Efficiency/Friendliness Tradeoffs in TCP Westwood

Ren Wang, Massimo Valla, M.Y. Sanadidi, Bryan Kwok Fai Ng, Mario Gerla
UCLA Computer Science Department, Los Angeles, CA 90095, USA
E-mail: {renwang,mvalla,medy,bryanng,gerla}@cs.ucla.edu

Abstract
In this paper, we propose a refinement of TCP Westwood allowing the management of the Efficiency/Friendliness-to-NewReno tradeoff. We show that the refined TCP Westwood is able to achieve higher efficiency yet at the same time maintain friendliness. TCP Westwood (TCPW) implements a novel window congestion control algorithm based on available bandwidth estimation. The performance of TCPW has been promising, exceeding that of TCP NewReno in high speed and/or wired/wireless networks. However, under certain circumstances, TCP NewReno may experience some performance degradation because TCPW possesses more information and thus can take better advantage of available bandwidth. In this paper we propose combining the original TCPW sampling strategy that produces available Bandwidth Estimates (BE), with a new strategy that produces Rate Estimates (RE). Our studies show that RE works best when packet losses are mostly due to congestion. If, on the other hand, the packet losses are mostly due to link errors, BE gives better performance. To achieve the "best of all worlds", we introduce a method we call Combined Rate and Bandwidth estimation (CRB). A connection first infers the predominant cause of packet losses, and then uses the most appropriate estimation method. We also introduce the Efficiency/Friendliness Tradeoff Graph that provides better tradeoff visualization. In our experiments, we found that CRB provides a better compromise between efficiency and friendliness, and the means to manage such a tradeoff.

1. Introduction

The Transmission Control Protocol (TCP) protocol provides end-to-end, reliable, congestion controlled connections over the Internet [CK74]. The congestion control method used originally in TCP Tahoe included two phases: slow-start and congestion avoidance [Jaco88]. In TCP Reno, recovery from sporadic packet losses is enhanced by Fast Retransmission and Fast Recovery [Jaco90]. SACK-based TCPs [RFC2883], [FF96], provide the sender with more complete information about which packets are lost. Another class of algorithms is referred to as "NewReno" [RFC2582] [Hoe96], which does not need SACK information and requires only modification on the sender side. Research shows that the majority of TCP implementations are NewReno [PF01]. Therefore, we implemented TCPW and its refinement variants with NewReno as a base.

Increasingly, TCP is called upon to provide reliable and efficient data transfer over a variety of link technologies including wired and wireless with increasing bandwidth capacity. The new ultra high speed wired/wireless environment is exceeding the range for which TCP was initially designed, tested and tuned. As a consequence, active research is in progress to extend the domain of effective TCP operability [GMLW99][KVM99][BSAK95] [BK98].

The use of path conditions estimate for enhancing congestion control in TCP has been proposed; e.g. Packet Pair [Kesh91] and TCP Vegas [BP95]. In TCP Vegas, the sender infers the network congestion level from observed changes in round trip time (RTT). If RTT becomes large, the source will decrease its congestion window (cwnd), thus reducing its transmission rate. Some fairness issues with TCP Vegas were pointed out in [HMM98] [BB00]. It was argued there that new arriving connections to a congestion in progress may not be able to get a fair share of the bottleneck bandwidth. In the Packet Pair scheme, a sender estimates the bottleneck backlog and adjusts its sending rate accordingly. However, the Packet Pair scheme explicitly assumes Round-Robin scheduling at the routers – a feature not available in many commercial routers.

Several network and link-layer enhancements have also been proposed to improve TCP performance under various conditions (congestion loss, random loss, handoff, out of order delivery, etc.), such as Random Early Detection (RED) [FJ93], Snoop [BPSK97], Explicit Congestion Notification (ECN) [Floy94] and Explicit Loss Notification (ECN)[BK98].

TCP Westwood [CGMSW01] design adheres to the end-to-end transparency guidelines set forth in [Clar88] and requires only sender side modification. The key innovation of TCPW is to use a bandwidth estimate directly to drive cwnd and ssthresh. The current estimation method in TCPW is based on “bandwidth Estimation”, i.e., BE. This TCPW BE strategy provides significant throughput gains, especially the large leaky pipes [CGMSW01]. Under certain congestion circumstances, BE exceeds the fair share of a connection resulting in possible unfriendliness to TCP NewReno connections. This friendliness/efficiency tradeoff is addressed in this work.

We present a refinement of standard TCPW where two estimators are maintained, along with a method to identify the predominant cause of packet loss. Depending on the outcome, the appropriate estimator is used. Both estimators use information obtained from ACKs received at the sender. One estimator BE, as in the standard TCPW, considers each ACK pair separately to obtain a bandwidth sample, filters the samples into a low-pass filter and returns as a result the available bandwidth that the TCP connection is estimated to be getting from the network. The other estimator we propose in this paper, called Rate Estimator (RE), considers the amount of data acknowledged during the latest interval of time T as sample, then feeds such samples into an appropriate low-pass filter to get the estimated rate, in this
case tending to estimate the throughput that the TCP Westwood connection has recently experienced. We call this TCP Westwood refinement Combined Rate-Estimation/ Bandwidth-Estimation (CRB).

To identify the predominant cause of packet loss, ECN and ELN can be used. However, ECN requires all the routers along a network path to support ECN, while ELN has its share of implementation problems as reported in [BP97]. We propose instead a method to identify the predominant cause of packet loss that does not require support from lower layers. We use the relationship between the current congestion window value and the estimated pipe size, the latter being defined as the product of RE and the minimum RTT observed. The pipe size corresponds to the ideal window required to achieve the rate RE. When the measured pipe size is significantly smaller than cwin, it is very likely that packet losses are due to congestion. An analysis of this relationship and a simulation study of this method are presented in Section 2.

The remainder of the paper is organized as follows. In Section 2 we introduce the two proposed estimators, use an “open loop” analysis along with simulation results to determine the semantics and the accuracy of the estimates they provide, and explain the differences between them. In Section 3 we present our CRB algorithm, and the method to identify the more likely cause of packet loss and its use in selecting an estimator for the present window adjustment. Section 4 presents a performance study under different network environments. A study on fairness, friendliness and link utilization is also presented. Finally, in Section 5, we provide concluding remarks and identify areas of future research.

2. Overview of the Two Estimators

In this section we discuss two sampling methods and compare their accuracy under different conditions. To start with, we provide a definition of fair bandwidth share. We then analyze the two sampling methods, and evaluate their estimates relative to the fair share estimate. TCP design aims to utilize all available bandwidth, while maintaining “fairness” in the allocations made to different flows. In this paper, we focus on bandwidth fair sharing of TCPW and TCP NewReno, both among themselves and against each other. To focus on the essential behavior of TCPW and TCP NewReno, we use the standard model where TCP flows share a single bottleneck link with bandwidth C. We also assume infinite backlog for all TCP sources, and assume that all flows have identical propagation delays. Thus fairness is achieved by equally allocating the available bandwidth to active TCP flows, unless some of them are inherently unable to use their share regardless of the existence of competing flows. For instance, on a “leaky” large pipe, NewReno utilization is dramatically reduced [LM95]. In this case, a flow using a new proposed protocol can achieve higher bandwidth share and preserve fairness. However, this should be accomplished without reduction in the legacy connections throughput.

We define the fair bandwidth share of a connection for the following cases:

- (a) N TCPW flows sharing a bottleneck link with capacity C. The fair share is C/N.
- (b) A total of N TCPW and TCP NewReno flows. Assuming that no random errors are possible, both protocols are roughly equivalent, and therefore the fair share for each flow is C/N.
- (c) Assuming that there are random errors on the path (e.g. from a wireless link) to which all flows are subjected. Because TCP NewReno flows are inherently unable to utilize the link capacity in this case, TCPW flows should not be considered aggressive by getting a larger bandwidth share than the NewReno flows. We define the fair share of a NewReno flow as the same value if all flows are TCP NewReno. For instance, suppose the fair share of the NewReno flow is Sr given that there are total N homogenous TCP NewReno flows, then when this total N flows includes some TCPW flows, the fair share of NewReno flow should remain Sr, while TCPW flows could have a fair share of higher value. Thus, a TCPW fair share can be higher than a NewReno share since the latter connection is inherently incapable of using the link capacity.

In TCPW, we proposed an available Bandwidth Estimation (BE) [CGMSW01] algorithm used by a TCPW sender to drive its cwnd and ststhresh. Let us call this version of TCP, TCPW BE. This protocol has been shown to achieve a high utilization when used over large leaky pipes. As will show below, in certain cases, BE may overestimate its fair share. In this case, TCP NewReno (and other TCP-like protocols) may experience performance degradation. For this reason we introduce and study here TCPW RE (Rate Estimation) to address the issue of friendliness to NewReno. The objective of the remainder of this section is to present and compare the two estimations and explain the rationale behind them. Both estimations are based on the ACK arrival process received by the TCP-W sender; thus, they are passive and introduce no extra link overhead.

2.1. Available Bandwidth Estimation (BE)

A TCPW sender uses ACKs to estimate BE. More precisely, the sender uses the following information: (1) the ACK reception rate and, (2) the information an ACK conveys regarding the amount of data recently delivered to the destination. For details regarding the processing of ACKs in (2), please refer to [CGMSW01]. Assume that an ACK is received at the source at time t, notifying that dl bytes have been received at the TCP receiver. We can measure the following sample bandwidth used by that connection as:

\[
\text{bandwidth estimate} = \frac{d_l}{t_k - t_{k-1}}
\]

where \(t_k\) is the time the previous ACK was received. Letting \(\Delta_t = t_k - t_{k-1}\), then:

\[
b_k = d_l / \Delta_t.
\]

For averaging over time the bandwidth samples obtained as above we have proposed in [CGMSW01] the following filtering, which is a discrete version of a continuous first order low-pass filter using the Tustin approximation [Ait97].

Let \(b_k\) be the bandwidth sample, and \(\tilde{b}_k\) the filtered estimate of the bandwidth at time \(t_k\). Let \(\alpha\) be the time-varying exponential filter coefficient at \(t_k\). The TCPW filter is then given by:

\[
\tilde{b}_k = \alpha_k \tilde{b}_{k-1} + (1 - \alpha_k) \left( \frac{b_k + b_{k-1}}{2} \right),
\]

where \(\alpha_k = \frac{2\tau - \Delta t_k}{2\tau + \Delta t_k}\), and \(1 / \tau\) is the filter cut-off frequency. For more details on this filter please refer to [CGMSW01].

2.2. Rate Estimation (RE)

Significant efficiency improvements are obtained using the BE estimator produced by the sampling and filtering methods above. This is particularly true in environments with...
...large leaky pipes. Further, note that when routers employ a round robin policy in scheduling transmissions, BE is accurate in estimating a connection fair share. However, for drop-tail routers, since TCP traffic tends to be “bursty”, i.e., sending out a full window of packets and then waiting for the acknowledgements, BE may over-estimate the connection fair share. The problem appears to be with the bandwidth sample definition above.

Consider an alternative bandwidth sample, defined as the amount of data reported to be delivered by all ACKs that arrived in the last T time units, divided by T. For reasons to be explained below, we call this method Rate Estimation (RE). This alternative is identical to the earlier TCPW sample definition if the ACKs are uniformly spaced in time. Simulation and measurements, however, show that ACKs tend to cluster in bursts. Thus, the BE sampling method “overestimates” the connection fair share, while providing (in the bursty case) a reasonably good estimate of the available bandwidth at the bottleneck. Thus, BE is more effective in environments with random error, and when single connection efficiency is paramount.

Let us define the Rate Estimation sample. The RE sample associated with the kth received ACK is expressed by:

\[ RE_k = \frac{\sum_{i \leq j < T} d_j}{T}, \]

where \( d_j \) is the amount of data reported by ACK j. Similarly, at the previous time instant, k-1, the sample k-1 is:

\[ RE_{k-1} = \frac{\sum_{i \leq j < T} d_j}{T}. \]

Therefore,

\[ RE_k - RE_{k-1} = \frac{1}{T} \left( \sum_{i \leq j < T} d_j - \sum_{i \leq j < T} d_j \right) \]

Which on rearrangement gives,

\[ RE_k = RE_{k-1} + \frac{1}{T} \sum_{i \leq j < T} d_j. \]

Thus, at any instant, a “sliding window” of length T is used to obtain a bandwidth sample.

The expression above is a recursive one, because the sample is calculated using its previous value as reference. Additionally, the technique places equal emphasis on all data points in the sampling range. Thus a value in the near past will have the same influence as a more current measurement when calculating the sample. This is a desirable feature when we are dealing with bursty TCP traffic in presence of congestion. Finally, sliding window samples are exponentially averaged in order to obtain a smoothed bandwidth share estimate over time. We use here a simple exponential averaging filter to calculate the Rate Estimate at the instant the kth ACK is received as:

\[ RE_k = \frac{19}{21} RE_{k-1} + \frac{1}{21} (RE_k + RE_{k-1}) \]

2.3. Comparison of BE and RE Estimates

To capture the essence of the two estimators, we use the network topologies shown in Figure 1. All the simulated results presented in this paper are obtained using Network Simulator ns2 [ns2]. The bottleneck link is shared by 2 TCP flows: a NewReno flow between nodes 2 and 4, and a TCPW flow between nodes 3 and 5. Flows are assumed to have identical propagation delay and infinitely backlogged flows. A NewReno flow between nodes 2 and 4, and a TCPW flow between nodes 3 and 5. Flows are assumed to have identical propagation delay and infinitely backlogged flows.

Figure 1. Network topology for open loop analysis

The ACK arrival process at node 3 is affected by two different factors: first, it is related to the sending process. Second, it is influenced by the queuing delays at the bottleneck router. Such delays depend on both the scheduling policy used by the router and the amount of “background” traffic, in this case, TCP NewReno flow sent from node 2. We assume the scheduling to be first-come-first-serve, which is the policy found on most routers today.

For a TCP flow, the sender tends to receive the ACKs in bursts due to the following two reasons: first, the sender sends packets in bursts as a consequence of the slow-start phase of the TCP protocol, in which packets belonging to the same window will be sent close to one another, while packets belonging to different windows will be separated. Second, the first-come-first-serve policy at the router can introduce further gaps (or increase the existing gaps) between groups of packets of the same flow having to wait for packets from other flows to be served by the router.

Let us assume that the TCPW sender will receive the kth ACK at time instant \( t_k \), confirming that \( d_k \) bytes are correctly received by the receiver. Let us define \( \Delta t \) the inter-ACK time elapsed between the \((k-1)\)th and the K-th ACK (same as in Section 2.1).

First consider that if two packets are sent back to back by the TCPW sender and not separated due to background traffic, then the corresponding ACKs would come back to the sender with an inter-ACK delay \( \Delta t \) such that \( b_k = d_k / \Delta t \) will be equal to \( C \), the bottleneck bandwidth. Note that \( b_k \) is exactly the sample used by the BE estimator.

In the case of multiple connections, the ACKs would likely arrive in bursts separated by gaps during which the server was occupied by other flows. For this reason the filter will receive bursts of bandwidth samples \( b_k \) interleaved by other lower bandwidth samples, and the total effect is that the BE may overestimate its fair share on the bottleneck link. In RE, a sample is calculated by dividing by \( T \) the number of bytes acknowledged in an interval of length \( T \). The RE sample is thus a smoother sample, and thus it produces an estimate that is closer to the throughput of the TCPW connection.

2.3.1. Estimation With No Random Errors on the Paths

In Figure 2, we compare the BE and RE estimations. The plot was obtained applying the BE and RE estimators to the ACK process received by the TCPW sender and generated using the previously described network configuration with the following parameters: bottleneck bandwidth \( C = 5 \text{ Mbps} \), packet size = 1400 bytes (fixed), round trip time \( \text{RTT} = 70 \text{ ms} \), buffer size (B) = pipe capacity which is equal to 32 packets; time interval \( T = 4 \text{ RTT} \). The link is error free and no packet is dropped in initial seconds due to congestion.

As seen in Figure 2, the BE is estimating about 3.6 Mbps, which is larger than its 2.5 Mbps fair share, while the RE estimator is estimating exactly the fair share value.
As we mentioned before, the BE and RE estimations would coincide if a round-robin scheduling policy was used by the router; in this case packets from the two flows would be served by the router alternately, the corresponding ACKs would come back to the sender equally spaced (expect for the gaps due to the slow start phase) and the BE estimate would coincide with the recent throughput estimate RE.

In conclusion, the BE estimator may overestimate the fair share because of the effects of the samples $b_k \equiv C$ generated by the bursts of packets served back-to-back by the router. The RE estimator gives a more accurate measure of the fair share in this case since its estimate is closer to the throughput recently achieved by the flow. On the other hand, RE could underestimate the connection fair share if the path suffers from random errors, and we will evaluate this case by simulation in Section 2.3.2.

The next simulation experiments include two TCPW connections sharing an error free bottleneck link. To illustrate the dynamics of the estimates, we maintain one connection active throughout the simulation run, while the other one is enabled and disabled twice during the simulation experiment. From Figure 3(a), we can see that when two connections are running, BE has a mean estimate of 3.1Mbps, which is an overestimate of the fair share (2.5 Mbps). However, Figure 3(b) shows that RE provides a more accurate estimate equal to the fair share. It is important to note that in either case the throughputs obtained by two connections are fair since the connection estimates are equal to each other. This is true for both RE and BE. One observation is that the two estimates oscillate due to the TCP’s bursty transmission. Nevertheless, the connections share the bandwidth fairly. An important observation here is that the connection starting second readily acquires its fair share against the established connection.

The results reported above in Figure 2 and Figure 3, all confirm that the use of RE is more appropriate when packet losses are due to congestion.

### 2.3.2. Estimation in Presence of Random Errors on the Path

We present simulation results to study scenario (c) defined in the beginning Section 2. First we run one TCPW and one NewReno flow sharing the bottleneck link with a random error rate of 0.5%. Estimates provided by BE and RE are shown in Figure 4. The fair share of TCPW is actually larger than 2.5Mbps in this case, since NewReno is inherently unable to utilize the link capacity. Through simulation, when two NewReno flows share the link, the throughput obtained by each flow is 1.4Mbps. Thus, in this configuration, TCPW flow can take as much as 3.6Mbps out of total 5 Mbps without hurting the NewReno flow. From Figure 4, RE estimate settles at about 1.7 Mbps, underestimating its fair share, while BE estimates a reasonable 3.4 Mbps. The results confirm that BE should be used when random error is the cause of packet losses.

![Figure 2. BE and RE with concurrent TCP NewReno connection](image)

![Figure 3. Estimate with Concurrent TCP Connection](image)

![Figure 4. BE and RE with concurrent NewReno connection (error rate 0.5%)](image)

### 2.3.3. Estimation When Available Bandwidth Fluctuates Due to Non-Adaptive Traffic

Using the same network configuration, we first examine how the two sampling methods perform when a single TCPW connection shares a bottleneck link with non-adaptive UDP traffic with time varying intensity. The UDP traffic is used here to simulate non-adaptive high priority traffic. Both BE and RE are able to track the bandwidth left unused by the UDP traffic. The actual available bandwidth and the estimated connection share are shown in Figure 5(a) and Figure 5(b). The results show that both methods track the available bandwidth accurately, with RE providing a more stable tracking. This is because the BE sample is sensitive to whether the packets are clustered, but the RE sample is not.

Figure 6 illustrates the results for a link shared by one NewReno, one TCPW, and one non-adaptive UDP flow with a fixed rate of 2Mbps. The shared bottleneck is the same as that in Figure 4, with capacity of 5Mbps and 0.5% link error rate. A simulation experiment showed that the fair share a NewReno flow can get is 1.1 Mbps when two NewReno flows share the link with the UDP flow. Thus, the fair share of a TCPW equals 5-2-1.1=1.9 Mbps. RE estimates fluctuate near 1.3Mbps, under its fair share, while BE has more accurate estimates around 2Mbps.

![Figure 5. Estimate with concurrent UDP traffic](image)

![Figure 6. BE and RE with concurrent NewReno and UDP (2Mbps) connection (error rate 0.5%)](image)
3. Combined RE/BE (CRB) Method

We addressed above the question of which estimate – achieved rate or available bandwidth – yields better bandwidth estimates. BE is more effective in environments with random error. On the other hand, the RE method appears to be more appropriate when packet losses are due to congestion (router buffer overflow).

Based on the tradeoff presented above, a hybrid method that combines both sampling strategies would be of interest, provided we can determine the cause of packet loss: errors or buffer overflow. This issue of course is beyond the sampling methodology investigation itself. In fact, if a sender were able to distinguish between error and buffer overflow losses, the sender reaction to the former would be to retransmit immediately with no change of window size. For our immediate purposes, it suffices to be able to determine the predominant cause of loss, and then select the sampling strategy accordingly. Below we present a method to determine the predominant cause of packet loss requiring no assistance from layers below TCP.

3.1. Identifying Predominant Cause of Packet Loss

We propose to use the relationship between the current cwin value and the estimated pipe size, the latter indicated by the product of RE and the minimum RTT. When RE * RTTmin is significantly smaller than cwin, it is more likely that packet losses are due to congestion. This is because the connection is using a cwin value much higher than its share of pipe size, thus congestion is likely. Figure 7(a) shows the relationship between the two quantities in a simulation experiment where congestion is the only cause of packet loss (no link errors).

The simulation includes two connections competing for a 5Mbps bottleneck link with no random error. The RTT is set to 70ms, and buffer size to 32 packets (equal to pipe size). Figure 7(a) shows that the RE* min RTT is much lower than the cwin at the point of loss due to congestion.

Next we consider an environment where a single connection is running over a bottleneck link with 2Mbps capacity, 100ms RTT, and 1% packet loss rate due to random link error. Figure 7(b) shows that although there is no significant difference between RE * min RTT and cwin, cwin is generally smaller than RE*RTTmin when a loss happens.

3.2. CRB Algorithm

In CRB, whenever a packet loss is indicated, the sender determines the predominant cause of loss as follows: when the ratio RE *RTTmin to cwin exceeds a threshold value θ, the use of RE is indicated. Below θ, BE is indicated. In our experiments, a threshold of θ=1.4 was found to give the best results. A packet loss is indicated either by a reception of 3 DUPACKs or a coarse timeout. Below we present the TCPW CRB algorithm to set ssthresh and cwin after a packet loss indicated by three duplicate ACKs. In the pseudo-code, seg_size identifies the length of a TCP segment in bits. The value RTTmin is set as the smallest RTT estimated by TCP, using its own RTT estimation algorithm.

4. Performance Evaluation

We compare the performance of RE, BE, CRB, and NewReno. All results are obtained using the ns2 simulator [ns2][TCPW]. We first evaluate the throughput of the different protocols with lossy links (wireless) in Section 4.1. Then in Section 4.2, we study the fairness, friendliness and utilization of the bottleneck link. We introduce the Efficiency/Friendliness tradeoff graph in Section 4.2.

The summary of the results is that the combined method provides a means of adapting to loss causes, and to effectively manage the Friendliness/Efficiency tradeoff.

4.1. Throughput with Lossy Links

We begin with the evaluation of the efficiency provided by our schemes. In this section, a number of scenarios are studied to compare the performance of BE, RE, and CRB in the wired/wireless configuration shown below.

![Simulation topology](image)

Figure 8. Simulation topology

Figure 8 shows a mixed network, where the wired portion has capacity of 45 Mbps and one-way propagation time of 35ms (roughly the delay from West to East coast.) The wireless portion of the network is a very short 11-Mbps wireless link with a propagation time of 0.01ms (e.g., WaveLAN.) The wireless link is assumed to connect a base station to a destination mobile terminal.

![Throughput vs. packet loss rate of the wireless link](image)

Figure 9. Throughput vs. packet loss rate of the wireless link

The throughput of BE, CRB, RE and NewReno are compared under packet loss rate varying from 0 to 5%. This range is similar to that used in [HPSK97]. The results in Figure 9 show that BE, RE and CRB throughput is higher than that of NewReno. The largest improvement is obtained...
around 0.1% to 1% loss rate, where BE gains up to 320% and CRB 267% over TCP NewReno. RE provides improvement also, but only about 190% over NewReno.

To assess the relation of the throughput gains to the E2E propagation time, we ran simulations with the wired portion propagation time varying from 0 to 200ms and the wireless link loss rate set to 0.1%. The results in Figure 10 show a significant gain for BE and CRB of up to 542% and 400% respectively, at a two way propagation time of 100ms. Again RE shows moderate improvement over TCP NewReno. When the propagation time is small (say, less than 5ms), all protocols are equally effective. This is because a small window is adequate and window optimization is not an issue. The “network aware” schemes reach maximum improvement over NewReno as the propagation time increases to about 100ms. After that, the gain starts to decrease. A potential reason for this may be that the feedback information used to estimate the available bandwidth arrives too late to be of significant help. This is a scaling problem that is worth future investigation.

![Figure 11. Throughput vs. bottleneck capacity](image)

Simulation results in Figure 11 show that BE and CRB gains increase significantly as the bottleneck link transmission speed increases. Thus, BE and CRB are more effective than TCP NewReno in utilizing the Gbps bandwidth provided by new-generation, high-speed networks. Figure 11 shows that the improvements obtained via BE and CRB increase to approximately 418% and 370% respectively when the wireless link speed reaches 20 Mbps. The error model is still Bernoulli with parameter 0.1%, and the E2E propagation time here is 70ms.

**4.2. Fairness, Friendliness and Utilization**

In this subsection, we focus on fairness, friendliness and their tradeoff with utilization. We distinguish between Fairness and Friendliness as follows. Fairness relates to the relative performance of a set of connections of the same TCP variant. Friendliness relates to how sets of connections running different TCP flavors affect the performance of each other. The simulation topology consists of a single bottleneck link with a capacity of 10 Mbps, and one-way propagation delay of 35ms. The buffer size at the bottleneck router is 64 packets (equal to the pipe size), except where otherwise stated. The link error rate varies depending on the scenario.

A set of simulations with 10 simultaneous flows was run to investigate the fairness property of the CRB scheme. The Jain’s fairness index of CRB reached 0.9944, and that of NewReno is 0.9952. Therefore, fairness of CRB is comparable to that of NewReno.

We ran simulations with two TCP connections of various protocols sharing the same bottleneck. Recall that BE may overestimate the fair share under congestion. The throughput results are shown in Figure 12(a) and Figure 12(b). As we can see from the Figures, among the three combinations, namely, BE/NewReno, CRB/NewReno and NewReno/NewReno, the BE scheme achieves the highest throughput and the combination of BE/NewReno achieves the highest total utilization of the bottleneck capacity.

![Figure 12. Friendliness and utilization comparison](image)

![Figure 13. Throughput ratio vs. bottleneck buffer Size](image)

However, BE appears to be unfriendly to NewReno. It deteriorates NewReno’s performance as shown in Figure 12(a). Meanwhile, CRB shows almost no effect on NewReno, being quite comparable to the NewReno/NewReno combination. This indicates that CRB only takes the unused bandwidth and does not steal bandwidth from NewReno. Under larger error rate, as shown in Figure 12(b), both BE and CRB hardly damage NewReno. The reason is that NewReno can only occupy very small fraction of bottleneck capacity because of its inefficiency in presence of random errors. Thus both BE and CRB are able to pick up the unused bandwidth without hurting a coexisting NewReno.

We also investigated the impact of buffer space on friendliness. We use a configuration with a pipe size of 64 packets. We can see from Figure 13, when the buffer size is much smaller than the pipe size, BE is unfriendly to NewReno; and the throughput ratio reaches about 2:1. In the small buffer case, buffer overflows happen frequently, both NewReno and BE cannot grow their windows to fill the pipe before another buffer overflow. NewReno, upon packet loss indicated by three duplicate ACKs, reduces its window by one half, which would be smaller than half the pipe size, while BE keeps its window higher corresponding to the available bandwidth estimation. RE and CRB, on the other hand, show much more friendliness to NewReno in the small buffer case.

Now consider the case where the buffer size is much larger than the pipe size. Here, NewReno gets more than its fair share. In the large buffer case, reducing the window by half is not enough to clear the buffer, and relieve congestion. On the other hand, our bandwidth aware schemes reduce the cwnd and ssthresh by more than a half, helping to clear the buffer backlog faster and provide better performance.

From the above experiments we conclude that BE provides high efficiency and RE provides more friendliness. CRB incorporates both schemes in order to achieve both high utilization and friendliness to NewReno. To better
understand and visualize how BE and CRB interact with TCP NewReno, we introduce the following Efficiency/Friendliness Tradeoff Graph.

### 4.2.1. Efficiency/Friendliness Tradeoff Graph

To better visualize the efficiency/friendliness tradeoff behavior of our sampling schemes, we introduce the Efficiency/Friendliness Tradeoff Graph. Basically we intend to study how the total link utilization and TCP NewReno’s throughput are impacted by the introduction of TCPW. The following two experiments are carried out to produce the graph:

1. A simulation with $N$ flows of TCP NewReno transmitting through the bottleneck is run as the base case. The throughput of each flow, and thus the total utilization, are measured.
2. Another simulation is then run with half of the flows replaced with either TCPW BE or TCPW CRB. The new throughput and utilization are measured.

Let $t_{R1}$ be the average throughput of the flows in the first simulation, and $U_1$ be the total link utilization. Similarly, let $t_{R2}$ be the average throughput of the NewReno flows in the second simulation, and $U_2$ be the total link utilization (TCPW + NewReno). We define:

\[
\text{Efficiency Improvement} \quad E = \frac{U_2}{U_1},
\]

\[
\text{Friendliness} \quad F = \frac{t_{R2}}{t_{R1}}.
\]

Notice that in the above equation we use $t_{R2}$, which is the average throughput of half of the NewReno flows in simulation 1. In fact, even if the same TCP variant is used, each flow can achieve different throughput and thus half of the group could “hurt” the other half: by using half of the flows as the base case, we normalize the ratio to 1 if the “new” protocol is in fact the same as the old one. Also, notice that $t_{R1}$ depends on how we group the flows into two halves and which half do we pick. Ideally, we should calculate all the possible ways to pick a half, calculate $F$, and then take the average. In our experiments, however, we take the average of the best and worst case scenarios. We find the best and worst case by sorting the list of throughputs in ascending order, divide them into two halves, top and bottom. We then have the two extreme cases if we use either one of the averages as $t_{R1}$. If we name the average throughput of the two extreme cases as $t_{R1\text{-worst}}$ and $t_{R1\text{-best}}$, we finally obtain the formula used in our results:

\[
\text{Friendliness} \quad F = \frac{t_{R2}}{t_{R1\text{-worst}}} + \frac{t_{R2}}{t_{R1\text{-best}}} / 2.
\]

For each network scenario, we compute both $E$ and $F$. We can then place the point $(F, E)$ on the Efficiency / Friendliness Tradeoff graph shown in figure 14. As we can see, a point can fall in one of four regions. In the “Negative Impact Region,” both $E$ and $F$ are less than 1, which is undesirable (the new protocol has both decreased total utilization and hurt the other protocol). On the contrary, in the “Unlikely Region,” both $E$ and $F$ are greater than one, which would be the optimum because the new protocol would help the efficiency of everyone, but since we are dealing with a tradeoff, this is an unlikely case. In most cases, in fact, the points fall in the two Tradeoff Areas 1 and 2, in which an increase in efficiency $E$ (respectively, fairness $F$) is compensated for by a decrease in fairness $F$ (efficiency $E$). With TCPW, we expect to see the points in the region where $E > 1$ and $F < 1$.

![Figure 14. Efficiency/Friendliness Tradeoff Graph](image)

It is obvious that the “target” points in the graph are anywhere on the line $F = 1$ (we are not hurting the other protocol), with $E > 1$ (but we increase total utilization), with $E$ as large as possible. With TCPW BE, we expect to have a higher $E$ and lower $F$, i.e. in the middle of the Tradeoff Region 1. TCPW CRB should reside in an area much closer to $F = 1$, because we expect it to be more friendly than BE. This is indeed confirmed in our experiments as we discuss below.

![Figure 15. Efficiency/Friendliness Tradeoff Graph results](image)

Two network scenarios in which TCPW has the most advantage (and thus drawing the most concern on friendliness) are simulated. The first one is a large leaky pipe (bottleneck bandwidth = 45Mbps, round trip propagation time = 74ms, random loss rate = 0.1%). The second one is wired/wireless configuration, (wireless bottleneck bandwidth = 11Mbps, random trip propagation time = 74ms, random loss rate = 2%). Both experiments show that TCPW CRB is friendly, yet provides relatively high efficiency.

The Efficiency/Friendliness Tradeoff Graphs are shown in Figure 15 (a) and (b). For each experiments we show the points for different number of total competing flows ($N$). When $N$ is small (e.g. 2, 4), most of the link is not utilized and the introduction of TCPW increases efficiency. With TCPW BE, however, NewReno experiences performance deterioration. On the other hand, TCPW CRB achieves less efficiency without much effecting on TCP NewReno. When $N$ is large (e.g. 20) the NewReno flows collectively almost fully utilize the link by themselves. TCPW BE and CRB have no room to improve total efficiency. Note however that TCPW CRB in this case has less detrimental impact on NewReno.

From the graphs, the point $N = 10$ for TCPW CRB is the farthest away from $F = 1$. It appears that this is due to the fact that neither error nor buffer overflow is a dominant cause of loss. A possible approach to handle this issue is a proper tuning of the threshold $\theta$ the algorithm uses to choose between BE and RE (in the experiments $\theta = 1.4$ was used). Nonetheless, from the graph above we can see that CRB provides great improvement on friendliness and achieved its

![Image](image)
goal: to utilize the link as much as possible with little affect on others.

5. Conclusion and future work

This work has been motivated by the need to make TCPW friendly to legacy protocols such as NewReno. We have shown that friendliness as well as throughput efficiency is impacted by the choice of bandwidth estimation in TCPW. We have introduced two estimators, BE and RE and have impacted by the choice of bandwidth estimation in TCPW. We have shown that friendliness is friendly to legacy protocols such as NewReno. We have shown that friendliness is also impacted by the choice of bandwidth estimation in TCPW. We have shown that friendliness is friendly to legacy protocols such as NewReno. We have shown that friendliness is friendly to legacy protocols such as NewReno.

Another important behavior uncovered by our study is the impact of buffer space on relative utilizations. When buffer space is large, NewReno appears to get more than its fair share. With small buffers, it is TCPW that gets more than its fair share. Robustness to buffer space is an important and desirable attribute of TCP protocols. AQM schemes seem to help reduce the impact of buffer size variation in TCPW; however, we are yet to evaluate this in detail.

The experimental results we presented in this paper provide preliminary insight into friendliness issues. Our work is continuing in order to study in more detail the impacts of buffer size, buffer management techniques (e.g. AQM), error rates, and link speeds on friendliness. Further, controlling the degree of friendliness through the parameter $\theta$ may be worth investigating. This also points to the possibility of devising continuously adaptive sampling methods instead of the binary selection among RE and BE presented in this paper.

References


