End-to-end differentiation of congestion and wireless losses

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

ECE Dept., †CSE Dept., Univ. of California at San Diego, La Jolla, CA 92093
{scen,pcosman}@code.ucsd.edu voelker@cs.ucsd.edu

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. One goal of this paper is to evaluate various LDAs under different wireless network topologies and competing traffic. A second goal of this paper is to propose and evaluate a new LDA, called ZigZag, as well as a class of hybrid algorithms based upon ZigZag.

We then evaluate these LDAs via simulation. Based upon our simulation results, we find that no single base algorithm performs well across all topologies and competition. However, the hybrid algorithms perform well across topologies, competition, and in some cases match or exceed the performance of the best base LDA for a given scenario.

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

1. 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.[1–3] 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 treats it as a received packet. Recall that the transport protocol is best-effort; wireless losses are tolerated using redundant encodings at the application layer without retransmissions.

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. The second is derived from Tobe et al.[7] that uses relative one-way trip times (ROTT).

A second goal of this paper is to propose and evaluate a new LDA, called ZigZag, as well as a class of hybrid algorithms based upon ZigZag. 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.

To achieve these goals, we evaluate these algorithms via simulation using *ns*. 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 satellite backbone links, wireless LAN networks, or high-performance backbones.[8] We then study the LDAs under various scenarios of competing traffic where multiple flows use the same LDA. Finally, we evaluate the hybrid LDAs that combine the individual strengths of the base algorithms.

Based upon our 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 hybrid algorithms based on ZigZag are able to differentiate both losses as well as topology, and use this to leverage the particular strengths of the base LDAs. As a result, the hybrid algorithms perform well across topologies and competition, and in some cases match or exceed the performance of the best base LDA for a given scenario.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 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 4 and 5 discuss the performance metrics and network parameters used in our simulation and evaluation of the LDAs. Sections 6 and 7 describes the simulation results in terms of throughput, network topology and traffic competition. Finally, Section 8 summarizes and concludes.

2. 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 significantly 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., trade off source and channel coding). Since we are interested in best-effort transport protocols and 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 wireless error rate is high. However, NCPLD was not evaluated on different wireless topologies.

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).[10] 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 3.1).

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 show that their scheme is able to differentiate congestion-related losses from random losses found on a wireless link, although they only investigate its use to detect congestion. In this paper we describe a version of this algorithm (Section 3.2) designed to explicitly differentiate between congestion and wireless losses, and evaluate its performance.

3. BASE ALGORITHMS

The three basic LDAs we experimented with are called Biaz, Spike, and ZigZag, and they are described in this section. All other schemes we tested are based on these three fundamental schemes, and are introduced in Section 6. 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 user which has precise knowledge of the cause of every packet loss.

3.1. 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_o denote an out-of-order packet received by the receiver. Let P_i denote the last in-sequence packet received before P_o. Let T_i denote the time between the arrivals of packets P_i and P_o. Finally, let the number of packets missing between P_i and P_o be n (assuming that all packets are of the same size).

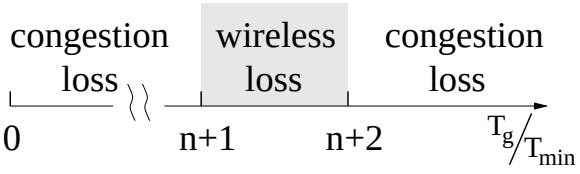

Figure 1: Biaz Scheme

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 if P_o 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_o arrives much earlier than it should, then packets ahead of it probably got 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 not shared by other connections competing for the link.

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 traffic would cause. All other basic schemes have lower congestion loss than that of omniscient traffic.

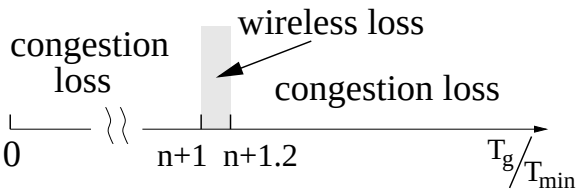

Figure 2: Modified Biaz Scheme

Checking the threshold 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, the average size of the lost packets must be smaller than the smallest packet received so far, which usually becomes more rare as the length of the connection gets longer. It does not occur in our experiment 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 when packets corrupted by wireless error are not buffered one after the other at the wireless link. 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 but 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 a higher share of bandwidth when competing with other traffic. The high congestion loss observed with the Biaz scheme indicates that the window is probably too large. Therefore, we lowered the upper limit in the Biaz scheme to determine a limit that provides a reasonable performance tradeoff. The results for values 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 when competing with other traffic in the wireless last hop topology (see Section 5.1). The Biaz scheme's performance is insensitive to the choice of upper limit in the wireless backbone topology. Therefore, we choose $(n + 1.2) * T_{min}$ in the modified Biaz scheme (Figure 2).

3.2. Spike scheme

The Spike scheme was derived from [7], although there was no explicit effort to differentiate wireless loss from congestion loss in that work. 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.

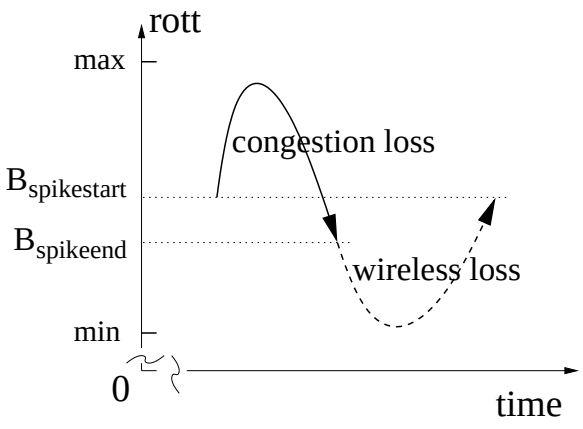

Figure 3: Spike Scheme

The 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 state 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 state changes out of the spike state. When the receiver detects a loss because of a gap in the sequence number of received packets, it classifies the losses based on the current state (see Figure 3).

In [7], the threshold values $B_{spikestart}$ and $B_{spikeend}$ were hard-coded to be 20 ms and 5 ms, respectively. However, intuitively these thresholds should be dependent on the overall network delays. For a connection that never experiences delays lower than 5ms or higher than 20ms, these thresholds will make the algorithm useless. Therefore, we think the thresholds should be chosen in the following manner:

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

where $rott_{max}$ and $rott_{min}$ are the maximum and minimum relative one-way trip time observed so far, and $\alpha \geq \beta$. 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. A higher α 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 $\alpha \leq 0$, then the misclassification of congestion loss is 0% and wireless is 100%. It is exactly the opposite for the value β. To explore the sensitivity of the performance of the Spike scheme on these parameters, we conducted tests with α ranging from 0.35 to 0.95 and β ranging from 0.05 to 0.45, and found $\alpha = 0.5$ 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 5.1). The Spike scheme's performance in the wireless backbone topology is insensitive to the choice of α and β.

3.3. ZigZag scheme

In addition to the above schemes derived from previous work, we propose a new scheme for differentiating wireless from congestion losses 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 $d = rott_i - rott_{mean}$. A loss is classified as wireless if (n=1 AND $d < -rott_{dev}$) OR (n=2 AND $d < -rott_{dev}/2$) OR (n=3 AND $d < 0$) OR ($n > 3$ AND $d < rott_{dev}/2$). Otherwise the loss is classified as congestion loss.

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

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

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 4 or more packets is more likely to be classified as wireless loss, as wireless losses often display bursts of correlated errors.

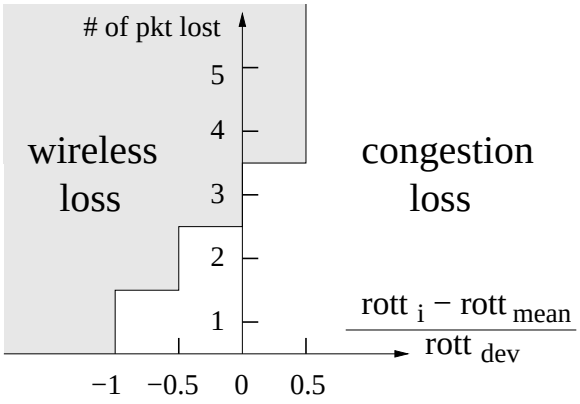

Figure 4: ZigZag Scheme

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.

4. 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, to a certain extent, we do not care what the misclassification rates are; ultimately, we are concerned with the throughput of the traffic stream that results from using the algorithm, and with whether the algorithm causes 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 user can have a throughput 200% higher than an unaware TFRC user, depending on the topology and wireless loss severity. A primary goal is to have a throughput close to that of the omniscient TFRC user.

Congestion Loss: The amount of congestion loss experienced by a TCP connection or other traffic when competing with a user equipped with our differentiation scheme is affected by the behavior of our scheme. When competing with TCP, or some other type of traffic, the throughput of those users should not be too much lower than when traffic using our scheme does not exist. For two 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 will limit 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) lower throughput when competing with itself, and (c) higher throughput when competing with different types of traffic (less friendly to those unaware of wireless loss, e.g., TCP and TFRC; more aggressive when competing with omniscient).

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

Thus the values of M_w and M_c should be considered together with the corresponding throughput and congestion loss.

5. NETWORK PARAMETERS

5.1. 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 less than $N * rate_{wlast}$, so the N streams compete for bandwidth at the common link. This type of topology simulates a cellular network or satellite Direct-TV system, where each wireless link has a relatively constant bandwidth.

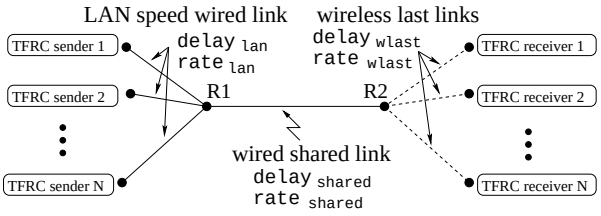

Figure 5: Wireless Last Hop Topology

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 either an 802.11 radio link, a satellite link, or another type of high bandwidth radio link as in the UCSD HP-WREN[8] project.

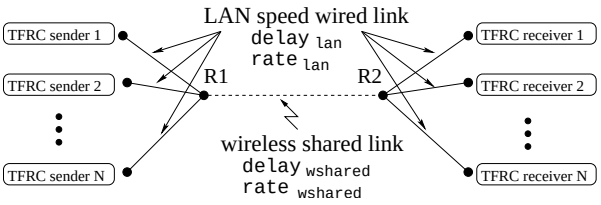

Figure 6: Wireless Backbone Topology

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 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. Our experiments show that a wireless LAN shows essentially identical results as a WB topology when the corresponding links bandwidth 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, only the WB topology is considered, with its results directly applicable to the wireless LAN case.

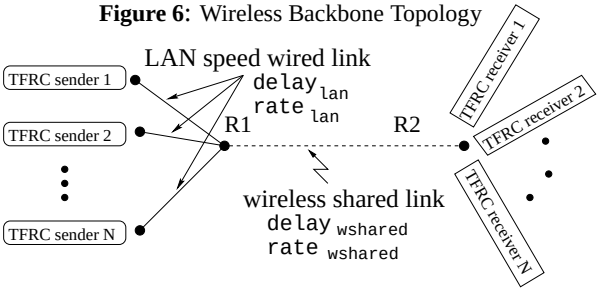

Figure 7: Wireless LAN Topology

5.2. Wireless Loss Model

Based on [11], we simulated various CDMA channels with different channel parameters. To generate the error pattern, 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 formed the error patterns used in our *ns* simulations. We used this error model for both the low bandwidth wireless last hop link and for the higher bandwidth wireless backbone link. For example, mobile base stations which would would be used as a backbone network in military applications would fit the same channel model as a cell-based CDMA system. Wireless error only exists in the forward direction from sender to receiver; there is no wireless loss in the reverse direction.

Other system parameters used 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 $f_D T_c = 2.62 \times 10^{-4}$, together with the three combinations of spreading gain and interleaver size given in Table 1. 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.

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}

Table 1. Packet loss probability of high, medium, and low wireless loss

5.3. 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, to make the wireless link be the slowest link, we set the capacity between routers R1 and R2 to be slightly less than twice the wireless link capacity.

For the WB topology, to make the average rate per flow comparable to that in the WLH topology, we set $rate_{wshared} = 800$ Kbps when there are 8 or fewer flows; and $rate_{wshared} = 1600$ Kbps for 10, 12 or 16 flows.

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

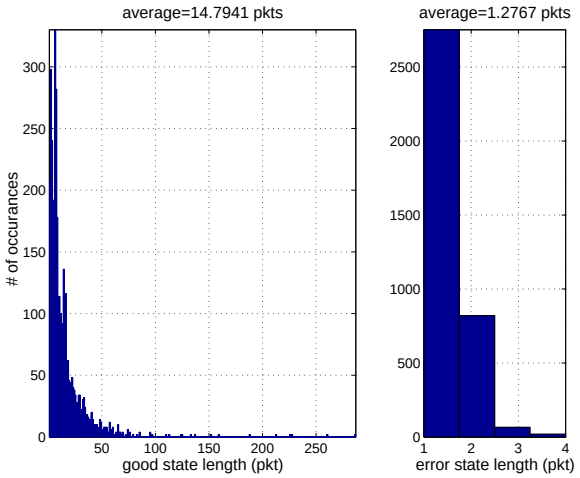

Figure 8: Histogram of good/error state length (high loss)

Delay: Total delays in the network are composed of processing, propagation, transmission and queuing delays. The *(processing + propagation)* delay is set explicitly:
$delay_{lan} = 1$ms; $delay_{wlast} = 10$ms; $delay_{shared} = delay_{wshared} = 20$ms.

The other two are determined implicitly by the choice of other parameters: bandwidth, queue size, etc.

Packet Size: The packet size was chosen to be 762 bytes. For a video coder that encodes video sequences 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 a multiple of 381 bytes, one of the specified packet sizes 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] (web site): $queue\_size(pkts) = max(link\_bandwidth/60K, 6)$

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 200 seconds. For each differentiation scheme in a group, experiments were performed with the same random seed that determines the starting order of and the wireless error pattern experienced by each flow. With different random seeds, the same set of experiments were repeated 10 times, and results were averaged.

6. 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 6.1) and then on the wireless backbone topology (Section 6.2). Finally, we evaluate the algorithms when other flows compete for network resources in both topologies (Sections 6.3 and 6.4).

6.1. 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 4 and 5, and then study the algorithms in the WB topology in Section 6.2.

Table 2 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 and wireless misclassification rates, and congestion loss for each type of flow as percentages. The throughput is normalized by the bandwidth of the bottleneck link; M_c is the fraction of packets lost to congestion that are misclassified as wireless loss; M_w is the corresponding measure for wireless loss; and congestion is the number of packets lost due to congestion divided by the throughput (rather than number of packets transmitted, which depends upon the algorithm). 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	TFRC	Omni	Biaz	mBiaz	Spike	ZigZag
thput	58	83	99	98	98	98.5	97
M_c	0	0	0	0.0	0.0	1.2	0.7
M_w	100	100	0	6.3	6.2	58	65
cong.	0.6	0.2	2.3	2.3	2.3	0.3	0.3

Table 2: Performance for wireless last hop, 1 flow

Looking at the flows that do not use an LDA, we see that TCP and TFRC 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. The Biaz algorithms misclassified no congestion losses, and made few mistakes on wireless losses; this is not too surprising since 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 reduce their sending rate in reaction.

By definition, 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 (containing 6/3 = 2 packets). 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. The low M_c of both Spike and ZigZag shows that the thresholds chosen to parameterize the algorithms are reasonably 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.

6.2. 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 3 shows the results of simulating the algorithms on the WB topology described in Section 5.1. 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 backbone link can effectively be ignored since its bandwidth is much higher than 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.

	TCP	TFRC	Omni	Biaz	mBiaz	Spike	ZigZag
thput	25	34	98	95	87	98	50
M_c	0	0	0	0.0	0.0	8.8	0.0
M_w	100	100	0	2.6	6.8	27	60
cong.	0.1	0.0	0.4	0.4	0.4	0.0	0.0

Table 3: Performance for wireless backbone, 1 flow

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 bottleneck more than topology; in subsequent experiments below, however, we will see more of an influence of topology on the performance of the algorithms.

From Table 3, 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 phase and enables it to open its congestion window more smoothly in the linear increase phase.

The performance of the LDAs 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 in 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.

6.3. Competition with Wireless Last Hop

Now that we have evaluated the algorithms on both topologies in isolation, we next evaluate them in the face of competing flows. We start with the WLH 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 2, which we include for comparison. Figure 9 shows three graphs, one that shows throughput (top left), a second that shows congestion loss (bottom left), and a third that shows M_c and M_w (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 competing flows, so a high throughput means that not only is the algorithm performing well but that it competes well with itself, too. 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 start to be misclassified as wireless losses. This high M_c causes very high congestion loss (8–11%) 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 3.1). Figure 9 shows that the modified Biaz algorithm 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, with more than one flow, packets from the same flow usually intersperse with packets from other flows. As a result, the packet after the wireless loss often does not come immediately after one T_{min} time interval. With the strict window size used to classify loss as wireless, the modified Biaz scheme makes more classification errors on wireless losses than the original Biaz. This is the primary cause of its lower throughput, especially when there are two flows in the network.

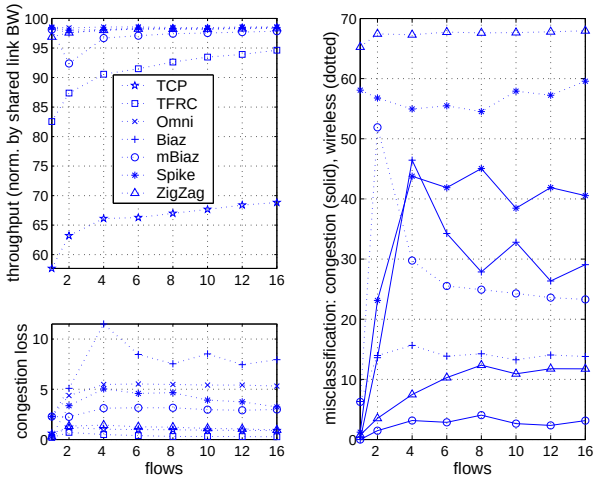

Figure 9: Competition with the wireless last hop topology

The Spike scheme has consistently high throughput across all numbers of flows. However, 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 the level of each individual buffer. Note that 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 mean ROTT 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 performance of 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 at this bottleneck rate in the WLH topology.

6.4. 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 metrics and 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. When the network has more than one flow, there are two main differences between the performance of the algorithms on the two topologies:

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

- In the WLH topology, the maximum receiving rate of any flow is bounded by the rate of the wireless last link, 150Kbps. Thus no flow can get more than $150/(N*130) = 1.15/N$ share of the common link bandwidth, where N is the number of flows.
- In the WB topology, the maximum receiving rate of a flow could potentially reach the capacity of the shared link; i.e., the upper limit on the share of the common link bandwidth for each flow is 1.

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 800/N Kbps when $N \leq 8$, and 1600/N Kbps for N = 10, 12, 16. We chose these average rates so that they would roughly correspond to average rate for each flow in the WLH topology.

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 90% and 97% for TCP and TFRC respectively, while with only one flow at 150Kbps, their utilization is only 58% and 83%. 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.

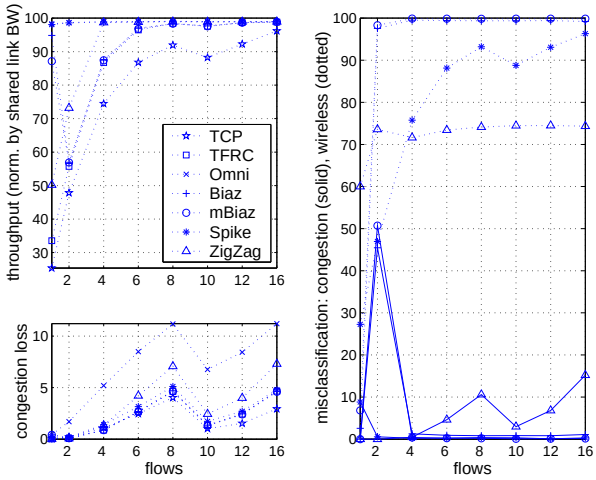

Figure 10: Competition in the wireless backbone topology

Both Biaz schemes have high 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 one after the other at the wireless link. This situation is unlikely when there are two or more flows sharing the link. As a result, their high M_w make the Biaz schemes essentially useless as LDAs for this topology; their throughput is about the same as original TFRC.

The Spike scheme works well in this topology as buffer build up can still 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_c and 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. When there are 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 400 Kbps per flow, 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 four or more flows. The reason for this is due partly to the fact that there are very few wireless losses involving more than 2 packets, and partly because of lower average rate per flow. Finally, although its M_c is mostly lower than in the WLH topology, due to the first difference between the two topologies stated above, its M_c is more costly in this environment. Therefore, it has higher congestion loss than in the WLH topology.

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 always or mostly happens.

6.5. 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 links bandwidth. Modified Biaz also performs well when there is a large (> 6) number of flows. The original Biaz and Spike schemes both have an unacceptably high M_c.

• When there is competition among flows in the WB topology, the best scheme is Spike. ZigZag is still useful, although it suffers from a high congestion loss. Both Biaz schemes lose their differentiation ability and perform the same as TFRC.

We conclude that none of the base algorithms performs consistently well across topologies and in the face of competition from other flows. As a result, we now investigate whether we can enhance the base algorithms to find a design that performs well across different topologies and different numbers of competing flows.

7. EVALUATION OF ALGORITHM HYBRIDS

Since no single base algorithm performed well across all topologies and in the face of competition, in this section we investigate hybrids of the base algorithms. In the WLH topology, ZigZag and modified Biaz behave very well, while in the WB topology, Spike is the best performer and ZigZag has some usefulness. Observing this behavior, can we design a switching algorithm that can select the right scheme for the right topology? The key is to differentiate between the two topologies based on some transmission characteristics. 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.

When the lowest bandwidth link is shared, the average packet inter-arrival time (T_{avg}) would be close to $N * T_{min}$, where N is the number of flows which share the link, and T_{min} is the minimum inter-arrival time. If the slowest link is not shared, then T_{avg} should be close to T_{min}. T_{avg} is computed by exponential averaging:

$$T_{avg\_new} = 0.75 * T_{avg\_previous} + 0.25 * inter\_arr\_time/pkts$$

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} is usually smaller than $inter\_arr\_time$ and in fact can take on a value smaller than the minimum $inter\_arr\_time$.

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. Also, when there are two flows in the WB topology, T_{narr} could oscillate between 1 and 2. In both cases, we cannot determine the topology with high confidence. Our solution is to use ZigZag during this period due to its relatively consistent performance in both topologies.

7.1. ZBS Algorithm

Based on this idea, we introduce a hybrid algorithm, ZBS (Figure 11), which works as follows:

if ($T_{narr} < 0.95$) use ZigZag;
else if ($T_{narr} < 1.1$) use mBiaz;
else if ($T_{narr} < 1.5$) use ZigZag;
else use Spike;

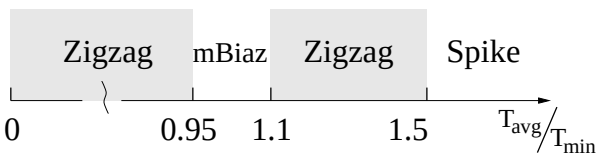

Figure 11: ZBS scheme

Modified Biaz is used for the WLH topology where the wireless link is not shared, to take advantage of its low M_c and M_w (compared to ZigZag), and high throughput. Spike is used in the WB topology where it has the best performance. ZigZag is not used for the WLH topology due to its sensitivity to the operating rate. Instead, we use it for cases where

the underlying topology is not clear, 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 underlying topology at that time. 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 which might otherwise occur from start/stop of short lived traffic streams, occasional severe wireless error, etc.

After the locking period, ZBS applies the above topology determination algorithm, and 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 topology determination algorithm at every packet arrival thereafter, and is free to switch when next indicated.

We derived the three thresholds as follows:

• **0.95:** If the underlying topology is WLH, the smallest T_{narr} happens when all packets are buffered one after the other before the last link and we have severe congestion loss. With a congestion loss rate of 9%, one out of 11 transmitted packets is lost. For the receiver, this means that, out of 10 inter-arrival periods of length T_{min}, one corresponds to 2 packets, and 9 correspond to 1 packet. Thus $T_{narr} = (9 + 0.5)/10 = 0.95$. As 9% congestion loss is very high, this is a generous condition for concluding a WLH topology.

• **1.1:** If the wireless link is fully utilized and not shared, T_{narr} would equal 1. It would be greater than 1 if the link is not fully utilized. Since the upper window of modified Biaz is $1.2 \times T_{min}$ and it classified most congestion loss correctly, we choose a roughly comparable threshold for T_{narr}. A smaller value of 1.1 is used because T_{narr} takes lost packets into account, including those lost to congestion which tend to give lower T_{narr}. Nevertheless, this upper limit for deciding WLH topology might be too restrictive as it allows very little gap between packets arriving at the wireless link buffer. We will experiment with less restrictive values in our future work.

• **1.5:** If the underlying topology is WB with 2 sharing users, the *average* inter-arrival time assuming the shared bottleneck link is fully utilized is $\sum_{i=1}^{\infty}(\frac{1}{2})^i = 2T_{min}$. However, due to congestion loss and variation, often it could be lower than 2. We took the average of 1 and 2.

7.2. ZBS2 Algorithm

ZBS assumes the wireless link is close to fully utilized. This assumption may not always hold. In the WLH topology, because the shared link bandwidth is less than the aggregated total of the wireless links, on average only 86% of the wireless link bandwidth is used. This causes ZBS to misclassify the topology as WB from time to time, and use Spike. As Spike causes high M_c in this topology, ZBS does too.

To solve the problem, notice that in the WLH topology, when its wireless link is underutilized, the variance of ROTT will be small compared to T_{min}. This is because the ROTT is determined by the transmission time of a packet over the shared link, while T_{min} is determined by the transmission time over the wireless last link which has smaller bandwidth. On the other hand, in the WB topology, the variance of ROTT is always comparable to T_{min} because both of them are determined by the bandwidth of the wireless shared link.

Based on this, we modify ZBS when $T_{narr} \geq 1.5$. ZigZag is used in the special case when Spike is not appropriate (causing too much misclassification of congestion loss). We call this ZBS2:

if $(T_{narr} < 0.95)$ use ZigZag;

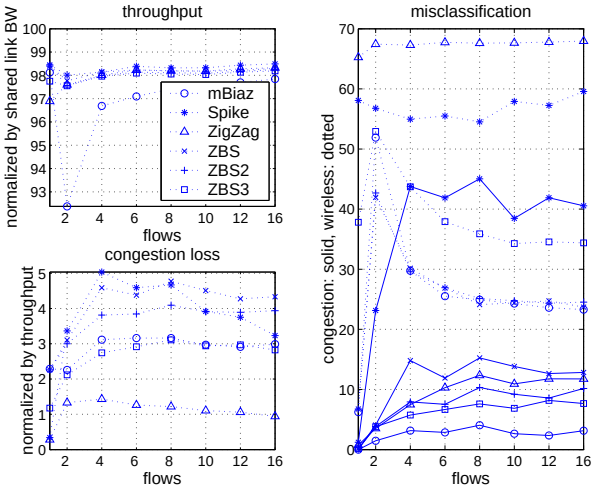

Figure 12: Switching schemes in wireless last hop topology

else if $(T_{narr} < 1.1)$ use mBiaz;
else if $(T_{narr} < 1.5)$ use ZigZag;
else if $(rott_{dev} < 0.5 * T_{min})$ use ZigZag;
else use Spike;

7.3. ZBS3 Algorithm

To obtain lower congestion loss, one could use ZigZag whenever $(rott_{dev} < 0.5 * T_{min})$ regardless of T_{narr}. There is one problem: with only 1 flow in a network with high bottleneck rate, ZigZag would be heavily used. As seen in Section 6.2, ZigZag has low utilization of the bottleneck bandwidth. To solve this problem, Spike is used if the difference between the current ROTT and ROTT$_{min}$ is less than 5% of the minimum inter-arrival time. If the minimum inter-arrival time observed equals the time to transmit one packet over the bottleneck link, this is equivalent to one additional packet buffered at a link with 20 ($\frac{1}{5\%}$) times bandwidth as the bottleneck link. One way to understand this is to think of it as giving ZigZag a push to send it into the normal operating window size quickly whenever the buffer level at the bottleneck is close to empty.

In summary, this new scheme, which we call ZBS3, works as follows:
if $(rott - rott_{min} < 0.05 * T_{min})$ use Spike;
else if $(rott_{dev} < 0.5 * T_{min})$ use ZigZag;
else use ZBS;

Figures 12 and 13 show the performance of the switching schemes. Modified Biaz, Spike and ZigZag are shown for comparison.

7.4. Summary

All three switching schemes achieved high throughput close to omniscient traffic in both topologies and across all numbers of flows.

In the WLH topology, both ZBS2 and ZBS3 have lower M_c and lower M_w than ZigZag; ZBS has higher M_c as discussed previously. The congestion loss of both ZBS2 and ZBS3 is low (3%–4%).

In the WB topology, all 3 switching schemes have M_c close to 0, and M_w similar to that of Spike.

Thus, all 3 switching schemes performed well across different topologies and different numbers of flows. In most cases, ZBS2 and ZBS3 matched or exceeded the performance of the best base algorithm for that scenario.

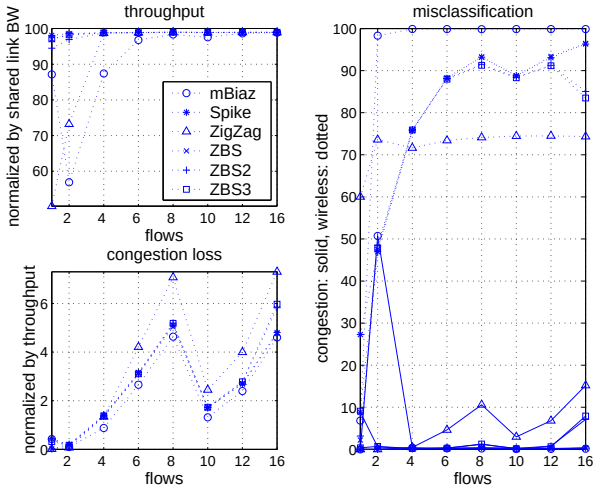

Figure 13: Switching schemes in wireless backbone topology

8. 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 and competition, 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 three hybrid schemes that attempt to choose a different base algorithm best suited to the current topology. The choice is mainly based on the relationship between the inter-arrival time and its minimum. Two of the three hybrid schemes fulfilled our goal: they have excellent performance across both topologies, regardless of the number of competing flows.

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." 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 *Proc. of ACM SIGCOMM '96*, pp. 256–269, Stanford, CA, Aug 1996.
10. 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.
11. Q. Zhao, P. Cosman, and L. Milstein, "Tradeoffs of source coding, channel coding and spreading in CDMA systems," in *Proc. Milcom 2000*, Los Angeles, CA, Oct 2000.