End-to-end differentiation of congestion and wireless losses

Song Cen, Pamela C. Cosman, and Geoffrey M. Voelker

Abstract— **In this paper, we explore end-to-end loss differentiation algorithms (LDAs) for use with congestion-sensitive video transport protocols for networks with either backbone or last-hop wireless links. As our basic video transport protocol, we use UDP in conjunction with a congestion control mechanism extended with an LDA. For congestion control, we use the TCP-Friendly Rate Control (TFRC) algorithm. We extend TFRC to use an LDA when a connection uses at least one wireless link in the path between the sender and receiver. We then evaluate various LDAs under different wireless network topologies, competing traffic, and fairness scenarios to determine their effectiveness. In addition to evaluating LDAs derived from previous work, we also propose and evaluate a new LDA, ZigZag, and a hybrid LDA, ZBS, that selects among base LDAs depending upon observed network conditions.**

We evaluate these LDAs via simulation, and find that no single base algorithm performs well across all topologies and competition. However, the hybrid algorithm performs well across topologies and competition, and in some cases exceeds the performance of the best base LDA for a given scenario. All of the LDAs are reasonably fair when competing with TCP, and their fairness among flows using the same LDA depends on the network topology. In general, ZigZag and the hybrid algorithm are the fairest among all LDAs.

Keywords— **wireless loss, loss differentiation, congestion control, TCP friendly rate control, video transport protocol**

I. Introduction

In this paper, we explore end-to-end loss differentiation algorithms (LDAs) for use with congestion-sensitive video transport protocols for networks with either backbone or last-hop wireless links. Video transport protocols can take advantage of loss differentiation in two key ways. The first is the well-known performance optimization where only congestion losses are used as congestion signals, and wireless losses do not restrict the sending rate [1], [2], [3]. The second is to provide useful feedback to the video encoder. For example, if wireless losses are dominating, the encoder can adjust the balance between bits devoted to source coding (representing the video) and bits devoted to channel coding (protecting the source coded bits). The focus of our initial work and this paper is on exploring and evaluating end-to-end LDAs for improving transport protocol performance.

As our basic video transport protocol, we use UDP in conjunction with a congestion control mechanism extended with an LDA. For congestion control, we use the TCP-Friendly Rate Control (TFRC) algorithm [4]. TFRC is an equation-based congestion control algorithm explicitly designed for best-effort unicast multimedia traffic. TFRC estimates the recent loss event rate of a connection at the receiver. The receiver communicates this loss rate back to the sender, which adapts its transmission rate to the degree of congestion estimated from the loss rate. To behave in a TCP-friendly manner, the sender adapts according to an equation that models the TCP response function in steady-state — but does so with significantly less fluctuation in the sending rate than the standard TCP congestion control algorithm. As a result, streaming applications can both smoothly and fairly react to congestion over longer time periods.

We extend TFRC to use an LDA when a connection uses at least one wireless link in the path between the sender and receiver. When a TFRC receiver detects losses, it invokes the LDA. If the LDA classifies the loss as a congestion loss, then the TFRC receiver includes it in its calculation of the loss event rate. However, if the LDA classifies it as a wireless loss, then the TFRC receiver does not count it in the loss event rate. Note that, either way, a lost packet is not retransmitted.

One goal of this paper is to evaluate LDAs under more realistic situations. Previous end-to-end approaches for loss differentiation [5], [6] were only evaluated under constrained conditions: a single wireless network topology, or without any competing traffic. As a result, we do not know how LDAs behave under the more realistic situations of varied wireless network topologies and competing traffic. We evaluate two LDAs derived from previous work. The first is based upon an algorithm proposed by Biaz et al. [5] that uses packet inter-arrival times to differentiate losses. The second is derived from Tobe et al. [7] and uses relative one-way trip times (ROTT).

A second goal of this paper is to propose and evaluate a new LDA, ZigZag, as well as a hybrid algorithm, ZBS, that switches among base LDAs depending upon observed network conditions. The goal of the two new LDAs is to achieve high throughput with low congestion losses. To distinguish losses, ZigZag uses ROTT as a function of loss count. The insight behind ZigZag is that ROTT combined with loss count is more insensitive to topology and competition as it exploits the characteristics of the multiplicative decrease linear increase (MDLI) congestion control algorithm used by TFRC. And since particular LDAs are well-suited to particular network conditions, the motivation behind the hybrid ZBS algorithm is to dynamically switch among base LDA algorithms according to observed network conditions.

To achieve these goals, we evaluate these algorithms via simulation using *ns* [15]. We study the performance and differentiation accuracy of the LDAs under two main wireless network topologies, networks with last-hop wireless links and networks with wireless backbones; the wireless last-hop topology corresponds to cellular networks or satellite modems, and the wireless backbone topology corresponds to high-bandwidth backbones or wireless LAN networks such as 802.11. We then study the LDAs under various scenarios of competing traffic where multiple flows use the same LDA. We further evaluate the hybrid LDA that combines the individual strengths of the base algorithms.

Finally, we evaluate the fairness and TCP-friendliness of the LDAs. Since an LDA cannot differentiate losses perfectly, it can obscure the congestion loss signal for TFRC and cause it to deviate from the standard TCP congestion control algorithm used for fairness on the Internet. To evaluate fairness, we mea-

sure the standard deviation of throughput among flows using the same LDA, and have each of them compete with standard TCP Reno that is free from any wireless loss.

Based upon our simulation results, we find that no single base algorithm performs well across all topologies and competition. At a high level, though, we find that LDAs based upon packet inter-arrival times do not behave well when there is competition for the bottleneck wireless link, and are only suitable for a particular topology and no competition. The LDAs based upon ROTT, however, are able to correlate congestion with particular losses much more accurately across a wide range of scenarios, although they may have relatively high wireless misclassification rates in particular situations. Finally, the ZBS hybrid algorithm performs well on both throughput and fairness by leveraging the strengths of the base LDAs.

The rest of this paper is organized as follows. Section II discusses related work. Section III describes previous algorithms for distinguishing between wireless and congestion losses, and introduces ZigZag, a novel algorithm for distinguishing losses that is TCP-friendly and relatively robust across different wireless topologies and competing traffic. Sections IV and V discuss the performance metrics and network parameters used in our simulation and evaluation of the LDAs. Sections VI, VII and VIII describe the simulation results in terms of throughput, network topology and traffic competition, and fairness and TCP-friendliness. Section IX discusses the computational complexity and other implementation issues of the LDAs. Finally, Section X summarizes and concludes.

II. Related Work

There has been considerable work characterizing the benefits of differentiating wireless losses from congestion losses for TCP connections, and developing various techniques for preventing TCP from reacting to wireless losses as if they indicated congestion. Examples of these techniques include splitting TCP connections at the base station [1], [3], and local retransmissions based on snooping at the wireless base station [2]. Balakrishnan et al. [9] evaluated a variety of these techniques, demonstrating that they can substantially improve TCP throughput and goodput.

However, most of these schemes assume a network where the wireless link is the last hop, and changes can be made at the wireless base station to accommodate the scheme. Furthermore, many of these schemes make wireless losses transparent to the sender, eliminating the opportunity for the sender to explicitly react at the application level to wireless losses (e.g., to trade off source and channel coding). Since we are interested in best-effort transport protocols, more general topologies, and networks where changes cannot be made to intermediate nodes, we have focused on end-to-end algorithms for differentiating and reacting to congestion and wireless losses.

There have been a few studies that have looked at this problem for TCP. Samaraweera proposed an end-to-end non-congestion packet loss detection (NCPLD) algorithm for a TCP connection in a network with a wireless backbone link, such as a low-bandwidth satellite link [6]. NCPLD measures round-trip time at the sender and compares it to the measured delay when there is no congestion to decide whether a loss is a wireless or congestion loss. Samaraweera simulates the algorithm and shows that, when a connection experiences congestion, NCPLD behaves as well as TCP when the wireless error rate is low, and improves throughput over TCP when the error rate is high. However, NCPLD was only evaluated for a wireless backbone topology.

Casetti et al. proposed an end-to-end modification of the TCP congestion window algorithm, called TCP Westwood [10]. TCP Westwood relies on end-to-end bandwidth estimation to discriminate the cause of packet loss. It continuously measures the rate of the connection at the TCP source by monitoring the rate of returning ACKs. The estimate is then used to compute the congestion window and slow start thresholds after a congestion episode. Through simulation and lab implementation, they show that TCP Westwood improves upon the performance of TCP Reno in wired as well as wireless networks, and the improvement is most significant in networks with mixed wired and wireless links. However, most of their evaluations are based on the wireless link being the last link to the receiver. This algorithm is also highly dependent on the TCP ACKing scheme, i.e., at least one ACK for every two packets received, which often does not exist in a best-effort transport protocol, e.g., TFRC.

Biaz and Vaidya have looked at two different approaches to end-to-end loss differentiation for TCP connections. They first looked at a set of "loss predictors" based upon three different analytic approaches to congestion avoidance that explicitly model connection throughput and/or round-trip time (e.g., TCP Vegas) [11]. Their results were negative in that these algorithms, formulated to do loss differentiation, were poor predictors of wireless loss. In subsequent work, they proposed a new algorithm that uses packet inter-arrival time to differentiate losses. Using simulation, they show that it works very well in a network where the last hop is wireless and is the bottleneck link [5]. However, they only evaluated their algorithm when a single flow was using the network in isolation. This algorithm, and a slightly modified version, are two of the algorithms that we evaluate in this paper in more general conditions (Section III-A).

Tobe et al. propose a rate control algorithm for UDP flows that uses spikes in relative one-way trip time (ROTT) as a congestion signaling mechanism [7]. They find that sequences of these spikes, or spike-trains, are only related to congestion-related losses and are not related to random losses exemplified by wireless losses. They use these spike-trains to classify paths, allowing for the use of different congestion control mechanisms on different paths. But they do not use it to differentiate the cause of each packet loss. In this paper we describe a version of this algorithm (Section III-B) designed to explicitly differentiate between congestion and wireless losses, and we evaluate its performance.

III. Base Algorithms

The three basic LDAs with which we experimented are called Biaz, Spike, and ZigZag, and they are described in this section. The hybrid scheme we evaluated is based on these three fundamental schemes, and is introduced in Section VI. In the following, we use the term original TFRC or unaware TFRC to refer to the original TFRC algorithm which is unaware of wireless loss, and treats every loss as due to congestion. We use the term omniscient TFRC to refer to an ideal TFRC implementation that

has precise knowledge of the cause of every packet loss.

A. Biaz scheme

The Biaz scheme [5] uses packet inter-arrival time to differentiate between loss types. As depicted in Figure 1, the algorithm works as follows. Let T_{min} denote the minimum packet inter-arrival time observed so far by the receiver during the connection. Let P_i denote the last in-sequence packet received by the receiver before a loss happened. Let P_{i+n+1} denote the first out-of-order packet received after the loss, where n is the number of packets lost. Let T_i denote the time between the arrivals of packets P_i and P_{i+n+1}. Finally, assume all packets are of the same size. If $(n+1)T_{min} \leq T_i < (n+2)T_{min}$, then the n missing packets are assumed to be lost due to wireless transmission errors. Otherwise, they are assumed to be lost due to congestion.

The concept here is that based on the arrival time of P_i, if P_{i+n+1} arrives right around the time that it should have arrived, we can assume the missing packets were properly transmitted and lost to wireless errors. If P_{i+n+1} arrives much earlier than it should, then at least some packets ahead of it (P_{i+1} ... P_{i+n}) probably were dropped at a buffer, and if it arrives much later than expected, then it is likely that queuing times at buffers have increased. Either way, we can attribute the loss to congestion. The Biaz scheme works best when the last link is both the wireless link and the bottleneck link of the connection, and is not shared by other connections competing for the link.

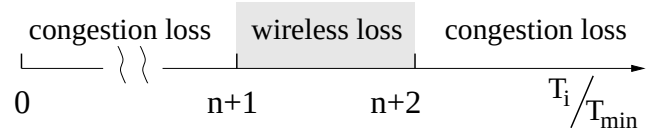


Fig. 1. Biaz Scheme. Here n is the number of consecutive packet(s) lost; T_i is the instantaneous packet inter-arrival time of the first packet received after the loss; T_{min} is the minimum packet inter-arrival time observed so far.

mBiaz: We found experimentally that the Biaz scheme often has high congestion loss in the wireless last hop topology (8–12% of throughput), almost twice as much as the omniscient TFRC traffic would cause. All other basic schemes have lower congestion loss than that of omniscient traffic, as will be seen in Section VI. This is mainly because Biaz misclassifies a significant number of congestion losses as wireless losses which prevents the sending rate of a flow from being reduced when the network is over crowded. In this section, we propose a modified version of Biaz, which we call mBiaz, that results in lower congestion loss than the original. We do this by adjusting the thresholds as follows.

Examining the thresholds used in the Biaz scheme more closely, we see that the lower threshold $(n+1) \times T_{min}$ would often be attained if in fact the wireless link is the last link with the lowest bandwidth and is not shared. This is because T_{min} equals the time to transmit the smallest packet over the wireless link, and when n packets were lost due to wireless error, the time it takes to transmit those n packets plus the next correctly received packet is at least $(n+1) \times T_{min}$. It equals $(n+1) \times T_{min}$ when all $n+1$ packets are buffered one after the other at the wireless link, and packets are of the same size. For T_i to be smaller than $(n+1) \times T_{min}$ in this case of n packets lost to wireless error, the average size of the lost packets must be smaller than the smallest packet received so far, which becomes more rare as the length of the connection gets longer. It does not occur in our experiments since all packets are of the same size.

On the other hand, the upper limit $(n+2) \times T_{min}$ provides a cushion window for the algorithm as the utilization of the last wireless link can not be 100% at all times. Whenever the wireless link is not 100% utilized, the packet inter-arrival time is greater than T_{min}. After a wireless loss of n packets, the expected arrival time of P_{i+n+1} after P_i would be greater than $(n+1) * T_{min}$. With the cushion provided by the upper window, the algorithm could still classify the loss correctly. Since the packet inter arrival time is directly related to the utilization of the wireless link, the window's upper limit should be related to it also: *the more the wireless link is close to fully utilized, the lower the upper limit should be.*

A very high upper limit is not appropriate because congestion loss accuracy would be sacrificed. The higher the upper limit, the more likely a loss will be classified as a wireless loss, i.e., the scheme trades off higher accuracy for classifying wireless loss with lower accuracy for congestion loss. Since the sending rate is not reduced when a loss is classified as a wireless loss, a higher upper limit potentially causes higher congestion and unfairness. The high congestion loss observed with the Biaz scheme indicates that the upper window limit of $(n+2) \times T_{min}$ is probably too high.

We want to find a reasonable value for the upper limit given the assumption that the wireless link has the lowest bandwidth. There are many reasons for the lowest bandwidth wireless link to not be 100% utilized: competition somewhere else in the network can limit the average utilization of the wireless link (see Section VI-C); even when the wireless link is the true bottleneck of the path, TCP and TFRC both have to probe the available bandwidth and generally are not able to maintain constant sending rate equal to the bottleneck link bandwidth.

To determine the value of the upper window limit, we tested two cases where (a) the wireless link is the true bottleneck link and is about 100% utilized, and (b) the average utilization of the wireless link is 86%. We consider that these utilization rates reasonably represent the two ends of possible scenarios, as the wireless link with the smallest bandwidth is unlikely to be much less utilized than this. The upper window limit that works well in both cases should also work well when the average utilization falls in between. Our experimental results with the upper window limit ranging from $[(n+1.1) \sim (n+1.8)] * T_{min}$ indicate that $[(n+1.2) \sim (n+1.3)] * T_{min}$ provides a good tradeoff between low congestion loss misclassification and high throughput in the wireless last hop topology (see Section V-A). The Biaz scheme's performance is insensitive to the choice of upper limit in the wireless backbone topology. Therefore, we choose $(n+1.25) * T_{min}$ in the modified Biaz scheme (Figure 2).

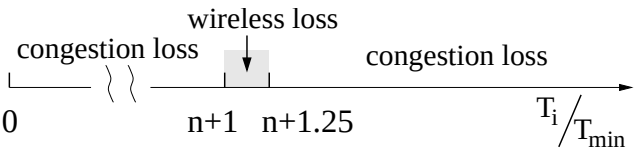


Fig. 2. Modified Biaz Scheme

B. Spike scheme

The Spike scheme was derived from [7], which differentiated among degrees of congestion but did not explicitly differentiate wireless loss from congestion loss. The Relative One-way Trip Time (ROTT) is a measure of the time a packet takes to travel from the sender to the receiver. Since the sending and receiving times are measured at the sender and receiver separately, the absolute value of delay is difficult to obtain due to the clock skew between the two, thus the name "relative." The ROTT is used to identify the state of the current connection. If the connection is in the *spike state*, losses are assumed to be due to congestion; otherwise, losses are assumed to be wireless. The spike state derives its name from the fact that plots of ROTT vs. time tend to show spikes during periods of congestion.

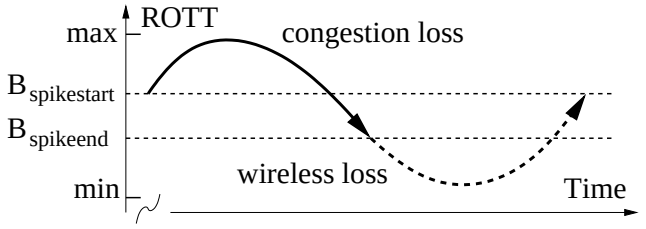


Fig. 3. Spike Scheme

The spike state is determined as follows. On receipt of a packet with sequence number i, if the connection is currently not in the spike state, and the ROTT for packet i exceeds the threshold $B_{spikestart}$, then the algorithm enters the spike state. Otherwise, if the connection is currently in the spike state, and the ROTT for packet i is less than a second threshold $B_{spikeend}$, the algorithm leaves the spike state. When the receiver detects a loss because of a gap in the sequence number of received packets, it classifies the loss based on the current state (see Figure 3).

In [7], the threshold values $B_{spikestart}$ and $B_{spikeend}$ were hard-coded to be $(rott_{min} + 20ms)$ and $(rott_{min} + 5ms)$, respectively. For a connection that rarely experiences extra delays (compared to the minimum) lower than 5ms or higher than 20ms, however, these thresholds will make the algorithm minimally useful. Instead, these thresholds should depend on the overall network delays. Therefore, we formulate the thresholds as follows: $B_{spikestart} = rott_{min} + \alpha * (rott_{max} - rott_{min})$

$$B_{spikeend} = rott_{min} + \beta * (rott_{max} - rott_{min})$$

where $rott_{max}$ and $rott_{min}$ are the maximum and minimum relative one-way trip time observed so far, and $\alpha \geq \beta$.

To use these formulas, we need to determine values for the parameters α and β. Suppose we consider all the buffers along the route from the sender to the receiver as one big buffer. The $rott_{min}$ occurs when that buffer is empty, and $rott_{max}$ occurs when that buffer is full. Setting $B_{spikestart}$ as above corresponds to the buffer being filled at level α, and $B_{spikeend}$ corresponds to the buffer being filled at level β. With a fixed distance of $d = \alpha - \beta$, a higher position of α and β means it is more likely that loss would be classified as wireless loss, resulting in higher congestion loss misclassification and lower wireless loss misclassification. If $\alpha \geq 1$, congestion loss misclassification is 100% while wireless loss misclassification is 0%; if $\beta \leq 0$, then the misclassification of congestion loss is 0% and wireless is 100%. The distance d between α and β determines the stability of the spike and non-spike states. Small d makes the algorithm oscillate between the two states easily, while large d makes both states more stable. To explore the sensitivity of the performance of the Spike scheme to these parameters, we conducted tests with β ranging from $[0.05, 0.5]$, and the distance of $(\alpha - \beta)$ ranging over $[0, 0.9]$, and found $\alpha = 1/2$ and $\beta = 1/3$ results in a good tradeoff of low congestion loss misclassification and reasonable wireless loss misclassification in the wireless last hop topology (see Section V-A). The Spike scheme's performance in the wireless backbone topology is relatively insensitive to the choice of α and β.

C. ZigZag scheme

In addition to the above schemes derived from previous work, we propose a new scheme called ZigZag. Using the same notation as in the Biaz scheme, ZigZag classifies losses as wireless based on the number of losses, n, and on the difference between $rott_i$ and its mean ($rott_{mean}$). A loss is classified as wireless if

$$\begin{aligned} &(n = 1 \text{ AND } rott_i < rott_{mean} - rott_{dev}) \\ \text{OR } &(n = 2 \text{ AND } rott_i < rott_{mean} - rott_{dev}/2) \\ \text{OR } &(n = 3 \text{ AND } rott_i < rott_{mean}) \\ \text{OR } &(n > 3 \text{ AND } rott_i < rott_{mean} - rott_{dev}/2) \end{aligned}$$

Otherwise the loss is classified as congestion loss.

Figure 4 illustrates this classification boundary. The mean ROTT $rott_{mean}$ and its deviation $rott_{dev}$ are calculated using the exponential average with $\alpha = 1/32$:

$$rott_{mean} = (1 - \alpha) * rott_{mean} + \alpha * rott$$

$$rott_{dev} = (1 - 2\alpha) * rott_{dev} + 2\alpha * |rott - rott_{mean}|$$

In this formula, $(1 - \alpha)$ is the exponential decaying factor that controls the smoothness of $rott_{mean}$ and $rott_{dev}$. We experimented with α of the form 2^N, where N varies among all integers from -2 to -8. Results show that $\alpha = 2^{-5} = 1/32$ provides the best results. We experimented only with powers of two for computational simplicity.

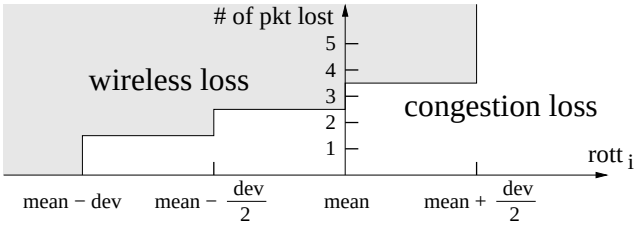


Fig. 4. ZigZag Scheme

By definition, ROTT has a high probability of having values greater than $(rott_{mean} - rott_{dev})$: 84% if it were a normalized Gaussian distributed random variable. As one packet loss is the most common loss pattern in a wired network, and congestion loss usually comes with higher delay, the threshold of $rott > rott_{mean} - rott_{dev}$ intuitively would classify most of the congestion loss correctly. The reasoning behind increasing the threshold with the number of losses encountered is that a more severe loss is associated with higher congestion, and with higher ROTT. This way, a loss event containing four or more packets would be classified as congestion loss only when relatively large ROTT were observed.

The insight behind this ROTT comparison is that with the multiplicative decrease and linear increase (MDLI) algorithm used in TCP/TFRC, the ROTT often exhibits a saw-tooth pattern: the instantaneous ROTT tends to be less than its mean after a multiplicative decrease action taken after congestion, and the

probability that the instantaneous ROTT is greater than its mean increases with the linear increase of window size. This pattern is characteristic of MDLI congestion control regardless of other network parameters. Therefore, as will be seen later, the misclassification rate of ZigZag is rather insensitive to changes in network topology.

IV. Performance Metrics

An algorithm that attempts to classify each loss into one of two classes can be judged by its misclassification rate, the fraction of cases which are classified incorrectly. Since misclassifying a wireless loss as a congestion loss does not have the same impact as the other way around, we can judge performance by examining the two separate misclassification rates. However, our ultimate concern is with the throughput of the traffic stream that results from using the algorithm, and with whether the algorithm causes severe congestion and thereby diminishes the throughput of other traffic streams. This leads us to a set of four performance measures.

Throughput: The most important goal is high throughput, where we are concerned with the improvement compared to the original TFRC (unaware of wireless losses) when transmitting through a network with a wireless link. Our experiments show that an omniscient TFRC connection can have a throughput 200% higher than an unaware TFRC connection, depending on the topology and wireless loss severity. A primary goal is to have a throughput close to that of omniscient TFRC.

Congestion Loss: The amount of congestion loss experienced by a TCP connection or other traffic when competing with traffic shaped by an LDA is affected by the behavior of the LDA. The throughput of the other connections should not be too much lower than without traffic using an LDA. For two LDA schemes with similar throughput, we would prefer the one which causes less congestion loss. Wireless loss is proportional to throughput, so it is not part of our performance measures.

Misclassification rates: We need to be conservative in misclassifying congestion loss as wireless loss, as such a mistake means rate will not be reduced when the network is congested. The congestion loss misclassification rate (M_c) of both the original TFRC and the omniscient TFRC is 0%. Misclassifying wireless loss (M_w) as congestion loss does not cause congestion problems for the network, but it often limits the protocol's ability to improve throughput. The M_w of the original TFRC is 100%, and for omniscient TFRC, 0%.

The relationships between throughput, congestion loss, M_c, and M_w are related to the actions taken for losses that were classified as wireless. Currently, we treat all lost packets classified as wireless error in the same way as received packets. Under such circumstances, a higher M_c means (a) higher congestion loss, (b) higher throughput when competing with different types of traffic — less friendly to those unaware of wireless loss, e.g., TCP and TFRC, and more aggressive when competing with omniscient, and (c) when competing with itself, lower throughput if M_c is too high.

On the other hand, higher M_w often means (a) lower congestion loss, (b) lower throughput when competing with different types of traffic — friendlier to TCP and TFRC, but less competitive with omniscient, and (c) when competing with itself, lower throughput if M_w is too high.

However, for an LDA that has both high M_c and high M_w, their effects can partially cancel. For example, the lower throughput that would have happened with high M_w may not be realized when there is similarly high M_c — as will be seen with the Spike scheme in Section VI-C. Thus the values of M_c and M_w should be considered together with the corresponding throughput and congestion loss. *From the standpoint of application and network requirements, the criteria for a good LDA are high throughput and low congestion loss.*

V. Network Parameters

In this section, we describe the topologies, wireless loss model, and other network parameters that we use in our simulations.

A. Topology

We tested the LDAs on three types of topologies which we call *Wireless Last Hop*, *Wireless Backbone*, and *Wireless LAN*.

Wireless Last Hop: In the Wireless Last Hop (WLH) topology (Figure 5), the last link to the receiver is a wireless link with bandwidth and delay of $rate_{wlast}$ and $delay_{wlast}$. N traffic streams share a common wired link with bandwidth and delay of $rate_{shared}$ and $delay_{shared}$. The $rate_{shared}$ is set to be 86% of the aggregated total of all wireless links' bandwidth when there is more than one flow in the network. So the $N(\geq 2)$ streams compete for bandwidth at the common link, and congestion can happen both at the wired shared link as well as at the wireless last link. This type of topology simulates a cellular network or satellite Direct-TV system, where each wireless link has a relatively constant bandwidth.

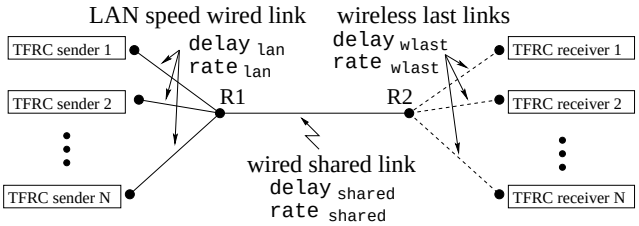


Fig. 5. Wireless Last Hop Topology

Wireless Backbone: In the Wireless Backbone (WB) topology (Figure 6), the shared link (backbone) between two LANs is a wireless link, with bandwidth and delay of $rate_{wshared}$ and $delay_{wshared}$. This topology simulates a scenario where LANs are connected by a high bandwidth wireless link.

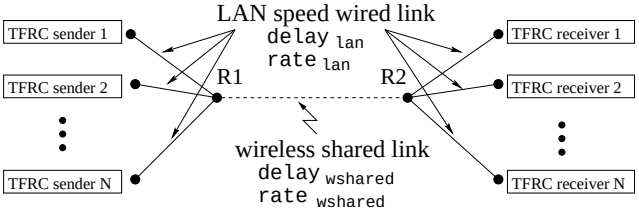


Fig. 6. Wireless Backbone Topology

Wireless LAN: In the Wireless LAN topology (Figure 7), the wireless link connects directly to multiple mobile receivers. This topology simulates an 802.11 wireless LAN. The only difference between this topology and the WB topology above is

the existence of the last link from router R2 to each individual receiver. As the bandwidth of the LAN speed links is typically much higher than that of the wireless shared link, there are no packets buffered at these links, so the only effect they have is additional delay. In our experiments, the wireless LAN shows essentially identical results as a WB topology when the corresponding link bandwidths are the same and the total fixed delay (processing + propagation) from sender to receiver is roughly equal between the topologies. Thus, in the following discussion, we only consider the WB topology, with its results directly applicable to the wireless LAN case.

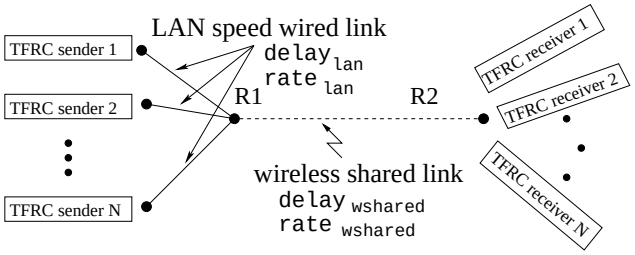


Fig. 7. Wireless LAN Topology

B. Wireless Loss Model

In our experiments, we use the Jakes model [12], [13] to simulate the loss patterns in the forward channel at the wireless links; for simplicity, we assume that wireless error only exists in the forward direction from sender to receiver, and that there is no wireless loss in the reverse direction. The Jakes model is a deterministic method for simulating a time-correlated Rayleigh fading channel. We generated the error pattern of the Jakes model via computer simulation as in [14]. Packets of size 381 bytes were transmitted for 12 seconds on a 150Kbps simulated wireless channel, and the receiver attempted to decode each packet and recorded whether it was corrupted by an uncorrectable wireless error. For a particular set of channel parameters, the results of 100 random trials, equivalent to 1200 seconds transmission, formed the error patterns used in our *ns* simulations. Other system parameters used in the error pattern simulations were: channel code rate: 1/2; number of concurrent users: 5; number of multi-paths resolved: 4; energy-per-bit/noise (E_b/N_0): 4dB; normalized Doppler $f_D T_c = 2.62 \times 10^{-4}$; and the three combinations of spreading gain and interleaver size given in Table I. These were chosen to represent high, medium and low wireless loss scenarios. Figure 8 shows the histogram of the good and error state length of the high wireless loss error pattern.

TABLE I
JAKES MODEL: HIGH, MEDIUM, AND LOW PACKET LOSS

Spreading Gain	Interleaver Size	Packet Loss Rate	Bit Error Rate
16	2 pkt	high: 7.8%	8.7×10^{-5}
16	3 pkt	medium: 3.1%	2.8×10^{-5}
32	2 pkt	low: 1.0%	7.1×10^{-6}

The Jakes model is a more accurate model for the wireless channel experienced by moving objects than the traditional two-state Markov error model. However, we also tested a simplified

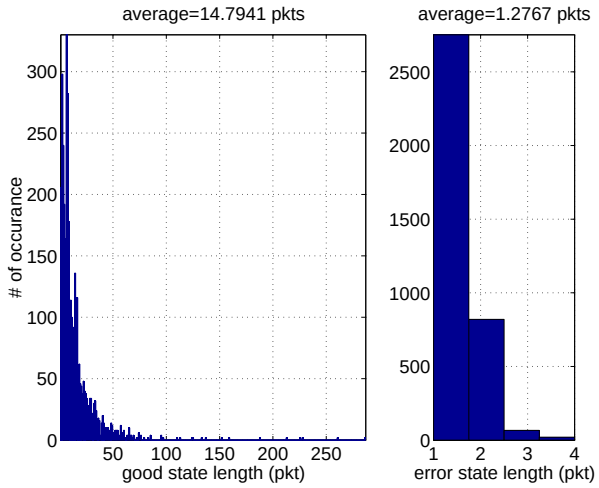


Fig. 8. Histogram of good/error state length (high loss)

version of the two-state Markov error model: the independent (Bernoulli) or "exponential" error model in which the time between successive errors is exponentially distributed [9] [10]. For each Jakes model with a particular set of parameters, we also tested a matched Bernoulli model which has roughly the same average packet loss rate, and the same distribution of the good and bad state lengths. Results from both error models matched very well with no discrepancies in terms of the relative performance of the LDAs. Therefore, we only include results from experiments using the Jakes model in this paper.

We note that fixed point high bandwidth radio links, such as those in the UCSD HPWREN [8] wireless backbone topology, often exhibit very long periods (days) of good states with packet loss rate well below 10^{-4} interspersed by occasional periods (minutes) of bad states where the wireless packet loss rate approaches 3%, which is the value we study in our medium loss scenario. For wireless backbones attached to a moving object, e.g., an airplane or a vehicle such as in a military application, the wireless loss pattern of such wireless backbones fits in the same model as that of the wireless last hop scenario.

C. Other Parameters

Bandwidth: As discussed later, we tested all schemes with N=1, 2, 4, 6, 8, 10, 12, and 16 traffic flows in the network.

The WLH topology simulates a cellular network, so we set $rate_{wlast} = 150$ Kbps, and $rate_{shared} = max(N, 2) * 130$ Kbps, i.e., 86% of the aggregated total bandwidth of the wireless links, except when there is only one traffic flow. With only one flow in the network, the capacity between routers R1 and R2 is set roughly twice the wireless link capacity so the wireless link is the bottleneck link.

For the WB topology, we set $rate_{wshared}$ = 800 Kbps for one flow, and 1600 Kbps otherwise. This way, average bandwidth for the single flow case is exactly the same as for two flows. When comparing the two, which represent isolation and competition, effects of average bandwidth difference are eliminated.

For all the LAN links, $rate_{lan} = 10$ Mbps.

Delay: Total delays in the network are composed of processing, propagation, transmission and queuing delays. The *(processing + propagation)* delay is set explicitly:

$$delay_{lan} = 1ms, \quad delay_{wlast} = 10ms;$$
$$delay_{shared} = delay_{wshared} = 20ms \sim 60ms.$$

The other two are determined implicitly by the choice of other parameters: bandwidth, queue size, etc. Results with different delays set on the common shared link ($delay_{shared}$ and $delay_{wshared}$) match very well with no discrepancies in terms of the relative performance of the LDAs. Thus, only results with the two delays set to 20ms are included in this paper.

Packet Size: The packet size is 762 bytes. For a video coder that encodes at the rate of 25 frames/sec, and a bit rate of 150Kbps, a frame on average would occupy $150K/25 = 6000bits = 750bytes$. 762 was chosen because it is twice 381 bytes, a specified packet size in the CDMA-2000 standard.

Queue Size: The size of a queue in a router usually scales with the capacity of the link it is connected to. The size of the queue measured in bits divided by the link bandwidth is the maximum queuing delay. We use a scale formula used in the simulation script from [4]:

$$queue\_size(pkts) = max(link\_bandwidth/60K, 6)$$

If all packets are 762 bytes, this leads to a maximum queuing delay of 100ms (if the link bandwidth $\geq$ 360Kbps) or higher (if the link bandwidth $<$ 360Kbps).

Queuing Policy: DropTail only.

Random Traffic: Similar to [5], we have two *ns Traffic/Expoo* agents warm up the network for 20 seconds before any TFRC or TCP traffic starts, and they stop within 2 seconds after TFRC or TCP starts.

Test Conditions: In all experiments, after the warm-up period, data was transmitted for about 200 seconds. For each differentiation scheme, experiments were performed with the same random seed that determines the starting order (within 2 seconds) of and the wireless error pattern experienced by each flow. With different random seeds, the same set of experiments was repeated 10 times, and results were averaged.

VI. Evaluation of Base Algorithms

In this section, we evaluate the performance of the base algorithms under a variety of experimental conditions. We begin by examining the performance of each algorithm in isolation, first on the wireless last-hop topology (Section VI-A) and then on the wireless backbone topology (Section VI-B). Finally, we evaluate the algorithms when other flows compete for network resources in both topologies (Sections VI-C and VI-D).

A. Wireless Last Hop

First, we want to understand the performance and behavior of each LDA in isolation. We start by evaluating the algorithms separately in the WLH topology using the metrics and simulation methodology described in Sections IV and V, and then study the algorithms in the WB topology in Section VI-B.

Table II shows the results of simulating one flow of each of the differentiation algorithms as well as TCP, TFRC, and omniscient TFRC on the WLH topology. The table shows the throughput, congestion loss rate, and misclassification rates for each type of flow as percentages. The throughput (thput) is normalized by the bandwidth of the bottleneck link; congestion (cong.) is the number of packets lost due to congestion divided by the throughput;

TABLE II
Performance for wireless last hop, 1 flow

	TCP	TFRC	Omni	Biaz	mBiaz	Spike	ZigZag
thput	55	84	99	99	99	99	98
cong.	0.8	0.2	2.3	2.3	2.3	0.4	0.3
M_c	0	0	0	0.0	0.0	0.0	0.0
M_w	100	100	0	6.3	6.6	58	66

M_c is the fraction of packets lost to congestion that are misclassified as wireless loss; and M_w is the corresponding measure for wireless loss. Unless stated otherwise, all results in this and subsequent sections are for the high wireless loss case. Trends for high wireless loss hold for low and medium loss as well: the relative order of their performance does not change, although the absolute differences between the algorithms tend to be smaller.

TCP and TFRC, which do not use an LDA, had comparatively low throughput. They react to wireless losses as congestion losses, unduly reducing their sending rate; TFRC had a higher rate than TCP because it does not react as drastically to loss. As expected, omniscient TFRC is able to get close to full utilization of the bottleneck link bandwidth.

All four LDAs almost fully utilize the bottleneck bandwidth and misclassified no congestion losses. The Biaz algorithms made few mistakes on wireless losses; these algorithms were designed for this kind of topology. Because of this, they have the same slightly higher congestion loss as the omniscient TFRC flow, while Spike and ZigZag have less congestion since they misclassify more wireless losses and therefore reduce their sending rate.

By definition, since there is only one buffer to fill up, the high M_w of the Spike algorithm indicates that half of the time the buffer of the wireless link is at least 1/3 full. However, here the high M_w does not hurt the throughput of the Spike flow because it only happens when the buffer is at least 1/3 full; with a non-empty buffer, the router always has packets to transmit on the link to maintain throughput.

ZigZag also has a high M_w, indicating that, as the ROTT oscillates around its mean, there is a high probability that the ROTT is larger than $(rott_{mean} - rott_{dev})$. As a result, ZigZag misclassifies many wireless losses. $M_c = 0$ for both Spike and ZigZag shows that the thresholds chosen to parameterize the algorithms are quite conservative.

Summary. From these results, we conclude that all of the LDAs perform well in isolation on this topology, achieving excellent throughput while reacting to congestion well. The Biaz algorithms are highly optimized for this particular situation, while Spike and ZigZag are more conservative in that they classify some wireless losses as congestion losses.

B. Wireless Backbone

Next, we want to understand the performance of the differentiation algorithms on the wireless backbone (WB) topology, and to see how performance changes as the topology changes.

Table III shows the results of simulating the algorithms on the WB topology described in Section V-A. At a high level, with only one flow the WB topology is very similar to the WLH topology since (1) the LAN link that follows the wireless back-

TABLE III
PERFORMANCE FOR WIRELESS BACKBONE, 1 FLOW

	TCP	TFRC	Omni	Biaz	mBiaz	Spike	ZigZag
thput	23	37	99	97	91	99	53
cong.	0.1	0.0	0.4	0.4	0.4	0.0	0.0
M_c	0	0	0	0.0	0.0	0.0	0.0
M_w	100	100	0	2.4	7.0	29	60

bone link can effectively be ignored since its bandwidth is much higher than that of the wireless backbone, and (2) there is no competition, so the flow has sole use of the wireless backbone in a manner similar to the wireless last hop link.

The main difference between the performance of the algorithms when the flows operate in isolation on the two topologies is the difference in bottleneck bandwidth: the wireless link in the WB topology is 800 Kbps, whereas the wireless link in the WLH topology is only 150 Kbps. As a result, the differences in performance are primarily due to this change in bandwidth more than topology; in subsequent experiments, we will see more of an influence of topology on performance.

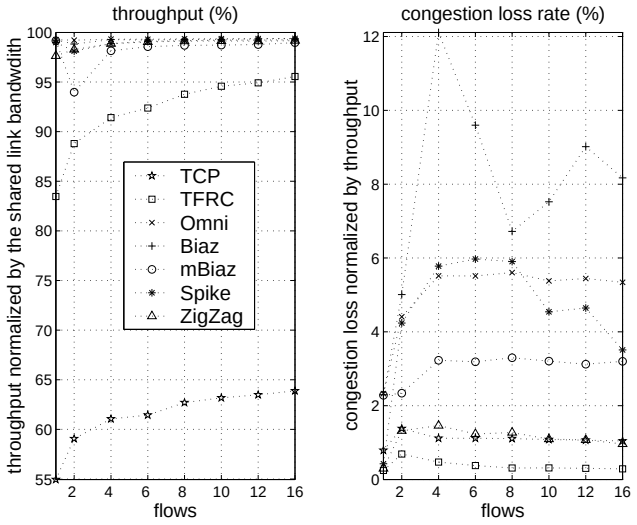
From Table III, we see that TCP and TFRC have a much lower usage of the available bandwidth when it is 800Kbps. This lower usage is due to the larger operating window size that comes with the higher bandwidth delay product, making the speed of the linear increase much slower than the speed of the multiplicative decrease caused by the high wireless loss. Omniscient TFRC still gets close to 100% utilization of the available bandwidth, but with much less congestion. This is also due to the higher operating window size, which makes the TFRC congestion control algorithm less likely to fall into the slow start mode and enables it to open its congestion window more smoothly in the linear increase phase.

The performance of the LDAs on the WB topology is for the most part similar to the WLH topology above. However, ZigZag has a much lower throughput that is similar to that of TCP and TFRC due to the larger window size at higher rates, and its high M_w. Unlike the Spike algorithm, which also has a relatively high M_w, the M_w in the ZigZag algorithm does not have any direct correlation with the buffer level (the ROTT can still oscillate around its mean even when the buffer is close to empty). For the same reason as TCP and original TFRC, it cannot recover the normal window size as quickly at the higher rate. The modified Biaz algorithm also has a lower throughput, although not as significant, due to its higher M_w and larger window size. Its higher M_w (compared to Biaz) results from a smaller window on average that allows less delay between packets when classifying loss as wireless.

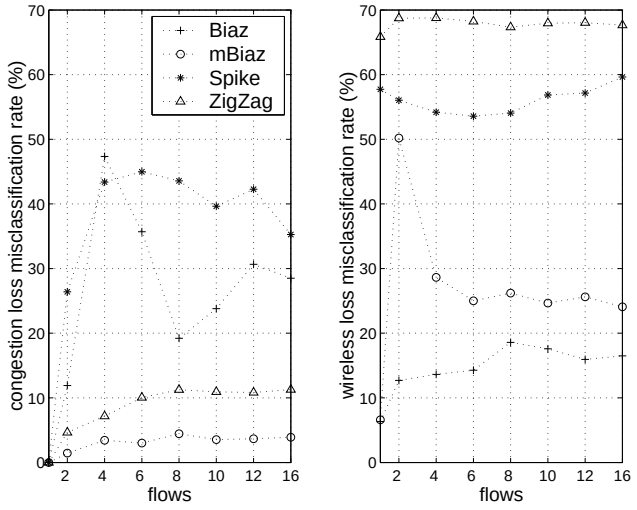
Summary. Since evaluating the LDAs in isolation on the WB topology essentially reduces to the WLH topology with a higher bandwidth wireless bottleneck link, the changes we see in performance are due to the change in bandwidth rather than topology. At the higher bottleneck bandwidth, TCP, TFRC, and ZigZag have even lower throughput due to their high wireless loss misclassification M_w; the other algorithms are able to maintain good throughput due to little or no M_w.

C. Competition in Wireless Last Hop

Now that we have evaluated the algorithms on both topologies in isolation, we next evaluate them with competing flows.



(a) throughput and congestion loss rate



(b) misclassification rates

Fig. 9. Competition with the wireless last hop topology

Figure 9 shows the performance of each algorithm on the WLH topology when there are one to 16 flows, all using the same algorithm; note that the single flow case corresponds to the results in Table II. Figure 9 has graphs to show throughput (top left), congestion loss (top right), M_c (bottom left), and M_w (bottom right). All graphs are a function of the number of flows competing on the network.

With more than one flow, the bottleneck link is the shared link whose bandwidth we purposely set to be 86% (130 Kbps/150 Kbps) of the aggregated sum of all wireless links to induce congestion. As a result, we show the throughput in the graph as the sum of all flows' throughput normalized by $rate_{shared}$. This throughput reflects the average throughput of the compet-

ing flows, so a high throughput means that on average the algorithm performs well when competing with itself. The misclassification rates and congestion loss are averages over all flows in the network as well. We know the misclassification rates for TCP, TFRC, and omniscient TFRC a priori, and therefore do not show them to improve clarity.

From Figure 9, we see that the average throughput of TCP and TFRC increases as the number of flows increases. The reason for this behavior is that not all flows will experience wireless error at the same time. As the number of flows increases, it is less likely wireless loss will be synchronized between different flows. The performance of omniscient TFRC is not affected by the change of flows.

Biaz maintains its high throughput regardless of the number of competing flows. However, its M_c increases dramatically as the number of flows goes beyond one because congestion losses at the shared bottleneck link become misclassified as wireless losses. This causes high congestion loss (7–12%) because Biaz does not scale back in the face of congestion when it should.

This problem with the Biaz algorithm motivated the modified Biaz scheme (Section III-A). Figure 9 shows that mBiaz addresses the problem of the original Biaz scheme in that it has the lowest M_c over all base algorithms. However, it now has the problem of a high M_w because, by using a lower upper window limit, it achieves high accuracy for congestion loss by trading off accuracy for wireless loss. However, the high M_w is also related to the choice of low utilization of the wireless link, which is disadvantageous to mBiaz. With more than one flow, the average utilization of the wireless link is only 86%, and so the packet inter-arrival time after a wireless loss of n packets is on average $(n+1) * 1.16 * T_{min} \geq (n+1.25)T_{min}$ — the upper window limit of mBiaz. So, the high M_w we see here will be reduced if the average utilization of the last wireless link is higher than 86%, which is likely to be true in a cellular network scenario. It is not wise to use a large classification window to accommodate connections temporarily starved with less than their fair share of the bandwidth because it also encourages connections that have high throughput to cause more congestion loss. As pointed out in Section III-A, the threshold of $(n+1.25)T_{min}$ provides a reasonable tradeoff between the accuracy of congestion loss and wireless loss in two extreme cases where utilization of the wireless link is about 100% (Section VI-A) and 86% (here).

The Spike scheme has consistently high throughput across all numbers of flows. However, it has high congestion loss, often higher than that of the omniscient TFRC, and both its M_c and M_w are very high. Its M_w is similar to the one flow case and persists in the face of competition. Its high M_c is due to its inability to correctly determine the buffer level at either the shared link or the wireless last link. Once a large ROTT is measured due to high buffer levels at both locations, it can no longer correctly gauge individual buffer levels. Congestion loss can occur with one of the buffers full and the other empty; the high ROTT measured previously will make the scheme miss congestion loss in such cases.

The ZigZag scheme has consistently high throughput and low congestion loss across all numbers of flows. Although it also is based on the idea that congestion loss accompanies high ROTT, unlike the Spike scheme, the exponentially averaged $rott_{mean}$ gradually forgets past history, making it immune to the occasional extreme value of ROTT observed. However, the wireless link buffer does cause higher M_c, especially as the number of flows increases. Nevertheless, it has the second lowest M_c and the variation is small compared to the other two base algorithms. Although its M_w is the highest among all base algorithms, at this operating rate, it does not affect the throughput.

Summary. All differentiation algorithms are able to achieve high throughput when competing with similar flows, although with a large variation in misclassification rates. With its consistently high throughput, low congestion loss, and low congestion misclassification rate M_c, ZigZag is the best performer under competition in the WLH topology.

D. Competition in Wireless Backbone

We now evaluate the algorithms when there is competition on the wireless backbone topology. Figure 10 shows the results of simulating the algorithms on the WB topology using the same graphs as in Figure 9. As Figure 10 shows, the performance of the algorithms when there is competition in the WB topology is quite different from the WLH topology. With more than one flow, there are two main differences between the two topologies that affect the performance of the algorithms:

1) The percentage of the shared link bandwidth that each flow can use (due to inherent characteristics of each topology):

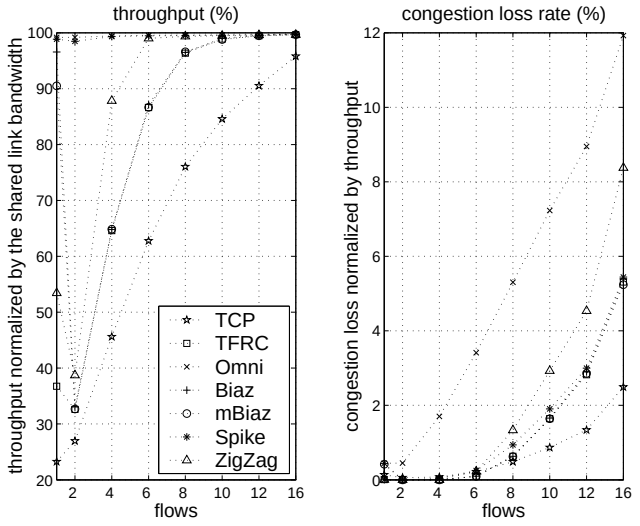
- In the WLH topology, the maximum receiving rate of any flow is bounded by the rate of the wireless last link, 150Kbps. Since the average bandwidth per flow is 130Kbps, no flow can get more than $150/130 = 115\%$ of its fair share in the common link bandwidth.
- In the WB topology, the receiving rate of a flow could potentially reach the capacity of the shared link; i.e., it can occupy the entire common link, reaching throughput that is N times its fair share, where N is the number of flows.

2) The average rate per flow (due to our choice of the network parameters):

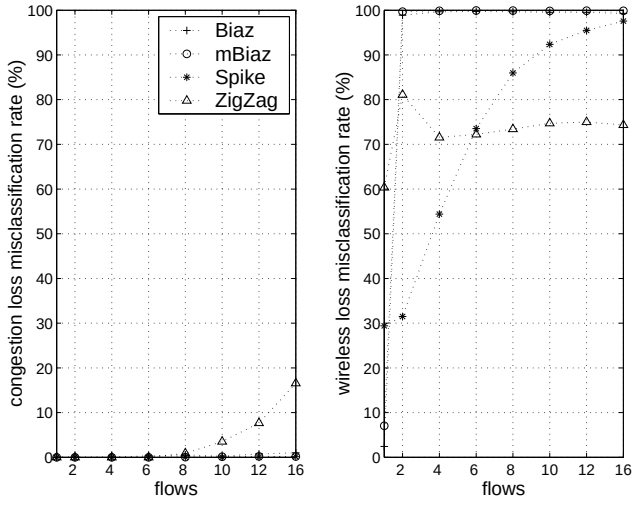
- The average rate per flow is fixed at 130Kbps for WLH
- In the WB topology, the average rate per flow is 1600/max(2,N) Kbps, i.e., in the range of 100 to 800 Kbps.

The quick and significant increase of TCP and TFRC throughput when the number of flows increases directly reflects both of these factors. On the one hand, as the de-synchronization effect of wireless error takes place, any flow that is temporarily not affected by wireless loss can increase its sending rate to potentially use all the unused bandwidth. On the other hand, as the average rate per flow decreases with increasing rate, TCP can get higher utilization of the bandwidth; with 10 flows, the average rate per flow is 160Kbps, and average utilization is 85% and 97% for TCP and TFRC respectively, while with only one flow at 150Kbps, their utilization is only 55% and 84% (see Table II). Omniscient TFRC can fully utilize the available bandwidth, but with much greater congestion loss. Since it is not affected by the wireless loss, it is mainly the average rate per flow that contributes to the variation on the graph of congestion loss vs. number of flows.

Both Biaz schemes have essentially 100% M_w for more than one flow because the wireless link is now shared. For Biaz to work accurately, packets from the same flow need to be buffered



(a) throughput and congestion loss rate



(b) misclassification rates

Fig. 10. Competition in the wireless backbone topology

one after the other at the wireless link. This situation is unlikely when there are two or more flows sharing the link, and they simply classified all losses as congestion losses — the same as the original unaware TFRC flow. As a result, the Biaz schemes are essentially useless as LDAs for this topology, and their throughput is the same as original TFRC.

The Spike scheme works well in this topology as buffer buildup can happen at only one place. Thus the Spike scheme accurately determines congestion loss (M_c close to 0). As the number of flows increases, the buffer level gets higher due to the de-synchronization effects of wireless loss. Therefore, its M_w, which is directly related to the average buffer level, increases accordingly. As described before, the increasing M_w does not affect its throughput performance.

The ZigZag scheme has similar M_w as in the WLH topology. Due to its high M_w, changes in ZigZag throughput follow the same pattern as TCP/TFRC flows. Figure 8 shows that about one quarter of wireless loss events involve two consecutive packets being lost. With two flows in the network, the probability that packets from both flows get hit by a wireless error near-simultaneously is relatively high. At the average rate of 800 Kbps per flow, as we have seen in Section VI-B, ZigZag is not able to return to the steady-state congestion window size quickly. However, the ZigZag scheme is able to fully use the available bandwidth when there are six or more flows. The reason for this is due partly to the de-synchronization effect of wireless errors, and partly because of lower average rate per flow ($\leq 1600/6 = 267$ Kbps). Finally, the M_c of ZigZag is mostly zero for 8 or fewer flows, where the average bandwidth per flow is ≥ 200 Kbps. At 10 or more flows, its M_c is lower than in the WLH topology, but the M_c is more costly in this environment, because the receiving rate of a flow could potentially reach the capacity of the shared link. Therefore, it has higher congestion loss than in the WLH topology for 10 or more flows.

Summary. The Spike scheme performs the best in this kind of topology since the change of ROTT directly comes from the buffer where congestion loss happens.

E. Summary

In summary, our evaluation of the base algorithms shows that:

• When there is only one flow in the network, the Biaz and Spike algorithms perform essentially the same on both topologies. ZigZag, however, is sensitive to the bottleneck link bandwidth due to its relatively high M_w: it performs well at the low link rates, but its throughput decreases significantly at higher link rates.

• When there is competition among flows in the WLH topology, ZigZag performs the best when the shared link bandwidth is less than or close to the total aggregated wireless link bandwidth. Modified Biaz also performs well when there is a large (≥ 4) number of flows. The original Biaz and Spike schemes both have an unacceptably high M_c and high congestion losses.

• When there is competition among flows in the WB topology, the best scheme is Spike. ZigZag is useful, although it suffers from its sensitivity to the average bandwidth per flow. Both Biaz schemes lose their differentiation ability and perform the same as TFRC.

Generally speaking, LDAs based upon packet inter-arrival times (Biaz and mBiaz) do not behave well when there is competition for the bottleneck wireless link, and as a result are only suitable for the WLH topology without competition on the wireless link. The LDAs based upon ROTT (Spike, ZigZag), however, are able to correlate congestion with particular losses much more accurately across a wide range of scenarios, although they may have relatively high M_w in particular situations.

We conclude that none of the base algorithms performs consistently very well across topologies and in the face of competition from other flows. This motivated us to explore a hybrid algorithm that can take advantage of the strengths of the individual base algorithms.

VII. Evaluation of a Hybrid Algorithm

In this section, we investigate a hybrid of the base algorithms. Since no single base algorithm performed well either across all

topologies or in the face of competition, we create a hybrid that dynamically uses different base algorithms depending on network characteristics. In the WLH topology, ZigZag and modified Biaz behave very well, while in the WB topology, Spike is the best performer and ZigZag performs reasonably well. Observing this behavior, can we design a switching algorithm that can select the right scheme for the right network conditions as observed under the different topologies? Looking at why Biaz failed in the WB topology provides some insight: the main difference between the two topologies is whether the wireless link with the lowest bandwidth is shared or not. Therefore, we should choose different schemes based on whether the lowest bandwidth wireless link is shared or not.

When the lowest bandwidth link is shared by N flows, the average packet inter-arrival time (T_{avg}) would be close to $N * T_{min}$, where T_{min} is the minimum inter-arrival time. If the slowest link is not shared, or $N = 1$, then T_{avg} should be close to T_{min}. We compute T_{avg} by exponential averaging:

$$T_{avg\_new} = 0.875 * T_{avg\_old} + 0.125 * \frac{inter\_arr\_time}{pkts}$$

Here $inter\_arr\_time$ is the instantaneous inter-arrival time (time between arrived packets) and we divide by the number of packets that separate the arrived packets; therefore, T_{avg} can be smaller than $inter\_arr\_time$ and in fact can take on a value even smaller than the minimum $inter\_arr\_time$, or T_{min}.

Let $T_{narr} = T_{avg}/T_{min}$. In the WLH topology, $T_{narr} \approx 1$; while in the WB topology, $T_{narr} \approx N$, where N is the number of flows sharing the link. However, when the connection starts up, the real T_{min} may not be observed immediately, thus T_{narr} could be < 1 at congestion loss. Also, there are certain ambiguities when the number of traffic flows on the wireless link increases from 1 to 2, because in both one and two flows, T_{narr} could often take on values between 1 and 2. In both cases, we cannot determine topology conditions with confidence. Our solution is to use ZigZag during these periods due to its relatively consistent performance whether or not there is competition for the shared bottleneck wireless link.

A. Hybrid Algorithm: ZBS

Based on this idea, we introduce a hybrid algorithm, ZBS, that dynamically uses one of the base algorithms according to current network conditions as follows:

if $(rott < (rott_{min} + 0.05 * T_{min}))$ use Spike;
else {
  if $(T_{narr} < 0.875)$ use ZigZag;
  else if $(T_{narr} < 1.5)$ use mBiaz;
  else if $(T_{narr} < 2.0)$ use ZigZag;
  else use Spike;
}

In the rest of this section, we explain the insight behind the use of the different algorithms and the derivation of the parameters used to decide among them. In the next section, we evaluate the performance of the hybrid algorithm.

Starting at the first line of the predicate, compared to one packet transmission time over the bottleneck link (T_{min}), it is very likely that the bottleneck link is empty or under-utilized when the relative one-way trip time is very close to its minimum. Both mBiaz and ZigZag do not perform well in this situation, so Spike is used.

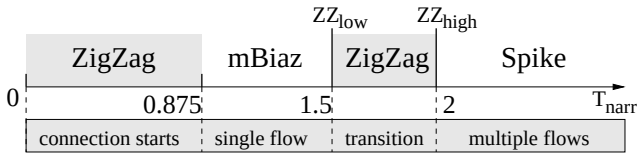


Fig. 11. ZBS scheme (when bottleneck link is not under-utilized). Axis $T_{narr} = T_{avg}/T_{min}$, where T_{avg} is the average packet inter-arrival time, T_{min} is the minimum packet inter-arrival time.

When the bottleneck link is not under-utilized, we use one of three algorithms as shown in Figure 11. Modified Biaz is used when network conditions indicate that the wireless link is the bottleneck and not shared ($T_{narr} \approx 1$), to take advantage of its low M_c and M_w (compared to ZigZag) and high throughput. Spike is used when conditions indicate multiple competing flows ($T_{narr} \gg 1$), which are the conditions under which it has the best performance. ZigZag is used for cases where the network conditions are ambiguous, mostly at the beginning of the connection and when the number of competing flows changes in the middle of the connection.

ZBS starts with the ZigZag scheme, as it has no knowledge about the network conditions at that time and ZigZag behaves well across the widest range of conditions. It then updates T_{avg} and monitors T_{min} at every packet arrival. We set a locking period of 3 seconds or 50 packets received, whichever comes first. The locking period is the minimum duration a scheme must be used before switching to a different one. This prevents frequent switches that might otherwise occur from start/stop of short-lived traffic streams (e.g., short HTTP downloads), occasional severe wireless error (caused by a vehicle passing through the shadow of a bridge or a building), etc.

After the locking period, ZBS decides the next scheme to use. If it uses a different base scheme, the locking period is reset, and the new scheme is frozen for that period. If a new scheme is not chosen at the expiration of the locking period, ZBS applies the switching algorithm at every packet arrival thereafter, and is free to switch when next indicated.

We derived the three thresholds in the switching predicate as follows:

• **0.875:** This is the lower threshold for deciding that the wireless link is not shared, as in the WLH topology. Because of the good performance of mBiaz in this case, we want to be generous in choosing this threshold. Therefore, we check the minimum possible T_{narr} when there is only one flow using the wireless link. In such a case, the minimum T_{narr} happens when the wireless link bandwidth is fully utilized. With no congestion loss, $T_{narr} = 1$. Considering a 5% congestion loss as we saw in Section VI-C, then on average $T_{narr} = 18.5/19 = 0.974$ (because for every 19 inter-arrival periods of length T_{min}, one corresponds to 2 packets due to a congestion loss, and the other 18 correspond to 1 packet). The lowest T_{narr} happens after a severe congestion loss, because the $inter\_arr\_time$ is still close to T_{min} at a congestion loss but will be divided by the number of packets lost. The threshold of 0.875 allows mBiaz to still be used after a congestion loss of up to 5 packets:

$$0.974 * 0.875 + \tfrac{1}{5} * 0.125 = 0.877 > 0.875.$$

As congestion loss of 6 or more consecutive packets is rare in our experiment, this is a generous condition for concluding an unshared wireless bottleneck link. The other reason that we are generous in choosing this threshold is because, if mBiaz were mis-used in the case where the wireless link is shared, the connection would experience lower throughput and longer packet inter arrival time, or higher T_{narr}, which self corrects the mistake.

• **1.5 and 2.0:** The underlying difference between the WLH and WB topologies is the number of flows using the wireless link. Because of mBiaz's poor performance once the link is shared, we want to switch to some other scheme when the wireless link is no longer used by a single connection. Using T_{narr}, the most difficult case to differentiate whether the bottleneck wireless link is being shared is when there are only two flows because, in both cases, T_{narr} can fall in the range of 1 to 2. Without sharing and competition, $T_{narr} > 1$ when the wireless link is not 100% utilized. With competition of 2 flows, $T_{narr} < 2$ when there is a severe congestion loss (as explained above). Our solution is to use ZigZag in the ambiguous area of $[ZZ_{low}, ZZ_{high}]$. The range of $[ZZ_{low}, ZZ_{high}]$ should not go below 1 because it is very likely that there is only one flow using the wireless link when $T_{narr} \approx 1$; it also should not go too high above 2 since the wireless link is likely shared by multiple connections in that case. With this guidance, we tested ZZ_{low} ranging over [1.0, 2.0], and the window size of $(ZZ_{high} - ZZ_{low})$ ranging over [0, 1]. The results confirmed what we expected based on the behavior of the three base algorithms:

• With a fixed window size of $(ZZ_{high} - ZZ_{low})$, the position of ZZ_{low} and ZZ_{high} affects the relative amount of time mBiaz and Spike are used in the hybrid scheme: low ZZ_{low} and ZZ_{high} mainly cause high M_c and high congestion loss in the WLH topology (no competition on bottleneck wireless link); while high ZZ_{low} and ZZ_{high} cause low throughput in the WB topology (competition for the wireless link).

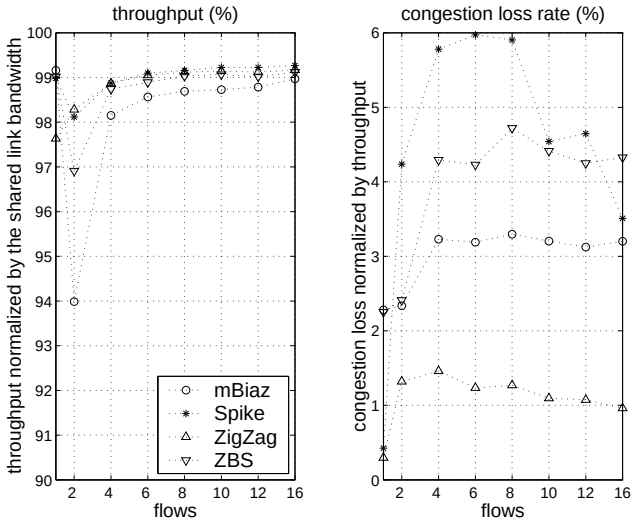
• The window between ZZ_{low} and ZZ_{high} affects the usage of ZigZag. We have tried removing ZigZag from the hybrid scheme by letting $ZZ_{high} = ZZ_{low}$ and using either mBiaz or Spike when $T_{narr} < 0.875$. The results showed that switching only between mBiaz and Spike has the problem of either causing high M_c and high congestion loss in the WLH topology or low throughput in the WB topology with two flows.

Based on our experiments, $ZZ_{low} = 1.5$ and $ZZ_{high} = 2.0$ provides the best results across network conditions. We will gain a better understanding of ZigZag's contribution in the next section where we examine the performance of the switching scheme.

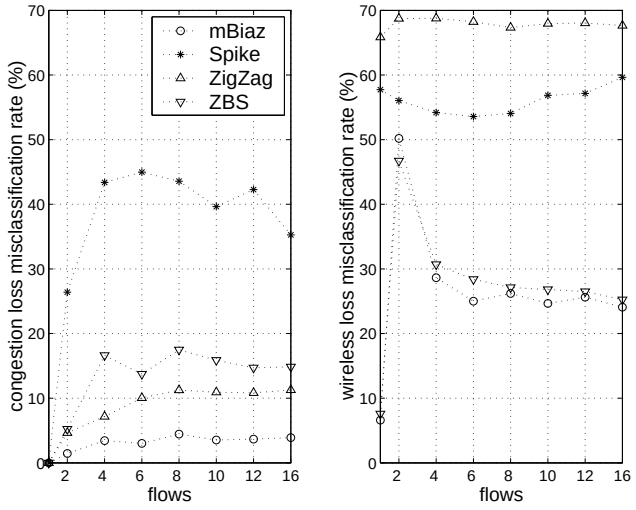
B. Performance of the ZBS algorithm

Figures 12 and 13 show the performance of ZBS in the WLH and WB topologies, respectively. Modified Biaz, Spike and ZigZag are shown for comparison. Figure 14 shows the fraction of time the three base algorithms are used by ZBS.

Overall, in both topologies, ZBS reaches throughput close to that of omniscient, and maintains relatively low M_c and congestion loss, regardless of the number of flows. ZBS uses mBiaz 85% of the time in the WLH topology, and uses Spike 95% of the time in the WB topology, i.e., it picks the right scheme for the given network conditions.



(a) throughput and congestion loss rate



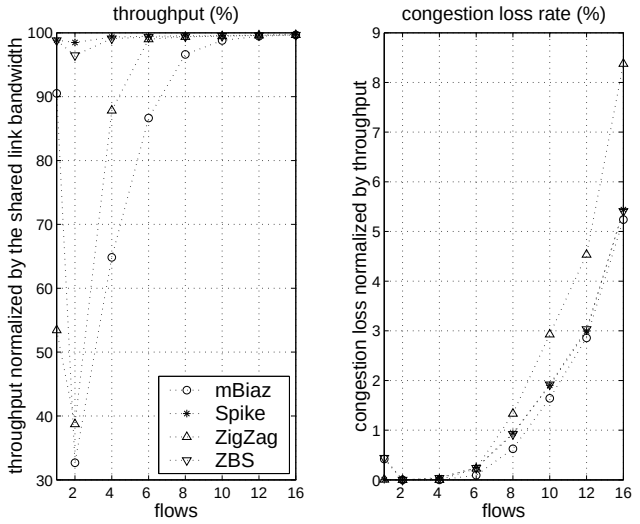
(b) misclassification rates

Fig. 12. The hybrid scheme in wireless last hop topology

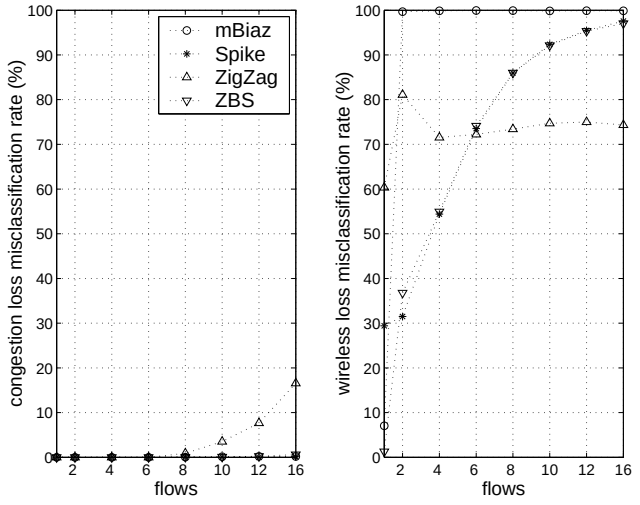
With one flow in both topologies, mBiaz is used more than 98% of the time because there is no real topological difference between the two, and mBiaz performs the best in both scenarios.

Spike is used only 3% of the time in the WLH topology. However, we see from Figure 12 that this already causes the M_c of ZBS to be higher than that of both mBiaz and ZigZag. This behavior provides better understanding of why, without ZigZag, switching only between Spike and mBiaz could easily incur high M_c in the WLH topology. By setting $ZZ_{high} = 2.0$, we are able to keep M_c at a reasonable level. As a result, ZigZag is heavily used (close to 50%) in the WB topology with two flows. However, this does not cause low throughput as ZigZag by itself would.

In the WB topology, it looks counterintuitive that Spike usage actually decreases as the number of flows increases above



(a) throughput and congestion loss rate



(b) misclassification rates

Fig. 13. The hybrid scheme in wireless backbone topology

six. However, this can be explained by the wireless loss desynchronization effect as the number of flows increases: it is less likely two packets belonging to a flow would be buffered consecutively at the wireless buffer with a large number of flows. In other words, the T_{min} a flow observed is often not the transmission time of one packet over the wireless link. Therefore, T_{narr} could fall below 2, in which case ZigZag is used.

Summary. The ZBS hybrid LDA performed well across different topologies and numbers of flows. In most cases, it closely matched or exceeded the performance of the best base algorithm for that scenario.

VIII. Fairness and TCP-Friendliness

So far we have examined the overall *average* performance of each LDA both in isolation as well as when it competes with other flows using the same LDA. Now we evaluate the fairness

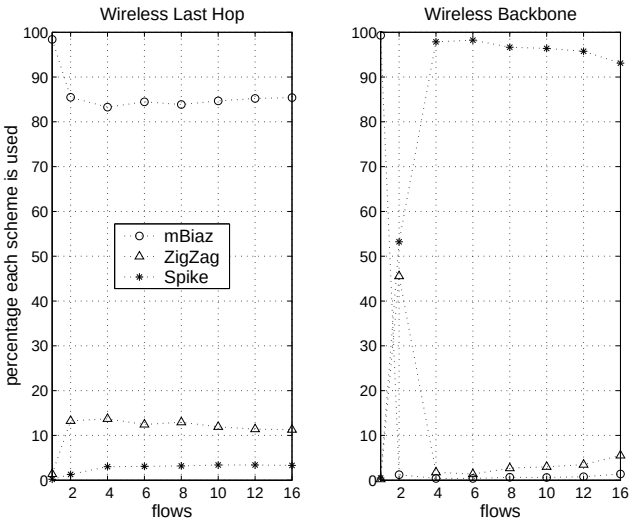


Fig. 14. Relative usage of the 3 base schemes by the switching algorithm

of each algorithm by examining (a) the deviation of the throughput of each individual connection when all flows are using the same LDA, and (b) how fair and competitive each LDA is when competing with TCP Reno without wireless losses, a scenario that approximates the use of a snoop agent [2]. Ideally, we would like an LDA to be fair and stable in both cases, which means that a flow using the LDA is able to obtain and keep its fair share of the available bandwidth and does not become starved or starve others.

A. Fairness among flows using the same LDA

Figure 15 shows the standard deviation (in %) among different flows when all connections are of the same type (e.g., use the same LDA). The left plot represents the WLH topology, and the right plot the WB topology. The last symbol on the legend, TCP(NWL), represents normal TCP traffic with *no* wireless loss on the wireless last link or the wireless backbone. Each point on the plot corresponds to a particular LDA and number of flows. For each such pair, we first normalize the throughput of each individual connection by the mean of all connections, and then compute the standard deviation of the normalized throughput. We then plot the average of standard deviations over 10 trials. For example, in one trial of ZBS in the WB topology with 4 flows, the throughputs of the 4 flows are 13303, 13826, 13123 and 11955 packets. Dividing by their mean, 13052 packets, the normalized throughputs are 1.02, 1.06, 1.006 and 0.92. The sample standard deviation of these normalized throughputs is computed as:

$\sigma = sqrt(\frac{(1.02-1)^2+(1.06-1)^2+(1.006-1)^2+(0.92-1)^2}{4-1}) = 6\%$

The same calculation is done for the other 9 trials, and the final point on the plot is the average of the 10 deviations computed.

Wireless Backbone topology. In the WB topology, all TFRC types of traffic have relatively consistent and low deviation in the range of 3–7%. The significant difference between TCP and TFRC when both experience wireless loss is due to the intrinsic mechanisms used to control sending rate: TCP is ACK-based while TFRC is rate-based. This result shows that compared to TCP, TFRC not only achieves lower fluctuations in sending

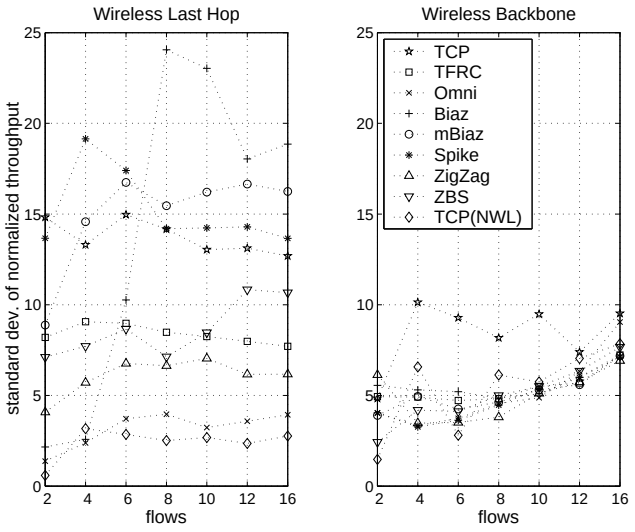


Fig. 15. Standard deviation of throughput among same type of flows

rate over time within a connection, but also is more fair among TFRC connections when there are wireless losses. Omniscient and all LDA traffic have low deviation in throughput because the conditions which affect their performance exist at only one place: the wireless link buffer. Since that buffer is common to all connections, its level and composition have the same effects on all flows. Therefore the deviation among flows is small. TCP with no wireless loss has a deviation among flows similar to omniscient and all LDAs, which means they are all fair in this topology. The deviation tends to increase with the number of flows, because with more flows in the network, the average bandwidth of each connection is lower. With a smaller average bandwidth, the same absolute difference of throughput produces a larger deviation.

Wireless Last Hop topology. In the WLH topology, the deviation of all traffic types except omniscient is higher than in the WB topology. For the original TFRC, this is because the wireless loss on the last link is different for different connections. For the LDAs, the main conditions which affect their performance now exist at two places: the common wired link buffer and the last wireless link buffer. Because what happens at the last link buffer is often different from connection to connection, the deviation is higher. Omniscient TFRC and TCP with no wireless loss are similar, consistently having the lowest deviation. Compared to TCP with no wireless loss, all LDAs in this topology are not as fair among different flows.

Looking more closely, the effect of the separate wireless link buffer, and therefore the actual fairness, is different for different LDAs. Spike and both Biaz schemes have very high deviations ($> 14\%$) in most cases. ZigZag, original TFRC, and ZBS have much lower deviation, 7–10% in most cases. This is because Spike and both Biaz schemes are very sensitive to the buffer level at the wireless last link, albeit in different manners.

- For the two Biaz schemes, connections temporarily having low sending rate are disadvantaged because they are likely to experience longer packet inter-arrival time which makes them classify wireless loss as congestion and reduce rate even further.
- For the Spike scheme, connections which obtained high sending rate quickly at the beginning (due to different start times), will observe large ROTT due to buffer build up at the wireless link. These connections are less aggressive as they are more likely to classify wireless loss as congestion.
- Similar to the argument in Section VI-C, ZigZag is not very sensitive to the history of the last wireless link buffer: the exponentially averaged $rott_{mean}$ and $rott_{dev}$ gradually forget the past, making any advantages or disadvantages diminish.
- The reason for the low throughput deviation of ZBS is interesting. In the WLH topology, as seen in Figure 14, mBiaz is used most of the time. However, whenever the throughput of a connection is low, T_{narr} increases and ZigZag or Spike is used. Both ZigZag and Spike are more aggressive than mBiaz in the WLH topology as they have higher M_c. Eventually the disadvantaged connection will catch up its fair share in throughput and switch back to mBiaz. This behavior shows that by switching among different base algorithms, our hybrid scheme not only has more consistent good performance on average across topologies and competition, but also makes each individual connection more stable and improves the overall fairness in the WLH topology.

Summary. When competing with the same type of traffic, all LDAs are as fair and stable as the standard TCP if most or all the conditions which affect the LDA's performance are common to all flows, as in the WB topology; they are less fair and stable than TCP if some of those conditions are different from flow to flow, as in the WLH topology. However, ZigZag and ZBS are the fairest among all LDAs.

B. Fairness with TCP

To evaluate the fairness of the LDAs with TCP traffic, we simulate connections using an LDA competing for network resources with connections that use TCP Reno that are wireless-loss-free, i.e., do not suffer wireless losses. This scenario approximates the use of a snoop agent [2] that hides all wireless loss from the sender, enabling TCP to obtain about the same throughput as if there were no wireless loss. Because snoop is designed to operate at the base station for mobile hosts, we only test the fairness and competitiveness of LDAs with snoop in the WLH topology.

To determine how TCP-friendly these schemes are on the WLH topology, we simulated a total number of flows ranging from 2 to 16. Half of the flows used TCP and are immune to any wireless loss, and the other half used one of the LDAs, TCP, TFRC, or omniscient TFRC and are subject to the same wireless loss seen earlier. The left subplot of Figure 16 shows the results with a low average bandwidth (BW) per flow (130Kbps), the right subplot is for a higher average BW per flow (800Kbps). The x-axis shows the number of total flows, and the y-axis shows the average normalized throughput of the TCP flows which do not experience any wireless loss. As the throughput is normalized by the fair share of a flow (130Kbps or 800Kbps), a value close to 100% means that the scheme is as TCP-friendly as TCP; a lower value means that the LDA is more aggressive than TCP. The 130Kbps average BW case has exactly the same network parameters as the previous WLH topology. In the 800Kbps case, the BW of each wireless last link is 930Kbps and the BW of the shared wired link is N * 800Kbps, where N is the number of flows. Therefore, in both cases, the max-

imum normalized throughput any flow can get is about 115% ($\approx 150/130 \approx 930/800$).

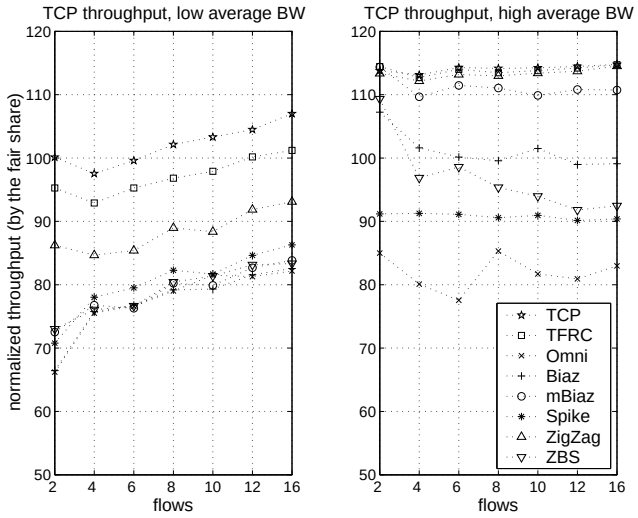


Fig. 16. TCP friendliness in the WLH topology: Shown in the left (right) subplot is the throughput of the aggregated TCP flows when 50% of the flows use TCP and the other 50% use the LDA indicated by the plotted symbol, at an average BW per flow of 130Kbps (800Kbps).

From Figure 16, we see that, overall, LDAs are more aggressive when the average BW is lower. In both cases, the omniscient TFRC is most aggressive as the throughput of TCP is the lowest. In the lower average throughput case, all LDAs are as aggressive as the omniscient TFRC except ZigZag which is more TCP friendly. In the higher average throughput case, mBiaz and ZigZag are as uncompetitive as the wireless unaware TCP and TFRC because the throughput of TCP is close to the maximum of 115% when competing with them. Biaz and ZBS are quite TCP-friendly, as the TCP throughput is close to 100% most of the time. Spike and omniscient are the most aggressive.

Summary. In the WLH topology, when competing with TCP flows which are free from wireless loss, omniscient TFRC is the most aggressive at either bandwidth. The LDAs are more TCP-friendly when the average BW is higher (800Kbps) than when the average BW is lower (130Kbps).

C. Summary

In summary, our evaluation of LDA fairness shows that:

- The fairness among flows using the same LDA depends on topology. They are as fair and stable as the standard TCP if the common path contains most conditions affecting the LDA.
- When competing with wireless-loss-free TCP flows (conceptually equivalent to the TCP with snoop agent), the aggressiveness of the LDAs is sensitive to the underlying bandwidth. At high bandwidths, the LDAs are quite TCP-friendly, but they become more aggressive at lower bandwidths. In all cases, though, omniscient TFRC is the most aggressive.

IX. Implementation Issues

In this section, we discuss the computational complexity of the LDAs as well as additional implementation issues.

A. Computational Complexity

On a high level, three parameters are used by LDAs to differentiate losses: relative one-way trip time (ROTT), packet inter-arrival time, and the number of packets lost consecutively in the most recent loss event. Various statistical values (e.g., minimum, maximum, mean and deviation) are calculated for the ROTT and the packet inter-arrival time in each LDA.

Table IV lists the statistical values used by each LDA. As there is no difference in the parameters used by Biaz and mBiaz, "Biaz" in Tables IV and V stands for both. Among the values listed in Table IV, some marked with an asterisk (*) need to be updated at every packet arrival, the others only need to be calculated after a loss event.

TABLE IV
STATISTICAL VALUES USED BY EACH LDA

Statistical Value	LDA(s)
number of packets lost: n	Biaz, ZigZag, ZBS
instan. packet inter-arrival time: T_i	Biaz, ZBS
*min. packet inter-arrival time: T_{min}	Biaz, ZBS
*ROTT min./max.: $rott_{min}, rott_{max}$	Spike, ZBS
spike thresholds: $B_{spikestart}, B_{spikeend}$	Spike, ZBS
*ROTT mean/dev.: $rott_{mean}, rott_{dev}$	ZigZag, ZBS
*average packet inter-arr. time: T_{avg}	ZBS
normalized T_{avg}: T_{narr}	ZBS
time, pkt sequence # of locking period	ZBS

Based on Table IV, Table V summarizes the computational complexity of each LDA at each packet arrival and after a loss event. The complexity of the underlying original TFRC based on [4] and its current implementation in *ns2* is also listed for comparison. It is obvious that the hybrid scheme is the most complex as it uses all three base algorithms to differentiate losses. We deem that this computational complexity is acceptable as it is comparable to that of the original TFRC. In terms of space requirements, the extra memory used by the hybrid scheme for all 13 variables listed in Table IV is minimal.

TABLE V
COMPUTATIONAL COMPLEXITY OF EACH LDA

LDA	additions		multiplications	
	pkt arrival	after loss	pkt arrival	after loss
Biaz	2	4	0	1
Spike	3	3	0	1
ZigZag	4	3	4	1
ZBS	8	11	7	3
TFRC	6	∼12	4	7 + n

B. Other Issues

Scalability. To test the scalability of the LDAs, we performed simulations with 128 flows in the WLH topology. Table VI summarizes the results. Values in Table VI are all normalized in the same way as in Figures 9 and 12. Comparing results shown in the table with those in the two figures, they match very well except for Spike. It performs better at 128 flows, with lower M_c

and therefore lower congestion loss. This preliminary experiment indicates that the LDAs scale well with large numbers of flows in the network.

TABLE VI
PERFORMANCE WITH 128 FLOWS IN THE WLH TOPOLOGY

	Omni	Biaz	mBiaz	Spike	ZigZag	ZBS
thput	99	99	99	99	99	99
cong.	5.3	8.5	3.3	1.9	0.7	4.2
M_c	0	32	4.4	29	11	14
M_w	0	17	23	64	68	25

Frequent connections arrival and departure. In our simulations, we constrained the types of traffic in the network and the arrival and departure of different flows (see Section V-C). In more complicated scenarios where there are other types of traffic and connections come and go more randomly and frequently, the stability of various LDAs depends on whether the parameters they use can still reflect the network conditions that characterize congestion and/or wireless losses. We expect parameters which represent statistical limits of an entire connection, e.g., T_{min}, $rott_{min}$, and $rott_{max}$, might need to be "refreshed" from time to time, i.e., representing limits not of a whole connection, but of a shorter period instead. However, to answer this question satisfactorily, more extensive research is needed which is beyond the scope of this paper.

Other queuing policies. It would be interesting to study how these LDAs perform when queuing policies other than DropTail are used at the intermediate routers, e.g., Random-Early-Drop (RED). Based on our understanding of each LDA, we expect that RED would not have any significant impact on the Biaz schemes but could hurt the performance of both Spike and ZigZag. Thorough investigations of the effects on the LDAs by RED and other queuing policies are left for future research.

X. CONCLUSION

In this paper we evaluated three base algorithms for differentiating congestion and wireless losses for use with congestion-sensitive video transport protocols. The Biaz algorithms perform well (in isolation) on the wireless last hop (WLH) topology for which they were designed, but lose their ability to differentiate when the wireless bottleneck link has competition from other flows. The Spike algorithm performs well in the wireless backbone (WB) topology, particularly when there are competing flows. The ZigZag algorithm, a new algorithm we propose in the paper, has relatively consistent performance across different topologies, competition, and fairness scenarios, but its performance is sensitive to its sending rate.

Generally speaking, we find that LDAs based upon packet inter-arrival times (Biaz and mBiaz) do not behave well when there is competition for the bottleneck wireless link, and are only suitable for a particular topology and no competition on the wireless link. The LDAs based upon ROTT (Spike, ZigZag), however, are able to correlate congestion with particular losses much more accurately across a wide range of scenarios, although they may have relatively high wireless misclassification rates in particular situations.

Based on the insight we obtained evaluating the base algorithms, we then proposed a hybrid scheme, ZBS, that chooses a different base algorithm best suited to the current network conditions. The choice is mainly based on the relationship between the inter-arrival time and its minimum. The hybrid has excellent performance across both topologies, regardless of the number of competing flows, while striking a good balance between performance and fairness.

Finally, we discussed the computational complexity and other implementation issues of the LDAs. We showed that the complexity of the LDAs is comparable to that of the underlying TFRC algorithm.

Acknowledgments

This work was supported by the State of California CoRe Program, the Center for Wireless Communications at UCSD, Ericsson, the California Institute for Telecommunications and Information Technology, and by the National Science Foundation under grant CCR-0105734. Portions of this work appeared in the ACM Multimedia Computing and Networking Conference 2002. The authors appreciate the many helpful suggestions made by the anonymous reviewers.

REFERENCES

[1] A. Bakre and B. Badrinath, "I-TCP: Indirect TCP for mobile hosts," in *Proc. 15th Intl. Conf. on Distributed Computing Systems (ICDCS)*, Vancouver, Canada, May 1995.

[2] H. Balakrishnan, S. Seshan, and R. Katz, "Improving reliable transport and handoff performance in cellular wireless networks," *ACM Wireless Networks* 1(4), pp. 469–481, Dec 1995.

[3] R. Yavatkar and N. Bhagwat, "Improving end-to-end performance of TCP over mobile internetworks," in *Workshop on Mobile Computing Systems and Applications*, pp. 146–152, Santa Cruz, CA, Dec 1994.

[4] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proc. SIGCOMM 2000 Conference*, pp. 43–56, Stockholm, Sweden, Aug 2000. http://www.aciri.org/tfrc.

[5] S. Biaz and N. Vaidya, "Discriminating congestion losses from wireless losses using inter-arrival times at the receiver," in *Proc. 1999 IEEE Symposium on Application-Specific Systems and Software Engr. and Techn.*, pp. 10–17, Richardson, TX, Mar 1999.

[6] N. Samaraweera, "Non-congestion packet loss detection for TCP error recovery using wireless links," in *IEE Proceedings of Communications*, 146(4), pp. 222–230, Aug 1999.

[7] Y. Tobe, Y. Tamura, A. Molano, S. Ghosh, and H. Tokuda, "Achieving moderate fairness for UDP flows by path-status classification," in *Proc. 25th Annual IEEE Conf. on Local Computer Networks (LCN 2000)*, pp. 252–61, Tampa, FL, Nov 2000.

[8] "High performance wireless research and education network." URL: http://hpwren.ucsd.edu.

[9] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," in *IEEE/ACM Transactions on Networking*, vol.5, (no.6), pp. 756-69, Dec. 1997.

[10] C. Casetti, M. Gerla, S. Mascolo, M.Y. Sanadidi, and R. Wang "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links," in *Proc. ACM Mobicom 2001 Conference*, pp. 287-297, Rome, Italy, July 16-21 2001

[11] S. Biaz and N. Vaidya, "Distinguishing congestion losses from wireless transmission losses: A negative result," in *Proc. 7th Intl. Conf. on Computer Communications and Networks*, Lafayette, LA, Oct 1998.

[12] W.C. Jakes, *Microwave Mobile Communications*, Wiley-Interscience, 1974.

[13] P. Dent, G.E. Bottomley, and T. Croft, "Jakes' model revisited," in *Electronics Letters*, 24th June, 1993, vol.29, No. 13.

[14] Q. Zhao, P.C. Cosman, and L. Milstein, "Tradeoffs of source coding, channel coding and spreading in CDMA systems," in *Proc. Milcom 2000*, Los Angeles, CA, Oct 2000.

[15] *ns-2* network simulator (ver 2). LBL, URL: http://www.isi.edu/nsnam/ns.