood and TCP-Newreno with ECN. TCP Westwood [18] is an end-to-end TCP variant that attempts to evaluate the available bandwidth and accordingly adjust the congestion window size. ns-2 (version 2.1b9a) was extended by the authors with TCP-WestwoodNR (TCP-Westwood based on TCP-Newreno to handle multiple losses within the same window). TCP-WestwoodNR code and related resources were downloaded from the official site of TCP Westwood at UCLA. In the following, TCP-WestwoodNR is called TCP-Westwood. Under ideal conditions (i.e., no random cross-traffic and high round trip time), TCP-Westwood is efficient and yields in our simulations up to 300% improvement. TCP-Westwood performance is impressive as TCP-Westwood advantage increases as the wireless cross-trace on the forward path increases, TCP-Westwood performance decreases. If the random cross-traffic is on the reverse path, TCP-Westwood does not perform as well as when random cross-traffic on the forward path. Cross traffic on the reverse path defeats in general TCP-Westwood.

The performance of TCP-Newreno is measured with using RED routers tuned to get the best throughput improvement. TCP-Newreno (+ECN) performs quite well when there is no random cross traffic. Degradation in performance is observed as the cross traffic increases. When there is no congestion, ECN is of no help to TCP-Newreno for reacting appropriately to wireless losses.

Figures 13(a) and 13(b) plot the improvement achieved when using TCP-Casablanca, TCP-Westwood, or TCP-Newreno(+ECN). Figures 13(a) plots the throughput improvement when there is no cross traffic. TCP-Westwood outperforms TCP-Casablanca and TCP-Newreno(+ECN).

### 7.5 TCP-Casablanca Friendliness

This section considers the coexistence of TCP-Casablanca flows with TCP-Newreno flows where TCP-Newreno senders do not mark out any packet. It is then expected that when congestion occurs at the bottleneck, the router will first drop packets marked out before starting to drop packets from TCP-Newreno flows. TCP-Casablanca performance is measured in presence of TCP-Newreno flows. TCP-Casablanca performance is measured in presence of TCP-Newreno flows. Experiments were run with \( N_1 \) TCP-Newreno TCP connections sharing the same bottleneck link with \( N_1 \) TCP-Casablanca TCP connections. All TCP connections share a last hop wireless link with error loss rate \( p_w \). Extensive simulations were conducted under different round trip times, bottleneck bandwidth, cross-traffic intensity, and different values for \( N_1 \). Figure 14(a) presents the ratio \( \frac{\text{Thgt(Casablanca)}}{\text{Thgt(Newreno)}} \) of TCP-Casablanca throughput over TCP-Newreno throughput versus the wireless error rate \( p_w \). First, observe that this ratio is sensitive to the number of connections \( N_1 \). As \( N_1 \) increases, contention over the bottleneck link leading to more congestion losses. As congestion increases, the ratio \( \frac{\text{Thgt(Casablanca)}}{\text{Thgt(Newreno)}} \) decreases. This is expected because the biased queue management drops more packets for TCP-Casablanca than for TCP-Newreno. However, TCP-Casablanca advantage increases as the wireless loss rate \( p_w \) increases. When congestion is light (\( N_1 = 1 \) or \( N_1 = 2 \)), TCP-Casablanca achieves better throughput than TCP-Newreno with any wireless error rate \( p_w \). But, when congestion is high (e.g., \( N_1 = 16 \)), TCP-Newreno achieves better throughput than TCP-Casablanca when wireless loss rate \( p_w \) is less than 1% (0.01).
8 TCP-Ifrane: a Sender-based TCP-Casablanca

TCP-Casablanca requires changes at the sender, the receiver, and intermediate network elements. The receiver is the best place to detect, analyze, and diagnose losses while the sender is the place where the appropriate action must be taken. This section proposes a lighter version of TCP-Casablanca that requires changes only at the sender: TCP-Ifrane. TCP-Ifrane does not require any change at the receiver and is implemented at the sender as follows: whenever the sender gets a dupack for some packet P, it detects the occurrence of a loss. TCP-Ifrane classifies this as follows: if the packet P was marked out (when sent), then the sender classifies it as a congestion loss, otherwise (if the packet P was sent marked in) the sender classifies it as a wireless loss. Note that when multiple losses occur in the same window of packets in flight, the sender has perfect knowledge only for the loss with the smallest sequence number. Subsequent lost packets cannot be identified with certainty by the sender. Such a simple discriminator provides a kind of lower bound on the accuracy that could be achieved. Extensive simulations of TCP-Ifrane were performed: the results are quite encouraging. Figure 15 presents how TCP-Ifrane performs in comparison to TCP-Casablanca. The vertical axis represents the ratio of the accuracy of congestion losses $A_c(Ifrane)$ of TCP-Ifrane over the accuracy $A_c(Casablanca)$ of TCP-Casablanca. Figure 15(a) presents the ratio $A_c(Ifrane) / A_c(Casablanca)$ for congestion losses while Figure 15(b) presents the ratio $A_w(Ifrane) / A_w(Casablanca)$ for wireless losses accuracy. The horizontal axis shows the wireless error rate $p_w$ on a logarithmic scale. Figure 15(a) shows that for any value of $k$ (4, 8, or 16), the ratio $A_c(Ifrane) / A_c(Casablanca)$ is lower than 1: the accuracy $A_c$ for TCP-Ifrane is lower than $A_c$ for TCP-Casablanca. However, for $(k=4)$ and $(k=8)$, the ratio $A_c(Ifrane) / A_c(Casablanca)$ remains higher than 0.93. For $k=16$, the ratio is about 0.75 to 0.85. Note that the accuracy $A_c(Casablanca)$ is quite high for $k=4$ and $k=8$, while it is pretty low for $k=16$. On the contrary, Figure 15(b) shows that the ratio $A_w(Ifrane) / A_w(Casablanca)$ is always higher than 1: the wireless accuracy of TCP-Ifrane is always better. The throughput improvement with TCP-Ifrane is marginally higher than TCP-Casablanca for two reasons: (1) TCP-Ifrane accuracy $A_c$ is slightly lower than $A_c$ for TCP-Casablanca, (2) TCP-Ifrane accuracy $A_w$ is slightly higher than $A_w$ TCP-Casablanca. Due to reasons (1) and (2), TCP-Ifrane backs off less often than TCP-Casablanca.

Other optimizations are possible for TCP-Ifrane. With TCP-Newreno, the sender can reconstruct the real pattern of losses through the partial acks. The diagnosis can be refined as subsequent losses within the same window are recovered. Based on a refined decision, TCP-Ifrane could accordingly readjust the congestion window size.

9 Implementation

This section discusses the issues related to the implementation of the biased queue management scheme and TCP-Casablanca. There are three components that must be addressed: (1) packet marking in or out, (2) the biased queue management, and (3) the endowment of TCP with Casablanca discriminator.

Packet Marking: The idea of marking packets is not new. Bala et al [1] used tagging (green and red packets) to implement a preventive congestion control mechanism. Frame relay and ATM have mechanisms to specify loss preference. A mechanism must be offered to TCP to request from the network layer (IP) different dropping priorities for the same connection. There are many mechanisms to mark packets in order to request different dropping priorities from the network layer. Recently, RFC 2475 defined an architecture to offer differentiated services (DiffServ). The DS codepoint, as defined in RFC 2474, could be exploited. The DS codepoint, set on a packet, is a byte composed as follows: the 5 leftmost bits specify the service expected from the core routers. A codepoint number could be assigned for the biased queue management. The two remaining rightmost bits are unused: they can be used to tag the packet in or out.

However, packet marking may well be specified without any reference to existing standards such as DiffServ. Every packet P could bear an N-bit vector V – the vector can be included in an option field in the header. The N-bit vector V would describe the marking pattern of the N most recent packets sent before P. For example, if k is chosen to be 8 (one packet is marked out every 8 packets), a byte will suffice for vector V with $N = 8$.

Biased Queue Management: The biased queue management can be defined within the DiffServ architecture [7] as a per hop behavior (PHB) associated with a specific DS codepoint. This biased queue management PHB can be defined as follows: when there is no congestion, all packets are processed exactly the same, i.e., packets marked out are treated exactly the same as packets marked in. But, when congestion occurs, the switch (router) must select a packet marked out to drop as long as there are still packets marked out in the queue.

A key issue is fairness between flows that do not mark out...
any packet and those who mark out some of their packets. It was shown in Section 7.5 that when there are no wireless losses and congestion is heavy, TCP-Casablanca does not perform well: its packets marked out are dropped first before packets from a TCP-Newreno flow are. The authors are investigating how the biased queue management could be coupled with an active queue management scheme such as CHOKe \cite{21} to enforce fairness.

**Short-lived TCP NewReno:** For short lived TCP connections, the problem is that congestion window may remain so small that outstanding packets marked out are rare. If congestion windows of TCP senders remain small (less than \( k \) outstanding packets) then some senders may not have packets marked out in flight. Since TCP does not always start with sequence number \( 0^4 \). There are multiple trivial mechanisms to insure that most short-lived TCP connections with small congestion window sizes have an outstanding packet marked out.

10 Conclusion and Future Work

The problem of distinguishing congestion losses from random wireless losses can be solved by “de-randomizing” congestion losses using a biased queue management. The “de-randomization” enables the design of an efficient discriminator: Casablanca achieves high accuracies diagnosing congestion losses \( (A_c \geq 0.95) \) and random wireless losses \( (A_w \geq 0.75) \).

The expression of the expected improvement is established for TCP when using a discriminator with accuracies \( A_c \) and \( A_w \). The established expression captures the bell shape of the improvement versus the wireless packet loss rate \( p_w \). Previous expressions based on the simplistic expression of TCP improvement derived in \cite{19} do not exhibit the bell shape of the improvement versus \( p_w \). The conditions under which it is worth distinguishing congestion losses from random wireless losses are identified. The improvement in throughput is shown to depend more on the accuracy \( A_w \) than \( A_c \). However, for the sake of the network, this work favors a very high accuracy \( A_c \) with a reasonable accuracy \( A_w \). It is shown that a high percentage of improvement can be achieved under specific conditions related to the congestion packet loss rate \( p_c \), the wireless loss rate \( p_w \), and the round trip time.

The value of \( k \) can be chosen dynamically for long-lived TCP connections: at the beginning of a connection, the \( k \) value could be set to a small value such that \( A_c \) is close to 1 and \( A_w \) close to \( 1 - \frac{1}{k} \). A small value of \( k \) guarantees that the scheme works for high congestion packet loss rates. Under these conditions, a good estimate of \( p_c \) can be made, leading to the choice of a more appropriate value for \( k \) with \( k \leq \frac{1}{p_c} \).

For future work, the function \( F(x, r, k) \) can be improved to yield a better accuracy \( A_w \), a key factor in improving the performance of TCP in presence of random losses.

The biased queue management raises the issue of fairness between flows that mark some of their packets out and the flows that do not. If some flows do not mark out any of their packet, then packets will be dropped only from flows that mark packets out. Only flows that mark packets out would be responsive to light congestion. Flows that do not mark packets out may monopolize the available link capacity.

The authors are working on an extension of this work to generalize the biased queue management to multiple dropping priorities (instead of just in and out). Multiple dropping priorities, used with a biased queue management, may well yield a higher wireless accuracy \( A_w \) and open the door to sound congestion control mechanisms.

Saad Biaz received a Ph.D. in Computer Science in 1999 from Texas A&M University and a Ph.D. in Electrical Engineering in 1989 from the University Henri Poincaré in Nancy (France). He is presently an Assistant Professor of Computer Science and Software Engineering at Auburn University. He has held faculty positions at the Ecole Supérieure de Technologie de Fès and Al Akhawayn University in Ifrane (Morocco). His current research is in the areas of distributed systems, wireless networking, and mobile computing. His research is funded by the National Science Foundation. Saad Biaz is a recipient in 1995 of the Excellence Fulbright Scholarship. Saad has served on the committees of several conferences and as editor for several journals. He is a member of the IEEE. For more information, please visit http://www.eng.auburn.edu/users/sbiaz.

Nitin Vaidya received the Ph.D. from the University of Massachusetts at Amherst. He is presently an Associate Professor of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign (UIUC). He has held visiting positions at Microsoft Research, Sun Microsystems and the Indian Institute of Technology-Bombay. His current research is in the areas of wireless networking and mobile computing. His research has been funded by various agencies, including the National Science Foundation, DARPA, Motorola, Microsoft Research and Sun Microsystems. Nitin Vaidya is a recipient of a CAREER award from the National Science Foundation. Nitin has served on the committees of several conferences, including as program co-chair for the 2003 ACM MobiCom and General Chair for 2001 ACM MobiHoc. He has served as editor for several journals, and presently serves on the IEEE Transactions on Mobile Computing editorial board, and as editor-in-chief of ACM SIGMOBILE periodical MC2R. He is a senior member of the IEEE and a member of the ACM. For more information, please visit http://www.crhc.uiuc.edu/~nhv/.

References


