Performance of TCP Congestion Predictors as Loss Predictors

Saad Biaz       Nitin H. Vaidya

Department of Computer Science
Texas A&M University
College Station, TX 77843-3112, USA
E-mail: {saadb,vaidya}@cs.tamu.edu
Web: http://www.cs.tamu.edu/faculty/vaidya/mobile.html

Technical Report 98-007

February 9, 1998 †

Abstract

In the context of TCP, several researchers have proposed heuristics to detect or predict congestion in the network. In this paper, the term congestion predictors refers to such heuristics. Past proposals require TCP sender to reduce its window size when congestion is detected or predicted (otherwise, the heuristic may dictate that the sender window be held constant or increased). The proposed heuristics to detect/predict congestion typically use simple statistics on observed round-trip times and/or observed throughput.

The primary objective of this paper is to investigate the ability of the congestion predictors to predict a packet loss. Our measurements indicate that the three congestion predictors studied in this paper are often poor in their ability to predict a packet loss due to congestion. To arrive at this conclusion we measure the frequency with which the predictors predict congestion, and how often they predict congestion just before a packet loss. A study of the variations in measured parameters, as a function of several network parameters (for instance, router queue size) yields several interesting observations as reported in the paper.

Although the results presented here are not related to wireless communication, this research was motivated by a desire for an end-to-end mechanism for differentiating between packet losses due to congestion and packet losses due to wireless transmission errors. One technique we considered would use congestion predictors for this purpose. The results presented here suggest that simple congestion predictors will not be effective in differentiating between the two forms of losses.

Key Words: TCP – Congestion Avoidance – Corruption Losses – Congestion Losses – Congestion Predictors – Loss Predictors

1 Introduction

TCP is a popular protocol for reliable data delivery in the Internet. TCP protocol is robust in that it can adapt to disparate network conditions [9]. TCP uses congestion control mechanisms to recover

*Research reported is supported in part by the Fulbright Program and the National Science Foundation.
†This report has been submitted on February 9th 1998 to a conference.
from congestion that may occur in the network. In the context of transmission control protocols, several researchers have proposed heuristics to detect or predict congestion in the network. In this paper, the term congestion predictors refers to such heuristics. The congestion predictors [11, 5, 15] indicate when a TCP sender should reduce its window size (if congestion is detected or predicted). The heuristics in [11, 5, 15] use simple statistics on observed round-trip times (RTT) and/or observed throughput of a TCP connection.

The primary objective of this paper is to investigate ability of the congestion predictors to predict a packet loss due to congestion. The paper also determines how the predictors react to changes in several network parameters. This research was motivated by a desire to improve performance of TCP over wireless networks. Although the measurements and results presented in this paper are NOT for wireless networks, it is worth discussing the motivation, in order to understand our experimental methodology, as well as the choice of measured parameters.

**Motivation:** In recent years, with the advent of mobile computing, there has been significant interest in using TCP over wireless links [12, 2, 6, 3, 2, 1, 7]. Previous work has shown that, unless the protocol is modified, TCP performs poorly on paths that include a wireless link subject to transmission errors. The reason for this is the implicit assumption in TCP that all packet losses are due to congestion. Whenever a TCP sender detects a packet loss, it activates congestion control mechanisms [9] (these mechanisms reduce sender’s window in response to the packet loss, reducing throughput temporarily). Taking congestion control actions may be appropriate when a packet loss is due to congestion, however, it can unnecessarily reduce throughput if packet losses happen to be 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 [1, 2, 3, 16]. 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, using some end-to-end technique (that does not get any help from any intermediate host)\(^1\). Once a sender knows that the packet loss is due to congestion or corruption, 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 (with high “accuracy”) whether a packet transmitted in the near future will be lost due to congestion.

- 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 the packet was lost due to transmission errors.

\(^1\)The TCP layer at receiver may not know the cause of a packet loss even if it is a transmission error on the last hop, as a lower layer may discard the packet before it reaches the TCP layer.
The obvious question now is how to design loss predictors that can predict congestion losses with high accuracy. As noted above, several congestion predictors have been previously proposed to predict/detect congestion using only simple statistics on round-trip times and/or observed throughput of a TCP connection [11, 5, 15]. Using congestion predictors as loss predictors, a potential instantiation of the above generic procedure is as follows:

- Use a congestion predictor that suggests reducing congestion window size when congestion is detected. (A good predictor will detect congestion only when congestion is truly likely.)
- If a packet loss occurs, the sender checks what the congestion predictor recommended before the packet was sent.
- If the predictor had recommended reducing sender’s window (because congestion was detected), then conclude that the packet loss is due to congestion, otherwise conclude that packet loss is due to transmission error.

This scheme will potentially work well, if the congestion predictor is “accurate” – with high probability, an accurate congestion predictor will predict congestion before a packet loss due to congestion occurs. The main objective of this research is to study the “accuracy” of various congestion predictors, and to determine how the accuracy is affected by network parameters (such as router queue size, round trip delay, etc.).

This paper is organized as follows. Section 2 describes the three congestion predictors evaluated in this paper. Performance parameters of interest are defined in Section 3. Section 4 presents and discusses the experimental results. Simulation model and simulation results are discussed in Section 5. Conclusions are presented in Section 6.

2 Congestion Predictors

2.1 Using Congestion Predictors as Loss Predictors

As noted above, a congestion predictor, based on some criterion, may either suggest that the sender’s window size be reduced, increased, or held constant. When the congestion predictor is used for congestion avoidance and control, the TCP protocol at the sender would change the window size as per the recommendation from the congestion predictor.

In this paper, we are considering the use of certain congestion predictors for predicting packet losses due to congestion. This means that the congestion avoidance and control may be done using some other heuristic. In our experiments and simulations, we use TCP-Reno. More specifically, for congestion avoidance and control, we use the slow start and congestion avoidance techniques proposed by Van Jacobson [9, 10].
We do not use the three congestion predictors in [11, 15, 5] for congestion avoidance and control. Instead, we see how often these predictors predict congestion (i.e., suggest that sender's window size be reduced) when used in conjunction with slow start and congestion avoidance. Thus, as in TCP-Reno [10], TCP congestion window size is reduced only when a packet loss occurs.

Rest of this section describes the three congestion predictors evaluated in this paper. To describe the predictors, we first need to introduce some terminology and notations.

### 2.2 Terminology

- **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 the three parameters below, to be used in implementing the congestion predictors.

  If a time-out occurs while waiting for the acknowledgement for a monitored packet, then the data for that monitored packet is not included in our calculations.

  The monitored packets are numbered sequentially starting from 1, excluding the packets which time-out.

- **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.

  From the definition of congestion window, it follows that $W_i$ cannot exceed the congestion window size $W$. Often, $W_i$ is equal to the congestion window size at the time when the monitored packet is transmitted. However, there are certain situations wherein $W_i$ may sometimes be smaller than the congestion window.

- **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** $T_i$ from 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 $T_i = W_i/RTT_i$. This ratio represents the throughput observed during the $RTT_i$ round-trip interval for the $i$-th monitored packet.

The three congestion predictors considered in this paper are summarized below. The three predictors 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. The predictors measure this response as a
function of round-trip times and/or throughput, and recommend reducing or increasing congestion window based on the observed response.

2.3 Congestion Predictor NDG

Jain proposed a congestion predictor based on *Normalized Delay Gradient (NDG)* [11]. We will refer to this predictor as NDG. Our implementation of this congestion predictor evaluates NDG as follows, when acknowledgement for the $i$-th monitored packet is received:

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

If $NDG > 0$, this congestion predictor suggests that congestion window size should be decreased, otherwise it suggests that the window size be increased.

2.4 Congestion Predictor NTG

Wang and Crowcroft [15] proposed a congestion predictor based on the *Normalized Throughput Gradient (NTG)*. We will refer to this congestion predictor as NTG. To calculate NTG, we need to define *throughput gradient* $TG_i$ for the $i$-th monitored packet $P_i$ as follows:

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

NTG predictor evaluates *normalized throughput gradient* $NTG$ as $TG_i/TG_1$, when acknowledgement for packet $P_i$ is received. Now, $TG_1 = (T_1 - T_0)/(W_1 - W_0) = 1/RTT_1$, as $W_1 = 1$ packet, $W_0 = 0$, $T_0 = 0$ and $T_1 = W_1/RTT_1 = 1/RTT_1$. Therefore, $NTG = \frac{TG_i}{1/RTT_1}$. Substituting above expression for $TG_i$ and simplifying, we get

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

If $NTG < 1/2$, then the predictor suggests that the window size be decreased, else it suggests that the window size be increased.

2.5 Congestion Predictor Vegas

TCP-Vegas [5] maintains a variable named $BaseRTT$, which is the minimum of all $RTTs$ measured during the TCP connection. $BaseRTT$ allows Vegas to compute the *Expected Throughput*. When acknowledgement for the $i$-th monitored packet is received, the *expected* throughput is calculated 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 difference $D$ is calculated as, $D = \text{expected throughput} - \text{actual throughput} = \frac{W_i}{BaseRTT} - \frac{W_i}{RTT_i}$. Reference [5] expresses this difference $D$ in terms of extra packets in the network, by multiplying $D$ by $BaseRTT$. We define *Vegas* as,

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

*Vegas* is compared to two thresholds $\alpha$ and $\beta$, where $\alpha < \beta$. If $Vegas < \alpha$, then congestion predictor Vegas suggests that the window size be increased. If $Vegas > \beta$, the congestion predictor Vegas suggests that sender's congestion window size be decreased. Otherwise, if $\alpha < Vegas < \beta$, then it suggests holding window size constant. In our experiments and simulations, values for $\alpha$ and $\beta$ are, respectively, 2 and 4 as suggested in [5].

### 3 Performance Metrics

To characterize the ability to predict congestion losses, we measured two parameters for each congestion predictor.

- **Frequency of Congestion Prediction $FCP$:** $FCP$ is obtained by dividing the number of times the predictor said “decrease the congestion window” by the total number of times the predictor was used during the TCP connection.

For instance, assume that a congestion predictor was used 100 times during the TCP connection – each time the predictor may suggest that window size be decreased, increased, or held constant. If the number of “decrease window size” recommendations is 20, then $FCP = \frac{20}{100} = 0.20$ (or 20%).

- **Accuracy of Prediction $AP$:** $AP$ is the fraction of packet losses due to congestion that were preceded by a “decrease window size” recommendation from the congestion predictor.

For instance, assume that 80 packets were lost due to congestion during a TCP transfer. The last recommendation made by the predictor before 60 of these packet losses was “decrease window size”. Then, $AP = \frac{60}{80} = 0.75$ (or 75%).

Now, consider a “random coin tossing” congestion predictor, that uses probabilistic coin tossing to determine whether to recommend “reduce congestion window” or not. Suppose it recommends “reduce congestion window” with probability $p$. Clearly, in this case, $FCP = p$. Also, as the recommendation made by the predictor is independent of network conditions, the probability that the predictor would recommend “reduce the window” before a packet loss is also $p$. Thus, in this case, $AP = p$. 

6
From the above discussion, it is clear that a simple coin tossing scheme can yield \( AP = FCP = p \) for any desired value of \( p \). Thus, by choosing \( p = 1 \), one can obtain a 100% "accuracy". However, this is not a desirable predictor, because with \( p = 1 \), the sender will conclude that each packet loss is due to congestion, although some losses may be due to transmission errors.

For a good predictor, one would expect a significantly larger \( AP \), as compared to \( FCP \). As our experiments and simulations show, for the three congestion predictors considered here, \( AP \) is typically not large enough compared to \( FCP \). We first present the experimental measurements, followed by simulation results and discussion of the simulation results.

4 Experimental Evaluation

4.1 Experimental Set-up

Four hosts were used in our experiments, two senders and two receivers. Let us refer the two sender hosts as host1 and host2. The receiver hosts are daedalus.crosslink.net (206.246.124.8) and all-purpose-gunk.near.net (199.94.220.184). Discard server at the receiver nodes was used as the TCP receiver. The discard servers on the receiver machines have a receive window limited to 32 Kbytes.

Host1 runs TCP-Reno [10, 13] as implemented in Free BSD version 2.1.5. Host2 runs TCP-Reno with a small modification in the congestion avoidance algorithm. When TCP-Reno leaves the slow-start phase, it increases the congestion window size by \( \frac{mss^2}{cw} \) after each acknowledgement where \( mss \) is the maximum segment size and \( cw \) is the current congestion window size. On host2, we modify the rate of increase: we increase the window size by \( M \times \frac{mss^2}{cw} \), with multiplication factor \( M \) taking the values 0.5, 1, 2, 3, 5 and 10 in different experiments. With the variation of \( M \), we want to investigate how a more (or less) aggressive increase policy may affect the congestion predictors.

For each measurement, we establish two simultaneous connections from host1 and host2 to the same destination, and send the same amount of data (1.5 MBytes to all-purpose-gunk.near.net and 5 MBytes to daedalus.crosslink.net). The two connections are established simultaneously so that both experience similar network conditions. We performed six sets of measurements. In each set, different value of \( M \) is used on host2 (whereas host1 always uses \( M = 1 \)). Each set of measurements consists of 25 connections from host1 and host2 to the same destination.

4.2 Methodology

The sender nodes run Free BSD operating system. We modified tcp_debug [13] to collect the data needed to evaluate \( FCP \) and \( AP \). First, using the modified tcp_debug, a trace for each connection was produced. Then, using a modified trpt [13], for each congestion predictor, we determined what the predictor would have recommended on receiving the acknowledgement for a monitored packet.
(the recommendation may be to reduce congestion window, or increase, or hold constant). This data was then used to calculate $FCP$ and $AP$. As the same traces are used for all congestion predictors, results for different predictors may be compared with each other.

4.3 Experimental Results

As results for both the destination hosts are similar in nature, for brevity, here we present results only for destination all-purpose-gunk.near.net. Results for daedalus.crosslink.net can be found in [4].

Figure 1 plots the measured values of $FCP$ and $AP$, for the $NDG$, $NTG$ and $Vegas$ predictors. Part (a) plots results for host1 and part (b) for host2. In each graph, six curves are drawn, one for frequency of congestion prediction $FCP$ and another for accuracy $AP$ for each predictor ($NDG$, $NTG$ and $Vegas$). As noted above, we performed 6 sets of experiments, these sets are numbered 1 through 6 in the graphs for sender host1. For host2, the 6 sets use different value of the multiplicative factor $M$. The sets 1 through 6 on host2 were obtained using $M = 0.5, 1, 2, 3, 5$ and 10, respectively. For each set, the results are presented averaged over all 25 trials in the set.

We observe that, for both host1 and host2, independent of the value of multiplication factor $M$, Accuracy of Prediction (AP) is typically only a little higher than the Frequency of Congestion Prediction (FCP). This means that higher the frequency of congestion prediction, higher is the accuracy of prediction.

As noted earlier, a random "coin tossing" predictor with frequency of congestion prediction of $p$ will give an accuracy of prediction of $p$. For the three predictors we evaluated, $AP$ is only marginally better than $FCP$. Thus, these congestion predictors do not seem to perform much better (as a loss predictor) than a random predictor.

4.4 Interpretation of Experimental Results

Having observed that, when used as loss predictors, the three congestion predictors do not perform much better than random coin tossing, it is useful to provide an intuitive explanation of this result. A predictor will be accurate only if the following qualitative conditions are fulfilled, as explained below:

a) Losses are preceded by a "long" queue build-up at some router

b) A queue build-up typically results in losses

c) The congestion predictor correctly senses "serious" queue build-up.

Condition a) means that the interval of time, say $T_{bo}$, between the instant when a router queue starts to build up and the instant when the queue overflows must be long enough. Otherwise, losses
Figure 1: Predictors NDG, NTG and Vegas with receiver all-purpose-gunk.near.net
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.

Condition c) above is necessary for congestion to be detected.

As noted earlier, the three predictors considered here are based on the expectation that a variation in the congestion window size must result in a "response" from the network which reflects the true state of the network. Unfortunately, the traffic on one connection is in general a small fraction of the overall traffic. Therefore, the network response is almost independent of one TCP connection's action. This suggests that the three predictors cannot correctly detect queue build-up. While our experiments (under specific network conditions) seem to support this conclusion, further measurements, with different network conditions, are needed to better understand behavior of the congestion predictors.

5 Simulations

5.1 Simulation Model and Methodology

We use the network simulator ns-2 (version 2.1b1) [14] from Berkeley. The system model used for simulations is illustrated in Figure 2. This model is simple, yet serves our purpose. We have a TCP connection from a source CS to a sink CK. We use the FullTcp [8] agent for this connection. This connection shares the link $R_1 \leftrightarrow R_2$ with a cross traffic issued from a random source $R_S$ to sink $R_K$. All the links in Figure 2 are labeled with a $(bandwidth, propagation delay)$ pair. The links $R_2 \leftrightarrow CK$ and $R_2 \leftrightarrow RK$ have bandwidth of 8 MBits/s and propagation delay of 1 ms in all our simulations. For the other links, we simulate the network with different values for parameters $bw$ and $\delta$ (please refer Figure 2). In different simulations, $bw$ takes the values 100 Kbits/s, 500 Kbits/s, 1000 Kbits/s, 1.5 Mbits/s, and 2 Mbits/s and $\delta$ takes values 2 ms, 4 ms, 7 ms, 9 ms, 12 ms, 17 ms, 22 ms, 37 ms, 49 ms, and 74 ms. The link $CS \rightarrow R_1$ has propagation delay of $(\delta + 1)$ ms so that the potential bottleneck, router $R_1$, is half way from node $CS$ to $CK$. Routers $R_1$ and $R_2$ use simple FIFO drop-tail queue policy.

The router $R_1$ will have an output queue (towards $R_2$) whose size is limited to $qs$ packets. $qs$ takes the values 5, 10, 15, or 20 in our simulations. All other queues at the two routers are unbounded (infinite). Obviously, the potential bottleneck here is the router $R_1$.

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 $\delta$ used in our simulations, $T_p$ varies in the range 10 ms to 300 ms.
The traffic from the random source is produced by the agent Traffic/Expo [14]: it is a constant-bit rate (CBR) source with idle time and busy time exponentially distributed with mean 0.5 s. UDP is the transport protocol used for the random source. For each set of parameters \((T_p, bw, qs)\), the peak rate of the random source is adjusted to produce a desired value of “aggregate loss rate” on the TCP connection, measured as the percentage (or fraction) of packets lost due to congestion. The loss rate is denoted as loss in the rest of the paper.

For each set of parameters \(T_p, bw, \text{ and } qs\), we perform many trials (the TCP connection transfers 2 Mbyte in each trial) to determine the peak rate of the random source, so as to get a desired loss rate \((loss)\) for the TCP connection. When this peak rate is determined, we make 10 additional TCP transfers of 2 MBytes each, using this peak rate for the random source, and collect statistics for these 10 transfers. Each transfer starts after a random warm-up period larger that 100 seconds. During the warm-up period, only the random source is active.

We perform our measurements exactly as we did with the live experiments. We monitor one packet per window: we log its round trip time and the number of packets sent between its transmission and its acknowledgement. For each connection, we transfer 2 MBytes from CS to CK. The congestion window size is limited to 32 packets. From the logged information, we can compute the congestion predictors \(NDG, NTG\) and \(Vegas\) as defined in section 2. Note that the three predictors are computed from the same set of logged data. For the ten transfers, the standard deviation on all congestion predictors is less than 0.02, which represents on the average 4%.

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

- the loss rate \((loss)\) from 1% to 10% (loss rate specifies fraction of packets lost by the TCP connection at router \(R_1\))
When \textit{loss} is held constant for a particular set of simulations, we hold it constant at 3\%, because in our experiments (reported in the previous section) we observed a loss rate of approximately 3\%.

- round-trip propagation time $T_p$ for the TCP connection is in the range 10 ms to 300 ms.

  When $T_p$ is held constant for a particular set of simulations, we hold it constant at 40 ms. (For our experiments, we estimate that the average round-trip delay was in the range 20 ms to 80 ms. 40 ms lies in the lower half of this range.)

- the bottleneck bandwidth $bw$ from 100 KBits/s to 2 Mbits/s.

  When $bw$ is held constant for a particular set of simulations, we hold it at 1.5 Mbits/s (T1 bandwidth).

- the queue size limit $qs$ at router $R_1$ from 5 to 20 packets (packet size is 1460 bytes).

  When $qs$ is held constant for a particular set of simulations, we hold it at $qs = 5$.

5.2 Simulation Results

The simulation results provide several interesting insights, and confirm the measurements done using experiments: the difference $AP - FCP$ is usually positive and small for the three predictors. This difference exceeds 0.20 the Vegas and NTG predictors only under certain conditions. In the following, we provide graphs showing only some of our simulation results. However, the conclusions reported here are drawn from a larger set of simulations.

5.2.1 Vegas Predictor

Recall that if $Vegas > \beta$, then the Vegas predictor predicts congestion. It is clear that the probability that $(Vegas > \beta)$ decreases as $Vegas$ decreases. Now we summarize our observations based on the simulation results, and attempt to provide intuitive (or mathematical) explanations. First we discuss variation trends for $FCP$, followed by $AP$.

- $FCP$ decreases when $T_p$ is increased, while holding $bw$, $qs$ and $loss$ constant. Refer Figure 3(a) for an illustration. In Figure 3(a), 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
\[ t_{\text{Base}} \leq t_i \] (\textit{BaseRTT} is the smallest round trip time experienced by the connection). Then, \[ \text{Vegas} = W_i \left(1 - \frac{t_{\text{Base}} + t_i}{T_p + t_i}\right). \] Thus, \[ \frac{\delta(\text{Vegas})}{\delta T_p} = W_i \left(\frac{t_{\text{Base}} - t_i}{(T_p + t_i)^2}\right). \] While, in general, \( t_i \geq t_{\text{Base}} \), typically we have \( t_i \geq t_{\text{Base}} \). Therefore, \( \frac{\delta(\text{Vegas})}{\delta T_p} \) is usually negative. This means that the value of \textit{Vegas} decreases when \( T_p \) is increased. Therefore, as \( T_p \) increases, \textit{FCP} for \textit{Vegas} predictor should decrease.

\[ \text{FCP decreases when } \text{bw is increased, keeping } T_p, t_s \text{ and loss constant, as illustrated in Figure 3(b).} \]

Similar to the above derivation, we provide a mathematical explanation for this observation. Let us express \( RTT_i \) as \( RTT_i = \text{BaseRTT} + d_{qi} \) where \( d_{qi} \) is the extra queueing delay for the \( i \)-th monitored packet, as compared to \textit{BaseRTT}. (assuming that the round trip time variation is due only to the queueing delays). Thus, \( \text{Vegas} = W_i \left(1 - \frac{\text{BaseRTT}}{\text{BaseRTT} + d_{qi}}\right). \) Since \[ \frac{\delta(\text{Vegas})}{\delta d_{qi}} = \left(\frac{W_i \text{BaseRTT}}{(\text{BaseRTT} + d_{qi})^2}\right) > 0, \] the \textit{Vegas} predictor increases with increasing queueing delay \( d_{qi} \). From queueing theory, it follows that, queueing delay variations decrease when the service rate increases, i.e., in this case, when \( \text{bw} \) increases. Therefore, when bandwidth \( \text{bw} \) is increased, queueing delay \( d_{qi} \) will decrease, and consequently \( \text{Vegas} \) will decrease (as \( \frac{\delta(\text{Vegas})}{\delta d_{qi}} > 0 \)). Finally, when \( \text{Vegas} \) decreases, the \textit{FCP} for the \textit{Vegas} predictor also decreases.

\[ \text{FCP increases when } t_s \text{ is increased, keeping } \text{bw, } T_p \text{ and loss constant, as illustrated in Figure 4(a).} \]

As \( t_s \) increases, with the loss rate held constant, the queueing delay variations increase. We showed above that the \textit{Vegas} predictor increases with queueing delays variations. Therefore, the \textit{Vegas} predictor increases with increasing \( t_s \). Thus, \textit{FCP} will increase with increasing \( t_s \).
FCP decreases when loss is increased, keeping bw, $T_p$ and $qs$ constant, as illustrated in Figure 4(b).

It is somewhat counter-intuitive that $FCP$ decreases with increasing loss rate. Now, $\frac{\delta_{Vegas}}{\delta W_i} = 1 - \frac{\text{BaseRTT}}{\text{RTT}_i}$. While, in general, $\text{BaseRTT} \leq \text{RTT}_i$, typically we have $\text{BaseRTT} < \text{RTT}_i$, therefore, $\frac{\delta_{Vegas}}{\delta W_i}$ is typically positive. Thus, if $W_i$ decreases, then $Vegas$ will also decrease. Now, note that, as loss increases, the average congestion window size, and thus $W_i$, decreases. Therefore, with increasing loss, $Vegas$ will decrease, consequently, the $FCP$ for the Vegas predictor will also decrease.

**Accuracy of prediction $AP$:** Accuracy of prediction usually follows $FCP$'s trends. Typically, $AP$ is only marginally higher than $FCP$. For our purpose of distinguishing corruption losses on wireless links from congestion losses, we need an accuracy $AP$ close to 1.00. Otherwise, mistaking congestion losses for corruption losses makes congestion control inefficient and jeopardizes performance. On the other hand, we need the Frequency of Congestion Prediction $FCP$ to be small. Otherwise, a high proportion of wireless transmission losses can be mistaken for congestion losses, and congestion control mechanism would be unnecessarily triggered leading to poor performance. For round trip propagation time less than 32 ms, low packet loss rate (less than 3%) and queue size larger than 10 packets, we have $AP$ larger than 0.75 and $FCP$ less than 0.40. These values ($AP = 0.75$ and $FCP = 0.40$) can be considered better than what a random coin tossing predictor would provide. Under real network conditions, the parameter values (such as RTT and loss rate) may not always correspond to the above ranges, therefore, in general, performance of the predictor is not very good.

As noted in the previous section, Vegas (and, also the other 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 the TCP connection. Therefore, accuracy of prediction tends to be poorer than one may expect. (This same reason causes other predictors to perform poorly as well.)

5.2.2 NTG Predictor

Recall that, if $NTG < \frac{1}{2}$, then the NTG predictor predicts congestion. Therefore, as $NTG$ increases, $FCP$ decreases. Now we present the observations for simulations using the NTG predictor.

![Graph](image)

(a) FCP and AP versus $T_p$

(b) FCP and AP versus $bw$

Figure 5: Effect of the propagation time $T_p$ and the bandwidth $bw$

- $FCP$ decreases when $T_p$ is increased, while holding $bw$, $qs$ and $loss$ constant. Refer Figure 5(a) for an illustration.

To support this observation, we show that $NTG$ 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 $NTG$ as:

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

Now note that, since we are using TCP-Reno, 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 $NTG \geq \frac{1}{2}$, provided $max\left(\frac{T_p+d_i}{T_p+d_{i-1}}, \frac{T_p+d_i}{T_p+d_i}\right) \geq \frac{1}{2}$. The condition $max\left(\frac{T_p+d_i}{T_p+d_{i-1}}, \frac{T_p+d_i}{T_p+d_i}\right) \geq \frac{1}{2}$ 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.

Now,

\[
\frac{\Delta NTG}{\Delta T_p} = \frac{T_p + d_i}{W_i - W_{i-1}} \left( \frac{W_{i-1}}{(T_p + d_{i-1})^2} - \frac{W_i}{(T_p + d_i)^2} \right) + \left( \frac{W_i}{T_p + d_i} - \frac{W_{i-1}}{T_p + d_{i-1}} \right).
\]

As noted before, typically \( W_i - W_{i-1} > 0 \). Also, typically, \( RTT_i > RTT_{i-1} \) (as \( W_i \) typically much larger than \( W_1 = 1 \)). Assuming this, it can be shown that, if \( d_{i-1} < d_i \), \( \frac{\Delta NTG}{\Delta T_p} > 0 \) provided that \( (T_p + d_i)^2 \geq (d_{i-1} - d_i)(d_i - d_{i-1}) \). 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, the NTG value computed by Equation 1 typically increases with increasing propagation time \( T_p \). Therefore, FCP decreases when \( T_p \) increases.

- FCP decreases when \( bw \) is increased, keeping \( T_p \), \( q_s \) and loss constant, as illustrated in Figure 5(b).

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. \( NTG \) becomes then:

\[
NTG = \frac{RTT_i}{W_i - W_{i-1}} - \left( \frac{W_i}{RTT_{i-1} + d_q} - \frac{W_{i-1}}{RTT_{i-1}} \right)
\]

Then, \( \frac{\Delta NTG}{\Delta d_q} = -\frac{RTT_i W_i}{(W_i - W_{i-1})(RTT_{i-1} + d_q)^2} \). As, for TCP-Reno, typically \( W_i > W_{i-1} \), we have \( \frac{\Delta NTG}{\Delta d_q} < 0 \). Thus, \( NTG \) 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 \( q_s \) is increased, keeping \( bw \), \( T_p \) and loss constant, as illustrated in Figure 6(a).

For a constant loss rate, as \( q_s \) increases, the amount of random source’s traffic in the queue ahead of a TCP packet increases. Therefore, queueing delay variation for TCP packets is larger. We showed above that \( NTG \) decreases with increasing queueing delay variations. Thus, FCP decreases with increasing \( q_s \).

- FCP decreases when loss is increased, keeping \( bw \), \( T_p \) and \( q_s \) constant, as illustrated in Figure 6(b).
As for \textit{Vegas} predictor, the trend of \textit{FCP} for \textit{NTG} predictor with the loss rate is related to the average congestion window size. However, while $\frac{\delta \text{Vegas}}{\delta W} > 0$ is typically true, $\frac{\delta \text{NTG}}{\delta W} < 0$ is true only when $\text{RTT}_i > \text{RTT}_{i-1}$. However, note that the probability that $\text{RTT}_i > \text{RTT}_{i-1}$ is greater than 0.5 (as, typically, $W_i > W_{i-1}$, and larger $W$ typically – though not always – results in greater RTT). The above two observations together imply that \textit{NTG} value should show an increasing trend as a function of \textit{loss}, however, the trend may not be as pronounced as the decreasing trend for \textit{Vegas}.

\begin{figure}[h]
\centering
\begin{subfigure}{0.45\textwidth}
\centering
\includegraphics[width=\textwidth]{a.png}
\caption{FCP and AP versus $qs$}
\end{subfigure}\hspace{1cm}
\begin{subfigure}{0.45\textwidth}
\centering
\includegraphics[width=\textwidth]{b.png}
\caption{FCP and AP versus $loss$}
\end{subfigure}
\caption{Effect of the queue size $qs$ and the loss rate $loss$}
\end{figure}

\textbf{Accuracy of prediction AP:} Accuracy of prediction is always marginally larger than to \textit{FCP} for the \textit{NTG} predictor than for the \textit{Vegas} predictor. If $AP$ is close to one, then \textit{FCP} is also close to 1. Therefore, \textit{NTG} is not much better than a random coin tossing predictor. Thus, \textit{NTG} is not a good loss predictor.

\subsection{5.3 NDG Predictor}

Recall that if \textit{NDG} is positive then the NDG predictor predicts congestion. Also, in our simulations, the agent \textit{FullTcp} uses the Jacobson congestion avoidance algorithms. Thus, often $W_{i-1} < W_i$ and the sign of \textit{NDG} depends only on the sign of $\left(\frac{\text{RTT}_{i-1}}{\text{RTT}_i} - 1\right)$.

From the simulation results, we observed that for NDG predictor, the value of \textit{FCP} is typically in the range 0.5 to 0.6. Now note that packet round trip time can vary due to the queueing delay at the router $R_i$. The queueing delay for a packet $P$ depends on the number of packets in front of $P$ in the queue. These packets can be originated at the random source $RS$ or at the controlled source $CS$. Since the two sources are independent, the sign of $\left(\text{RTT}_i - \text{RTT}_{i-1}\right)$ is about equally likely to be positive or negative. This can explain why the frequency of congestion prediction \textit{FCP} is around
Actually, $FCP$ is typically slightly larger than $0.5$ because, on the average, a larger congestion window will likely contribute to larger queueing delays.

![Graph](image)

(a) $FCP$ and $AP$ versus $T_p$

![Graph](image)

(b) $FCP$ and $AP$ versus $bw$

Figure 7: Effect of the propagation time $T_p$ and the bandwidth $bw$

The simulation results indicate that $FCP$ and $AP$ for the NDG predictor do not show any trends (increasing or decreasing) as a function of the four parameters $T_p$, $bw$, $qs$ and $loss$. Now we attempt to provide intuitive explanation for this.

- **Variation of $FCP$ when $T_p$ is increased, while holding $bw$, $qs$ and $loss$ constant.** Refer Figure 7(a) for an illustration.

  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 $T_p$.

- **Variation of $FCP$ when $bw$ is increased, while holding $T_p$, $qs$ and $loss$ constant.** Refer Figure 7(b) for an illustration.

  The actual value of bandwidth $bw$ has an impact on the magnitude of $RTT_i - RTT_{i-1}$, but not on its sign. The sign of $(RTT_i - RTT_{i-1})$ depends on the random behavior of the random source. Therefore $FCP$ is independent from $bw$.

- **Variation of $FCP$ when $qs$ is increased, while holding $T_p$, $bw$ and $loss$ constant.** Refer Figure 8(a) for an illustration.

  Queue size $qs$ has an impact on the magnitude of queueing delays. Since $NDG$ depends on the difference between queueing delays for different packets, but not on the magnitude, $NDG$ is independent of $qs$. Figure 8(a) suggests a slight increase in $FCP$ with increasing $qs$—however, in other simulations with different parameters, no such trend appears. We presented this plot for consistency with results provided for $Vegas$ and $NTG$.  

18
Variation of FCP when loss is increased, while holding $T_p$, $bw$ and $qs$ constant. Refer Figure 8(b) for an illustration.

The loss rate affects size of the TCP congestion window. Although $NDG$ 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.

**Accuracy of prediction AP:** AP curves usually track FCP curves, and $AP - FCP$ difference is small. Moreover, $AP$ rarely reaches the value 1.00. A random coin tossing predictor with probability 0.5 would perform similarly. Thus, $NDG$ is not a good loss predictor. $NDG$ is the worst loss predictor in comparison with Vegas and $NDG$.

### 6 Conclusion

We studied three heuristics used for congestion avoidance with the objective of distinguishing corruption losses on wireless links from congestion losses. The three heuristics Vegas, NTG and $NDG$ were studied through experimentation and simulation. The experimentation, under specific network conditions, showed that the three heuristics are poor as loss predictors. Our simulations, varying the round trip propagation time, link bandwidth, congestion loss rate and queue size, also showed that under most of the conditions, these three heuristics perform poorly as congestion loss predictors. However, it appears that Vegas predictor performed somewhat better than the other two predictors, while $NDG$ predictor appears to be the worst, in terms of loss prediction.
References


