Distinguishing Congestion Losses from Wireless Transmission Losses : A Negative Result *

Saad Biaz† Nitin H. Vaidya
Department of Computer Science
Texas A&M University
College Station, TX 77843-3112, USA
{saadb,vaidya}@cs.tamu.edu

Abstract

TCP is a popular transport protocol used in present-day internet. When packet losses occur, TCP assumes that the packet losses are due to congestion, and responds by reducing its congestion window. *When a TCP connection traverses a wireless link, a significant fraction of packet losses may occur due to transmission errors. TCP responds to such losses also by reducing congestion window. This results in unnecessary degradation in TCP performance.*

We define a class of functions named loss predictors *which may be used by a TCP sender to guess the actual cause of a packet loss (congestion or transmission error) and take appropriate actions. These loss predictors use simple statistics on round-trip times and/or throughput, to determine the cause of a packet loss. We investigate their ability to determine the cause of a packet loss. Unfortunately, our simulation measurements suggest that the three loss predictors do not perform too well.*

1. Introduction

TCP is a popular protocol for reliable data delivery in the internet. TCP is robust in that it can adapt to disparate network conditions [12, 9]. When a packet loss occurs, TCP sender assumes that congestion has occurred in the network, and drastically reduces its *congestion window*. Reducing congestion window temporarily reduces the number of packets sent by the sender, and reduces the throughput. The congestion window can grow again gradually, until another packet loss occurs.

*Research reported is supported in part by the Fulbright Program, National Science Foundation grant CDA 9529442, and Texas Advance Technology Program grants 009741-052-C and 010115-248.

†On leave from the Ecole Superieure de Technologie de Fes (MOROCCO).

TCP makes the implicit assumption that all packet losses are due to congestion. This assumption is not accurate when a TCP connection traverses a wireless link.

Due to increasing acceptance of wireless networking technology, there is considerable interest in using TCP over wireless links (e.g, [2, 8]). Previous work has shown that, unless the TCP protocol is modified, it performs poorly on paths that include a wireless link subject to transmission errors. The reason for this is that a TCP sender activates congestion control mechanisms [9] even if a packet loss is due to wireless transmission errors.

Past proposals for improving performance of TCP over wireless require some cooperation from an intermediate node on the path from the sender to the receiver [2]. For several practical reasons [11], our interest is in mechanisms that impose minimal demands (if any) on any host other than the sender or the receiver. Ideally, it would help if the sender could differentiate between packet losses due to congestion from the packet losses due to wireless transmission errors. Once a sender knows that the packet loss is due to congestion or due to transmission error, it can respond appropriately. One possible approach to distinguish between the two types of packet losses is as follows:

- Use a "loss predictor" that can guess whether a packet transmitted in the near future will be lost due to congestion or transmission error.
- When a packet is lost: If the *loss predictor* predicted that the packet will be lost due to congestion, conclude that the packet loss is indeed due to congestion. Otherwise, conclude that it was due to transmission errors.

The obvious question now is how to design a *loss predictor* that can predict the cause of a future loss. In this paper, we consider three loss predictors derived directly from previously proposed techniques for congestion avoidance. The Congestion Avoidance Techniques (CATs) were proposed to determine when it is appropriate to increase or decrease

TCP congestion window [10, 7, 13]. In the basic TCP, congestion window is decreased only when TCP sender determines that a packet has been lost. Otherwise, the congestion window gradually increases whenever receipt of new data is acknowledged by the receiver. The congestion avoidance techniques [10, 7, 13] monitor the level of congestion in the network, and recommend when the congestion window should be increased or decreased.

The CATs in [10, 7, 13] use simple statistics on observed round-trip times (RTT) and/or observed throughput of a TCP connection. An objective of this paper is to investigate the ability of *loss predictors*, based on these CATs, to determine the cause of a *packet loss*. The paper also evaluates how the loss predictors react to changes in several network parameters, such as link bandwidth and packet loss rates.

Rest of this paper is organized as follows. Section 2 describes the three congestion avoidance techniques (CATs) used in this paper. Section 3 describes how loss predictors are derived from the CATs. Performance parameters of interest are defined in Section 4. Simulation model and simulation results are discussed in Section 5. Conclusions are presented in Section 6.

2. Congestion Avoidance Techniques

To describe the congestion avoidance techniques, we first need to introduce some terminology.

2.1. Terminology and Notations

- Sender's Congestion Window W: The congestion window determines the maximum amount of unacknowledged data sent by the TCP sender.
- i-th monitored packet P_i: At any time, one packet sent by the sender is monitored. For the i-th monitored packet P_i, we define three parameters below, to be used in implementing the CATs.
- Window size W_i for the i-th monitored packet: W_i is the amount of data transmitted (including the monitored packet) during the interval from the time when the monitored packet is transmitted, until when an acknowledgement for the monitored packet is received.
- Round-trip time RTT_i for i-th monitored packet : *Round-trip time* RTT_i for the i-th monitored packet P_i is the duration from the time when P_i is transmitted, until the time when an acknowledgement for P_i is received by the sender.
- Throughput T_i for the i-th monitored packet : For the i-th monitored packet P_i, the window size is W_i, and round-trip time is RTT_i. In this case, throughput T_i is defined as $T_i = W_i/RTT_i$.

The congestion avoidance techniques considered here are motivated by the following expectation of network behavior [10]. As illustrated in Figure 1, when network load is small, increasing the load should result in a comparable increase in network throughput with only a small increase in round-trip times (RTT). At some point, when the load is large enough, increasing the load further should result in a smaller increase in throughput, and a larger increase in round-trip times (this occurs at the "knee" of the load-throughput curve). If the load is increased further, at some point, the network throughput should drop sharply, while round-trip times should become extremely large.

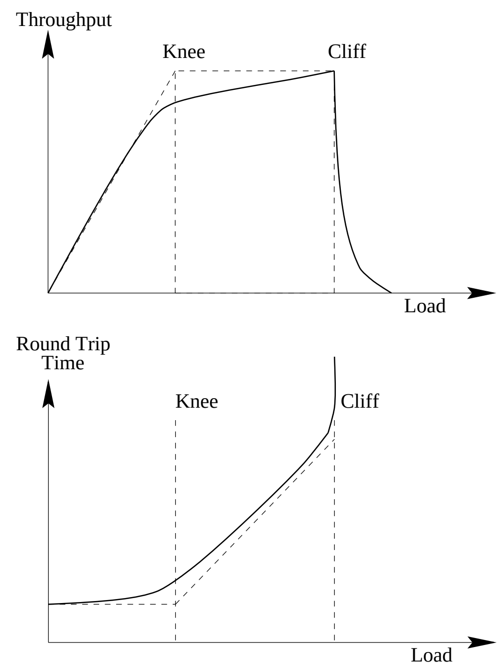

Figure 1. Throughput and RTT versus network load [10]

The three CATs considered in this paper are summarized below. The CATs are implicitly based on the notion that there will be some *response* from the network to a congestion window size change for a TCP connection.

2.2. Congestion Avoidance Technique 1:

TCP-Vegas [7] requires a TCP sender to keep track of the $BaseRTT$, defined as the minimum of all RTTs measured during the TCP connection. When acknowledgement for the i-th monitored packet is received, the sender calculates the *expected* throughput as,

$$\text{Expected Throughput} = \frac{W_i}{BaseRTT}$$

The actual throughput T_i (as defined earlier), is calculated as $\frac{W_i}{RTT_i}$. Then the difference D is calculated as, $D =$ expected throughput $-$ actual throughput $= \frac{W_i}{BaseRTT} - \frac{W_i}{RTT_i}$. Reference [7] expresses this difference D in terms of *extra packets* in the network, by multiplying D by $BaseRTT$. We define f_{Vegas} as,

$$f_{Vegas} = BaseRTT \times D = W_i \left(1 - \frac{BaseRTT}{RTT_i}\right)$$

f_{Vegas} is compared to two thresholds α and β, where $\alpha < \beta$. If $f_{Vegas} < \alpha$ (*resp.* $f_{Vegas} > \beta$), then this congestion avoidance technique suggests that the window size be increased (*resp.* decreased).

2.3. Congestion Avoidance Technique 2:

Wang and Crowcroft [13] proposed a congestion avoidance technique based on the *Normalized Throughput Gradient* (f_{NTG}). To calculate f_{NTG}, we need to define *throughput gradient* TG_i for the i-th monitored packet P_i:

$$TG_i = \frac{T_i - T_{i-1}}{W_i - W_{i-1}}$$

This congestion avoidance technique evaluates the *normalized* throughput gradient f_{NTG} as TG_i/TG_1, when acknowledgement for packet P_i is received. TG_1 is defined as follows : $TG_1 = (T_1 - T_0)/(W_1 - W_0) = 1/RTT_1$, as $W_1 = 1$ packet, $W_0 = 0$, $T_0 = 0$ and $T_1 = W_1/RTT_1 = 1/RTT_1$. Therefore, $f_{NTG} = \frac{TG_i}{1/RTT_1}$. Substituting above expression for TG_i and simplifying, we get

$$f_{NTG} = \frac{RTT_1}{W_i - W_{i-1}} \left(\frac{W_i}{RTT_i} - \frac{W_{i-1}}{RTT_{i-1}}\right)$$

If $f_{NTG} < 1/2$, then this congestion avoidance technique suggests that the congestion window size be decreased, else it suggests that the window size be increased.

2.4. Congestion Avoidance Technique 3:

Jain proposed a congestion avoidance technique based on *Normalized Delay Gradient* [10]. Our implementation of this heuristic evaluates f_{NDG} as follows, when acknowledgement for the i-th monitored packet is received:

$$f_{NDG} = \frac{(RTT_i - RTT_{i-1})}{(RTT_i + RTT_{i-1})} \frac{(W_i + W_{i-1})}{(W_i - W_{i-1})}$$

If $f_{NDG} > 0$, this congestion avoidance technique suggests that congestion window size should be decreased, otherwise it suggests that the window size be increased.

3. Loss Predictors

In this section, we describe how loss predictors are obtained using the CATs described above. In general, whenever a CAT suggests that congestion window be decreased, the corresponding loss predictor would predict that next packet loss will be due to congestion. The motivation behind our definition of the loss predictors is as follows. A good congestion avoidance technique should suggest that congestion window be increased only if congestion is not very likely to occur in the near future. Thus, if a loss occurs when the congestion avoidance technique is recommending increasing window size, it may be reasonable to assume that the loss is due to transmission errors (and vice versa).

Loss predictor *Vegas*: Loss predictor Vegas is obtained using congestion avoidance technique 1 described in section 2.2. Whenever acknowledgement for a monitored packet is received, the loss predictor calculates the quantity named f_{Vegas}, as defined in Section 2.2. If $f_{Vegas} > 1$ (*resp.* $f_{Vegas} \leq 1$), then the cause of the next loss will be assumed to be congestion (*resp.* transmission errors).

Loss predictor NTG: Loss predictor NTG is obtained using congestion avoidance technique 2 described in Section 2.3. Whenever acknowledgement for a monitored packet is received, the loss predictor calculates the quantity named f_{NTG}, as defined in Section 2.3. If $f_{NTG} < 1/2$ (*resp.* $f_{NTG} \geq 1/2$), then the cause of next packet loss will be assumed to be congestion (*resp.* transmission errors).

Loss predictor NDG: Loss predictor NDG is obtained using the congestion avoidance technique 3 described in Section 2.4. Whenever acknowledgement for a monitored packet is received, the loss predictor calculates the quantity named f_{NDG}, as defined in Section 2.4. If $f_{NDG} > 0$ (*resp.* $f_{NDG} \leq 0$), then the cause of next packet loss will be assumed to be congestion (*resp.* transmission errors).

4. Performance Metrics

To characterize the ability to distinguish congestion losses from wireless transmission error losses, we define four metrics for each loss predictor.

- Frequency of Congestion Loss Prediction (FCP): FCP is obtained by dividing the number of times the loss predictor predicts that the next loss will be due to congestion, by the total number of times the predictor (i.e., value f_{Vegas}, f_{NTG} or f_{NDG}) is evaluated during the TCP connection.

- Frequency of Wireless Loss Prediction (FWP): FWP is obtained by dividing the number of times the loss predictor predicts that the next loss will be due to wireless transmission error, by the total number of times the predictor (i.e., value f_{Vegas}, f_{NTG} or f_{NDG}) is evaluated during the TCP connection. It follows that $FWP = 1 - FCP$.

- Accuracy of Congestion Loss Prediction A_c: A_c is the fraction of packet losses due to congestion that are correctly diagnosed. A congestion loss is correctly diagnosed if the latest prediction before this loss was a *congestion loss*.

- Accuracy of Wireless Loss Prediction A_w: A_w is the fraction of packet losses due to wireless transmission error losses that are correctly diagnosed.

Now, consider a "random coin tossing" loss predictor that uses probabilistic coin tossing to determine whether to predict congestion loss or wireless loss. Suppose that it predicts that next packet loss will be *congestion loss* with probability p. Clearly, in this case, $FCP = p$ and $FWP = 1 - p$. Also, as the prediction made by the predictor is independent of network conditions, in this case, $A_c = p$ and $A_w = 1 - p$. Thus, a simple coin tossing scheme can yield $A_c = FCP = p$ and $A_w = FWP = 1 - p$ for any desired value of p. Choosing high p will result in high A_c, but low A_w, and vice versa.

5. Simulations

5.1. Simulation Model and Methodology

We use the network simulator *ns-2* (version 2.1b1) [1] from Berkeley. The system model used for simulations is illustrated in Figure 2. We have a TCP connection from a source CS to a sink CK. We use the *Reno* agent from *ns-2* for this connection. This connection shares the link $R_1 \longleftrightarrow R_2$ with a cross traffic issued by a $Traffic/Expoo$ [1] agent from RS to sink RK. The $Traffic/Expoo$ agent from *ns-2* [1] is a constant-bit rate (CBR) source with idle time and busy time exponentially distributed with mean 0.1 s. UDP is the transport protocol used for this source.

All the links in Figure 2 are labeled with a *(bandwidth, propagation delay)* pair. Note that propagation delay does *not* include transmission time or queueing delays. The links $R_2 \longleftrightarrow CK$ and $R_2 \longleftrightarrow RK$ are assumed to have a negligible propagation delay. In our simulations, this propagation delay is assumed to be 0. The link $R_2 \longrightarrow CK$ is a wireless link with transmission loss rate r_w (i.e, fraction r_w packets are lost due to transmission errors). All other links are error-free. We simulate the network with different values for bandwidth bw and delay δ (please refer Figure 2). In different simulations, bw takes the values 100 Kbits/s, 500 Kbits/s, 1000 Kbits/s, 1500 Kbits/s and 2 Mbits/s, and δ takes values 3 ms, 5 ms, 8 ms, 13 ms, 18 ms, 23 ms, 38 ms, 50 ms, and 75 ms.

Router R_1 has an output queue (towards R_2) whose size is limited to qs packets. qs takes the values 5, 10, or 15 in our simulations. All other queues at the two routers are unbounded (infinite). Obviously, the potential bottleneck here is the link $R_1 \longrightarrow R_2$.

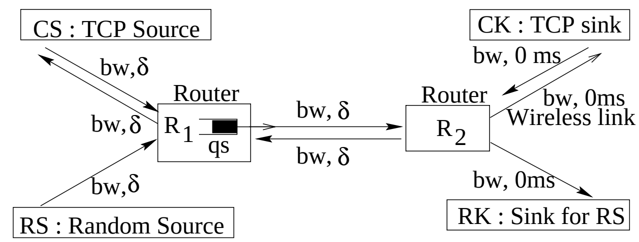

Figure 2. ns **network topology**

Let T_p denote the round-trip propagation delay for the TCP connection (i.e., from CS to CK and back to CS). Then, with the values of δ used in our simulations, T_p varies in the range 12 ms to 300 ms.

We denote the congestion loss rate for the TCP connection as r_c. r_c is measured as a fraction (or percentage) of packets lost due to congestion. In our simulations, for each set of parameters (T_p, bw, qs), the rate of the constant-bit rate source RS ($Traffic/Expoo$ agent) is adjusted to produce a desired value of r_c. Then, we make 10 additional TCP transfers, which last between 200 and 4000 seconds depending on bw, and collect statistics. Each transfer starts after a random warm-up period larger that 100 seconds.

For the measurements, we monitor one packet per window : we log its round trip time (RTT_i) and the number of packets (W_i) sent between its transmission and its acknowledgement. The congestion window size is limited to 32 packets. From the logged information, we compute the values f_{Vegas}, f_{NTG}, and f_{NDG}, as defined in section 2. Note that the three values are computed from the same set of data. Using these values, the performance metrics (FCP, FWP, A_c and A_w) for the loss predictors can be determined. For the ten transfers, the standard deviation on the performance metrics for each loss predictor is less than 0.05.

For each loss predictor, we perform 4 sets of experiments. In each set, one of the four parameters, namely, T_p (or δ), bw, qs and r_c, is varied, while the other three parameters are held constant. Thus, each set of experiments helps us to determine the variations in FCP, A_c, FWP, and A_w as a function of each of the four parameters. The following values for the parameters are used:

- Extensive simulations were done with $r_w = 1\%$, 3%

and 5%. The results are similar for these values, therefore, we only present the results for $r_w = 1\%$.

- Congestion loss rate (r_c) from 1% to 10% (congestion loss rate specifies the fraction of packets lost by the TCP connection at router R_1)

  When r_c is held constant for some plots presented in this paper, we hold it constant at 3%, because the trends observed are representative of what we observed for other r_c values.

- Round-trip propagation time T_p in the range 10 ms to 300 ms. Note that T_p does not include the queueing and transmission delays.

  When T_p is held constant for some plots, we hold it constant at 32 ms because it represents a typical value of round trip propagation time on WANs.

- Bandwidth bw from 100 Kbits/s to 2 Mbits/s.

  When bw is held constant for some plots presented in this paper, we hold it at 1.5 Mbits/s (T1 bandwidth).

- Queue size limit qs at router R_1 from 5 to 15 packets (packet size is 1000 bytes).

  When qs is held constant for some plots in this paper, we hold it at $qs = 5$ because the trends are similar for the other values of qs.

5.2. Simulation Results

Objective of our simulation experiments was two-fold: (a) determine the magnitudes of frequencies (FCP and FWP) and accuracies (A_c and A_w) achieved using the loss predictors, and (b) determine the variations in these metrics as a function of network parameters (such as bw and r_c). The simulation results are summarized below. We present graphs showing only some of our simulation results. However, the conclusions reported here are drawn from a larger set of simulations [4].

5.3. Loss Predictor *Vegas*

In this section, we summarize our observations for the loss predictor Vegas, and attempt to provide intuitive (or mathematical) explanations. First we discuss variation trends for FCP, and then the trends for A_c and A_w.

Recall that if $f_{Vegas} > 1$, then the Vegas predictor predicts congestion losses. The probability that f_{Vegas} will be greater than 1 decreases if f_{Vegas} decreases. Thus, if f_{Vegas} decreases, FCP for the predictor will decrease. This relationship will be used in our explanations below.

Variations in Frequency of Congestion Loss Prediction:

- FCP for Vegas predictor decreases when T_p is increased, while holding bw, qs, r_c and r_w constant. Refer Figure 3 for an illustration. In Figure 3, the horizontal axis corresponds to T_p – the values listed in the parenthesis along the horizontal axis are held constant for all simulations reported in this figure.

  This observation is supported by a simple mathematical analysis. Note that RTT_i can be expressed as $RTT_i = T_p + t_i$, where t_i is a random variable depending on the transmission time, the queueing delay and the processing time for the monitored packet. Similarly, $BaseRTT$ can be expressed as $BaseRTT = T_p + t_{Base}$ where t_{Base} is a random variable similar to t_i with $t_{Base} \leq t_i$ ($BaseRTT$ is the smallest round trip time experienced by the connection.) Then, $f_{Vegas} = W_i\ (1 - \frac{T_p + t_{Base}}{T_p + t_i})$. Thus, $\frac{\delta(Vegas)}{\delta T_p} = W_i \left(\frac{t_{Base} - t_i}{(T_p + t_i)^2}\right)$. Since $t_i \geq t_{Base}$, $\frac{\delta(f_{Vegas})}{\delta T_p}$ is usually negative. This means that the value of f_{Vegas} decreases when T_p is increased. Therefore, as T_p increases, FCP for Vegas predictor should decrease.

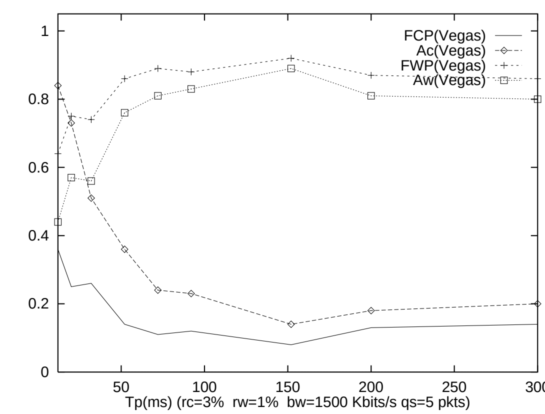

Figure 3. FCP, A_c and A_w versus T_p

- FCP decreases when bw is increased, keeping T_p, qs, r_c, and r_w constant, as illustrated in Figure 4.

  Similar to the above derivation, we provide a mathematical explanation for this observation. Let us express RTT_i as $RTT_i = BaseRTT + dq_i$ where dq_i is the extra queueing delay for the i-th monitored packet, as compared to $BaseRTT$ (assuming that the round trip time variation is due only to the queueing delays). Thus, $f_{Vegas} = W_i \times (1 - \frac{BaseRTT}{BaseRTT + dq_i})$.

  Since $\frac{\delta(f_{Vegas})}{\delta dq_i} = \left(\frac{W_i\ BaseRTT}{(BaseRTT + dq_i)^2}\right) > 0$, the value f_{Vegas} increases with increasing queueing delay dq_i. From queueing theory, it follows that, queueing delay variations decrease when the service rate increases,

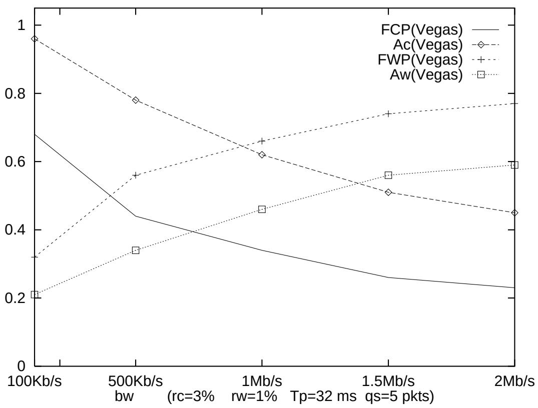

Figure 4. FCP, A_c and A_w versus bw

i.e., in this case, when bw increases. Therefore, when bandwidth bw is increased, queueing delay dq will decrease, and consequently f_{Vegas} will decrease (as $\frac{\delta(f_{Vegas})}{\delta dq_i} > 0$). Finally, when f_{Vegas} decreases, the FCP for the Vegas predictor also decreases.

- FCP increases when qs is increased, keeping bw, T_p, r_c and r_w constant, as illustrated in Figure 5.

  As qs increases, with the congestion loss rate r_c held constant , the average queueing delay variations increase. We showed above that the value f_{Vegas} increases with larger queueing delays variations. Therefore, f_{Vegas} increases with increasing qs. Thus, FCP will increase with increasing qs.

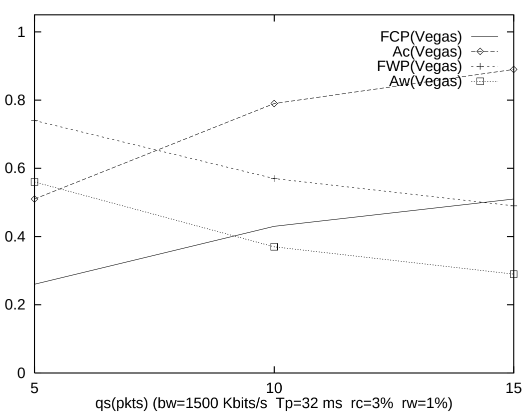

Figure 5. FCP, A_c and A_w versus qs

- FCP decreases when r_c is increased, keeping bw, T_p, qs, and r_w constant, as illustrated in Figure 6.

  It is somewhat counter-intuitive that FCP decreases with increasing congestion loss rate. Note that, $\frac{\delta f_{Vegas}}{\delta W_i} = 1 - \frac{BaseRTT}{RTT_i}$. Since $BaseRTT \leq RTT_i$,

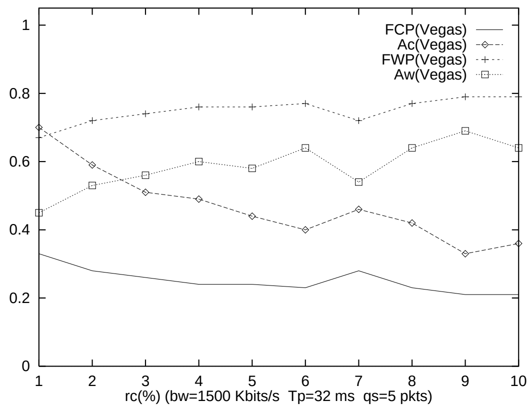

Figure 6. FCP, A_c and A_w versus r_c

  then, $\frac{\delta f_{Vegas}}{\delta W_i}$ is typically positive. Thus, if W_i decreases, then f_{Vegas} will also decrease. Now, note that, as r_c increases, the average congestion window size, and thus W_i, decreases. Therefore, with increasing r_c, f_{Vegas} will decrease, consequently, the FCP for the Vegas predictor will also decrease.

Accuracy of Congestion Loss Prediction A_c: Accuracy of congestion loss prediction A_c for $Vegas$ usually follows FCP's trends. Typically, A_c is higher than FCP. The difference between A_c and FCP is significant with certain parameter values (for instance, small T_p ($<$ 32 ms), low congestion loss rate (1-2%) and small queue size ($q_s = 5$ packets). Thus, for congestion loss prediction, the Vegas predictor is capable of performing better than a random coin tossing predictor under certain circumstances (as discussed in Section 3, for the random predictor $FCP = A_c$).

The absolute value of A_c varies a lot depending on the network parameters. Accuracy A_c in the range of 0.5 to 0.8 was observed in a large number of cases. As noted in the previous section, Vegas predictor (and, also the other loss predictors) determine their predictions based on the network's *response* to congestion window size change for the TCP connection. Typically, a single TCP connection constitutes a small fraction of the total network traffic. Thus, the observed network response also depends on other traffic, and not just on window size changes for a single TCP connection. Therefore, accuracy of congestion loss prediction tends to be poorer than one may expect. (This same reason causes other predictors to perform below expectation as well.)

Whenever sender mistakes a congestion loss as a transmission error loss, it would not take congestion control actions. Therefore, low A_c may be detrimental to overall network performance. Thus, design of loss predictors that can consistently yield high A_c is of interest.

Accuracy of Wireless Loss Prediction A_w: Wireless transmission losses occur independently of the network conditions. Therefore, it is not reasonable to expect RTT and throughput estimates to yield any indication of an impending transmission loss. However, such estimates may provide an indication of an impending congestion loss. Therefore, in our loss predictors, a lack of an indication of congestion loss is used as an "indication" of a wireless loss. Therefore, A_w (and FWP) follow trends that are opposite of A_c and FCP (that is, when A_c increases, A_w decreases).

The Vegas predictor does not perform very well at diagnosing wireless losses, when compared to a random coin tossing predictor. In general, $A_w < FWP$ for the Vegas predictor, whereas $A_w = FWP$ for a random predictor.

It is important to emphasize that a good loss predictor needs to be able to diagnose both types of packet losses reasonably well. Ideally, we would like to have high A_c and A_w both. However, if a compromise is to be made, a high A_c and moderate A_w may be acceptable. Low or moderate A_w may often result in erroneously identifying wireless losses as congestion losses. This would affect performance of the TCP connection using this loss predictor, but it cannot adversely affect performance of other network traffic (unlike a low A_c).

5.4. Loss Predictor NTG

This section presents the observations from simulation results obtained for the NTG predictor. Recall that, if $f_{NTG} < \frac{1}{2}$, then the NTG predictor predicts congestion. Therefore, as f_{NTG} increases, FCP decreases. This relationship will be used in the explanations below.

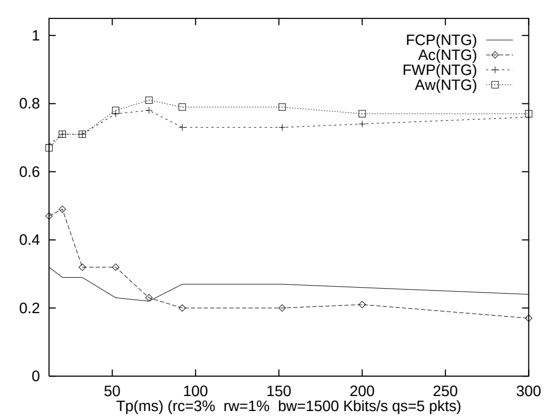

Figure 7. FCP, A_c and A_w versus T_p

Variations in Frequency of Congestion Loss Prediction:

- FCP decreases when T_p is increased, while holding bw, qs, r_c, and r_w constant. Refer Figure 7.

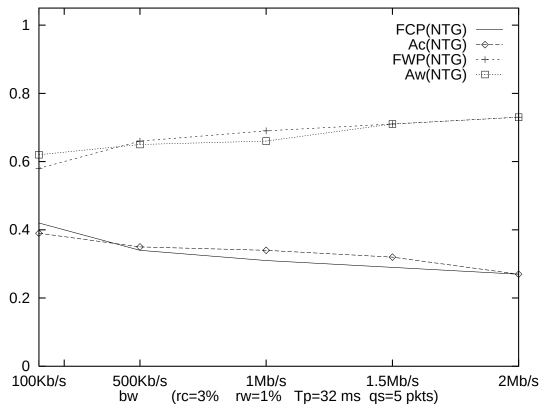

Figure 8. FCP, A_c and A_w versus bw

To support this observation, we show that f_{NTG} is increasing with increasing T_p. We can write $RTT_{i-1} = T_p + d_{i-1}$ and $RTT_i = T_p + d_i$ where d_{i-1} and d_i are positive random variables depending on the transmission time, the queueing delays and the processing time for i-th and $i+1$-th monitored packets. We can then rewrite f_{NTG} as :

$$f_{NTG} = \frac{T_p + d_1}{W_i - W_{i-1}} \left(\frac{W_i}{T_p + d_i} - \frac{W_{i-1}}{T_p + d_{i-1}} \right)$$

Now note that, since we are using TCP-Reno in our simulations, most of the time the TCP connection is in congestion avoidance phase. Therefore, very often, $W_i - W_{i-1} = 1$ packet. Assuming this, it can be shown that, if $d_{i-1} \geq d_i$ then $f_{NTG} \geq \frac{1}{2}$, provided $max(\frac{T_p+d_1}{T_p+d_i}, \frac{T_p+d_1}{T_p+d_{i-1}}) \geq \frac{1}{2}$. The condition $max(\frac{T_p+d_1}{T_p+d_i}, \frac{T_p+d_1}{T_p+d_{i-1}}) \geq \frac{1}{2}$ means that the round-trip time for any monitored packet is less than twice the round trip time for the first packet. This is in general true, unless the propagation time is very small and the queueing delay variations very large. In conclusion, if $d_{i-1} \geq d_i$ then NTG predictor will typically not predict congestion.

As noted before, typically $W_i - W_{i-1} > 0$. It can be shown that, if $d_{i-1} < d_i$, $\frac{\delta f_{NTG}}{\delta T_p} > 0$ provided that $(T_p + d_1)^2 \geq (d_{i-1} - d_1).(d_i - d_1)$. This last condition means that the variations in the delays should not exceed the absolute value of the first round trip time, which is in general true. Therefore, f_{NTG} typically increases with increasing propagation time T_p. Therefore, FCP decreases when T_p increases.

- FCP decreases when bw is increased, keeping T_p, qs, r_c, and r_w constant, as illustrated in Figure 8.

Similar to the above derivation, we provide a mathematical explanation for this observation. We can express RTT_i as $RTT_i = RTT_{i-1} + d_q$ where d_q is the difference in the queueing delay between the two monitored packets P_i and P_{i-1}. Note that d_q can be positive or negative. It can be shown that $\frac{\delta f_{NTG}}{\delta d_q} = -\frac{RTT_1 \; W_i}{(W_i - W_{i-1}) \; (RTT_i + d_q)^2}$. As, for TCP-Reno, typically $W_i > W_{i-1}$, we have $\frac{\delta f_{NTG}}{\delta d_q} < 0$. Thus, f_{NTG} decreases with increasing d_q. Therefore, FCP increases when d_q increases, and vice-versa. Now, d_q decreases when the bandwidth bw increases (because queueing delay magnitudes and variations decrease when service rate increases). Hence, FCP decreases when bw increases.

- FCP increases when qs is increased, keeping bw, T_p and r_c constant, as illustrated in Figure 9.

  For a constant loss rate, as qs increases, the amount of random source's traffic in the queue ahead of a TCP packet can increase. Therefore, queueing delay variation for TCP packets is larger. We showed above that f_{NTG} decreases with increasing queueing delay variations. Thus, FCP increases with increasing qs.

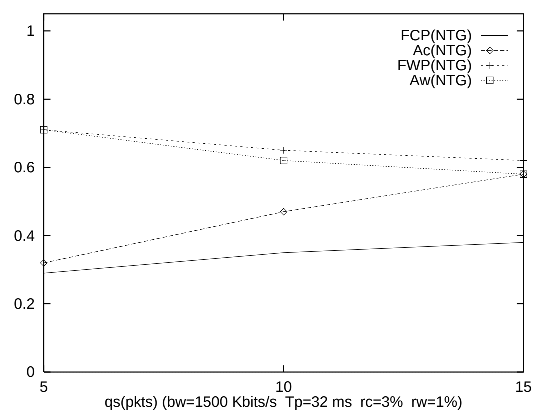

Figure 9. FCP, A_c and A_w versus qs

- FCP does not exhibit any trend when r_c is increased, keeping bw, T_p, r_w, and qs constant, as illustrated in Figure 10.

  As for the $Vegas$ predictor, the trend of FCP for NTG predictor when congestion loss rate r_c is varied is related to the average congestion window size. However, unlike Vegas predictor, in this case, the sign of $\frac{\delta f_{NTG}}{\delta W_i}$ depends on the sign of $RTT_i - RTT_{i-1}$. Now, RTT_i and RTT_{i-1} correspond to window size W_i and W_{i-1}, where typically $W_i > W_{i-1}$. When bandwidth bw is not small, the RTT is essentially independent of the window size. Therefore, the sign of $RTT_i - RTT_{i-1}$ does not depend of the window size. This, in turn, implies that the f_{NTG} is independent of W_i and loss rate r_c, when bw is high.

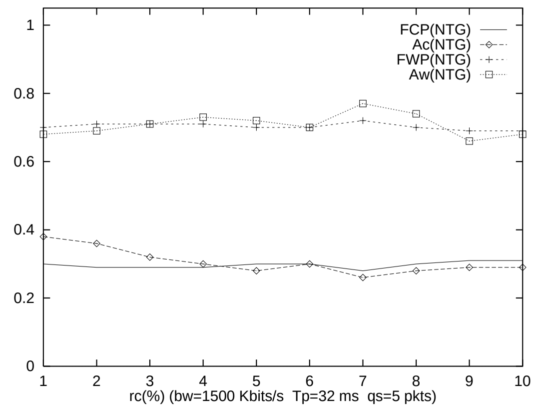

Figure 10. FCP, A_c and A_w versus r_c

Accuracy of Congestion Loss Prediction A_c Accuracy of congestion loss prediction A_c follows closely FCP in most cases. In general, for NTG, A_c tends to be smaller and closer to FCP, as compared to the case of Vegas predictor. Thus, NTG behaves more like a random coin tossing predictor – this implies that NTG is unable to capture the indications of an impending congestion (if it exists) from the RTT or throughput statistics. Based on our simulations, it appears that NTG is a poor loss predictor.

Accuracy of Wireless Loss Prediction A_w: A_w trends for NTG are similar to those for the $Vegas$ predictor, except that A_w follows FWP much more closely for NTG. This confirms that NTG performs similar to a random coin tossing predictor.

5.5. Loss Predictor NDG

Recall that if f_{NDG} is positive then the NDG predictor predicts congestion. Also, in our simulations, the agent *TCP-Reno* uses the Jacobson congestion avoidance algorithms. Thus, often $W_{i-1} < W_i$ and the sign of f_{NDG} depends only on the sign of $(RTT_i - RTT_{i-1})$.

Variations in Frequency of Congestion Loss Prediction: The simulation results indicate that FCP, A_c, FWP and A_w for the NDG predictor do not show any trends (increasing or decreasing) as a function of the four parameters T_p, bw, qs and r_c. Now we attempt to provide intuitive explanation for this.

- Variation of FCP when T_p is increased, while holding bw, qs, r_c and r_w constant. Refer Figure 11.

  We can write RTT_i as $RTT_i = T_p + t_i$ where t_i is a random variable. Therefore, the sign of $(RTT_i - RTT_{i-1})$ is the sign of $(t_i - t_{i-1})$, independent of T_p. Thus, FCP does not depend on T_p.

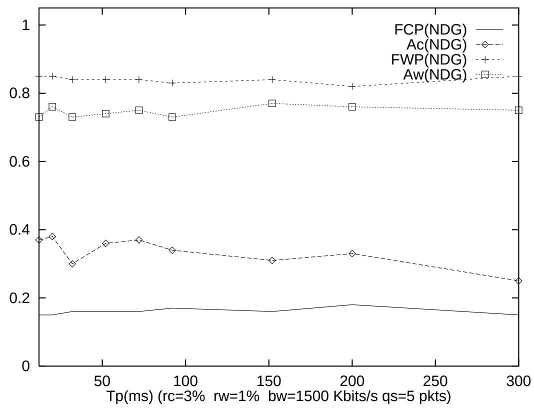

Figure 11. FCP, A_c and A_w versus T_p

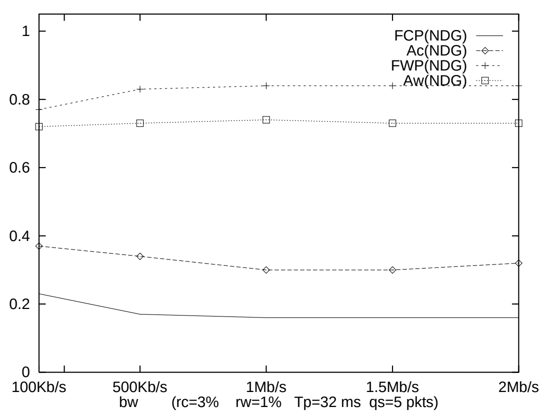

Figure 12. FCP, A_c and A_w versus bw

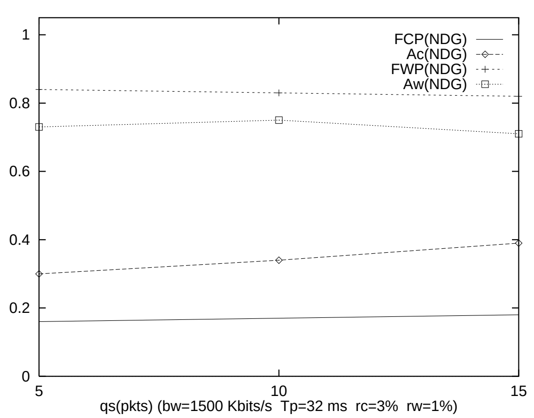

Figure 13. FCP, A_c and A_w versus qs

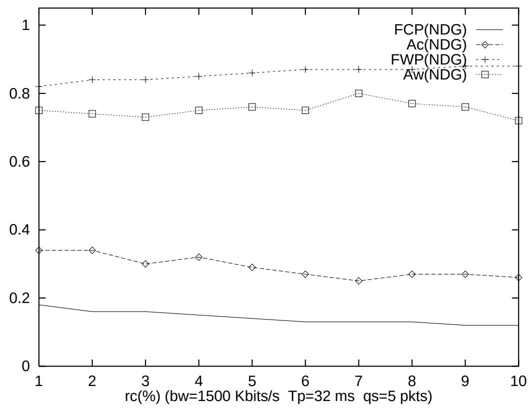

Figure 14. FCP, A_c and A_w versus r_c

- Variation of FCP when bw is increased, while holding T_p, qs, r_c and r_w constant. Refer Figure 12.

  When bandwidth bw is small, a larger window size typically results in a greater round-trip time. However, at higher bandwidths the round-trip time tends to be independent of the window size. Therefore, with low bandwidth bw, it is more likely that an increase in congestion window size induces a larger round trip time. In this case $f_{NDG} > 0$. In short, when bw is low, NDG will predict *congestion loss* more often. Therefore, at low bandwidths, FCP decreases when bw increases. On the other hand, when bw is reasonably high, FCP becomes independent of bw. In Figure 12, FCP decreases slightly initially, but is essentially constant for larger bw.

- Variation of FCP when qs is increased, while holding T_p, bw, r_c, and r_w constant. Refer Figure 13.

  Queue size qs has an impact on the magnitude of queueing delays. Since f_{NDG} depends on the sign of difference between queueing delays for different packets, but not on the magnitude of the difference, f_{NDG} is independent of qs.

- Variation of FCP when r_c is increased, while holding T_p, bw, r_w, and qs constant. Refer Figure 14.

  The congestion loss rate affects size of the TCP congestion window. Although f_{NDG} depends on the difference $W_i - W_{i-1}$, it does not depend on the *absolute* values of the congestion window size. So, FCP is independent of the loss rate.

Accuracies A_c and A_w: A_c curves usually tracks FCP curves, and A_w curves closely follows FWP curves. A_c and FCP values for NDG is typically smaller than the Vegas loss predictor.

6. Discussion and Conclusion

Simulation results indicate that, the loss predictors cannot *always* perform better than a random coin predictor. Under some network conditions, Vegas is able to perform better than a random predictor, when A_c is considered. In general, our results suggest that Vegas is a better loss predictor than NDG and NTG. However, all three predictors do perform like a random predictor under some circumstances. It is useful to provide an intuitive explanation of this result. A predictor will accurately diagnose congestion losses only if the following *qualitative* conditions are fulfilled: (a) Congestion losses are preceded by a "long" queue build-up at some router, (b) A queue build-up typically results in congestion losses, and (c) The loss predictor correctly senses "serious" queue build-up. Condition (a) means that the interval of time between the instant when a router queue starts to build up and the instant when the queue overflows must be long enough. Otherwise, congestion losses will occur before the predictor has a chance to detect congestion. To fulfill condition (a), favorable values of network parameters are as follows: round-trip time small, router queue size large, and input bandwidth to the bottleneck small. Condition (b) above will tend to be satisfied if queue size is small. We can see that conditions (a) and (b) have contradictory requirements on the queue size.

As noted earlier, the three predictors are designed based on the congestion avoidance techniques. These congestion avoidance techniques are motivated by the expectation that a variation in the congestion window size will result in a "response" from the network which reflects the true state of the network. Unfortunately, the traffic of one connection is, in general, a small fraction of the overall traffic. Therefore, the network response is often independent of one TCP connection's action. This suggests that the three predictors cannot correctly detect queue build-up, and hence cannot diagnose congestion losses accurately. Incidentally, based on a very different type of experiment, Bolot [6] has observed that congestion losses appear to be random. We believe that our experiments [3] support Bolot's observation, and provide additional insight into packet losses due to congestion and wireless errors.

We must also note that the three congestion avoidance techniques were not designed as "loss predictors". These congestion avoidance techniques were designed to let the sender operate at the knee of the throughput-delay curve [10]. While it is not a surprise that these predictors are unable to perfectly diagnose cause of packet losses, it is indeed a surprise that they often behave similar to a random coin tossing predictor. The three congestion avoidance techniques seem inadequate for the design of good loss predictors.

Based on the results obtained for Vegas, it appears that RTT and throughput statistics hold some information that correlates to the cause of packet losses. However, it is not yet clear if there is sufficient correlation to develop loss predictors that can yield high A_c and A_w both. Future work on this topic would investigate design of better loss predictors. The loss predictors presented in this paper are sender-based in that the TCP sender attempts to distinguish between the type of packet losses. At present, we are studying a receiver-based technique [5]. This technique, implemented at the receiver, uses statistics on the inter-arrival times of the packets. Preliminary results show that this technique may be useful if the last link on the TCP path is wireless, and is the bottleneck.

References

[1] VINT project U.C. berkeley/LBNL, ns2:network simulator. http://www-mash.cs.berkeley.edu/ns/.

[2] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz. A comparison of mechanisms for improving TCP performance over wireless links. In *ACM SIGCOMM'96*, Aug. 1996.

[3] S. Biaz and N. Vaidya. Using end-to-end statistics to distinguish congestion and corruption losses : A negative result. Technical Report (draft version), CS Dept., Texas A&M University, Aug. 1997.

[4] S. Biaz and N. Vaidya. Sender-based heuristics for distinguishing congestion losses from wireless transmission losses. Technical Report TR98-013, CS Dept., Texas A&M University, June 1998.

[5] S. Biaz and N. H. Vaidya. Discriminating congestion losses from wireless losses using inter-arrival times at the receiver. Technical Report 98-014, CS Dept., Texas A&M University, June 1998. Revised August 1998.

[6] J. Bolot. Characterizing end-to-end packet delay and loss in the internet. *Journal of High-Speed Networks*, 2(3):289–298, Sept. 1993.

[7] L. Brakmo and S. O'Malley. TCP-vegas : New techniques for congestion detection and avoidance. In *ACM SIGCOMM'94*, pages 24–35, Oct. 1994.

[8] R. Caceres and L. Iftode. Improving the performance of reliable transport protocols in mobile computing environments. *IEEE JSAC Special issue on Mobile Computing Networks*, 13(5), June 1995.

[9] V. Jacobson. Congestion avoidance and control. In *ACM SIGCOMM'88*, pages 314–329, Aug. 1988.

[10] R. Jain. A delay-based approach for congestion avoidance in interconnected heterogeneous computer networks. *ACM CCR*, 19:56–71, 1989.

[11] M. Mehta. Improving performance of TCP over wireless networks. Master's thesis, Texas A&M University, Aug. 1998.

[12] J. Postel. Transmission control protocol, Sept. 1988. RFC 793.

[13] Z. Wang and J. Crowcroft. A new congestion control scheme : Slow start and search (tri-s). *ACM CCR*, 21:32–43, Jan. 1991.