

Bayesian Packet Loss Detection for TCP

Nahur Fonseca and Mark Crovella

Abstract— One of TCP’s critical tasks is to determine which packets are lost in the network, as a basis for control actions (slow control and packet retransmission). Modern TCP implementations use two mechanisms: timeout, and fast retransmit. Detection via timeout is necessarily a time-consuming operation; fast retransmit, while much quicker, is only effective for a small fraction of packet losses. In this paper we consider the problem of packet loss detection in TCP more generally. We concentrate on the fact that TCP’s control actions are necessarily triggered by *inference* of packet loss, rather than conclusive knowledge. This suggests that one might analyze TCP’s packet loss detection in a standard inferencing framework based on probability of detection and probability of false alarm. This paper makes two contributions to that end: First, we study an example of more general packet loss inference, namely optimal Bayesian packet loss detection based on round trip time. We show that for long-lived flows, it is frequently possible to achieve high detection probability and low false alarm probability based on measured round trip time. Second, we construct an analytic performance model that incorporates general packet loss inference into TCP. We show that for realistic detection and false alarm probabilities (as are achievable via our Bayesian detector) and for moderate packet loss rates, the use of more general packet loss inference in TCP can improve throughput by as much as 25%.

Index Terms— Queuing theory/Performance Evaluation, Network Measurements

I. INTRODUCTION

The detection of packet loss is at the core of modern TCP implementations. TCP works in a cycle, it increases the utilization of network resources gradually up to the limit when packets are dropped; at this point TCP has to backoff its sending rate and retransmit the lost packets, ending the cycle. TCP detects a packet loss by two mechanisms, it either waits for a timeout of the retransmission timer, or it waits for the arrival of a few dupacks – ACKs with the same sequence number – from the receiver. The first mechanism is necessarily time consum-

ing¹, because the retransmission timer must be set high enough to enable the network recover from severe congestion events, and also to avoid unnecessary timeouts caused by transient network conditions. The second mechanism is faster, however it assumes that if a packet receives a few (usually three) dupacks then it was lost, although a reordered packet may also cause a receiver to generate dupacks. Thus TCP control actions are necessarily triggered by *inference* of packet loss, rather than conclusive knowledge. The retransmission timer mechanism is guaranteed to detect all packet losses, but also generates some false positives; whereas the fast retransmit mechanism may not detect all packet losses, and may generate false positives.

We propose to leverage TCP inference mechanisms with a Bayesian framework based on round trip delay measures (or simply RTT). Our inference framework is based on the following phenomenon illustrated in Figure 1. First remember that RTT is the sum of propagation, transmission and queuing delays. Second when TCP sends a sequence of back-to-back packets they are put in the end of a queue at the bottleneck link. If this queue does not have enough room only the packets in front of the sequence will be put in the queue, the others will be dropped. We are assuming a FIFO queue. In the event of a packet loss the last successfully transmitted packet will likely experience a high delay, because it will have to wait for the transmission of a queue full of packets. This case is illustrated by the sequence of packets (a_1, a_2, a_3, a_4) in Figure 1. On the other hand, in the event of packet reordering, the last packet transmitted in sequence may not experience a high delay, because the reason for a reordering event is not a buffer overflow. The cause is related to network structure (e.g. multi-channel paths, load balancing, etc.). This latter case is illustrated by the sequence of packets (b_1, b_3, b_2) in Figure 1.

This phenomenon may be captured by conditional probability density functions. We are interested in two conditions, or hypotheses, either a packet is *lost* or *reordered*. For each of these we have a conditional probability density function of delay, $p(y|\theta)$, where $\theta = \{\text{loss, reordering}\}$. Given a particular delay value y_0 , $p(y_0|\theta)$ gives the likelihood of y_0 for each hypothe-

Authors are with the Boston University Computer Science Department. Email: {nahur,crovella}@cs.bu.edu. This work was supported by NSF Grants ANI-0095988, CCR-0325701, and ANI-0322990. This work was performed while M. Crovella was at Laboratoire d’Informatique de Paris 6 (LIP6), with support from Centre National de la Recherche Scientifique (CNRS) France and Sprint Labs.

¹The recommended minimum retransmission timer is 1 second [1].

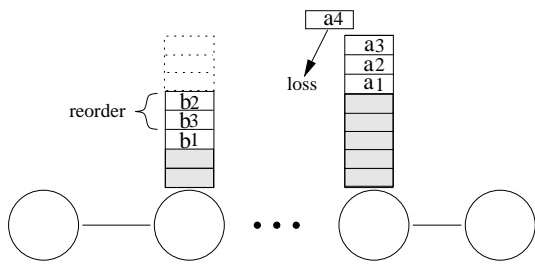


Fig. 1. Illustration of the Queue Build Up Phenomenon

sis. Thus we believe that $p(y_0|\text{loss})$ will be higher than $p(y_0|\text{reordering})$ for high y_0 , and vice-versa. Using the Bayes's rule, it is possible to invert the conditional in this function in order to get $p(\theta|y) \propto p(y|\theta)p(\theta)$. This gives a way to compute the odds of each hypothesis for a particular value, encoded by y , which could be a sequence of observations. This is a typical inference problem, casted using a Bayesian framework.

With this task at hand we need good models for delay under each hypothesis, loss and reordering. These models are encoded using conditional density functions. Unfortunately there are no out-the-shelf models of delay for TCP, since there are many factors like network topology, time of day, degree of multiplexing that may change the characteristics of RTT. Therefore, in Section IV we look at three different techniques to estimate the distribution of RTT conditioned to loss and reordering. Once a dupack arrives at the sender, it computes the odds of each hypothesis based on the current and previous RTT measurements. If a loss is inferred, TCP triggers fast retransmit, otherwise, it does nothing. We do not propose to substitute the current TCP mechanisms, so, if a packet is not acknowledged before the retransmission timer expires, TCP will trigger slow-start, and if enough dupacks arrive for a packet, TCP will trigger fast retransmit.

We evaluate this framework using traces of real TCP traffic collected in front of a Web server of a University campus, and also from routers located at different points in the *middle* of the Internet. After filtering the relevant connections, we classify the packets for which dupacks were received, as either lost, reordered or unknown. For each connection, we emulate a TCP sender that uses our Bayes detector to infer packet losses, and compare the output of the inference mechanism to the offline classification. We find that the fraction of losses that are detected by the inference mechanism is in the range 80 – 90%, and the fraction of reordered packets that are misclassified as lost is in the range 10 – 40%.

Next we ask the following question *what is the benefit of using such a detector in TCP?* In Section V, we answer it by extending a model of TCP developed by Padhye et

al.[2], with which is possible to calculate the throughput of TCP. We find that for the kind of detectors obtained as above, and in fair conditions of loss and reordering rates (5% and 0.2%, respectively) the improvement in throughput for TCP is as much as 30%.

The rest of the paper is organized as follows. Related work is presented in Section II. The traces we used to evaluate the Bayes framework is presented in Section III. In Section IV we present the Bayes framework, the three techniques to estimate the delay distributions and the performance of the Bayes detector thus obtained. The extended TCP model and its evaluation is presented in Section V. Finally, we conclude the paper in VI with a discussion of the applicability of this work.

II. RELATED WORK

The packet loss detector proposed in this paper resembles the idea of the detector used in TCP Vegas [3]. Vegas includes a simple packet loss detector based on delay. On arrival of a dupack, Vegas checks the delay since the transmission of the first unacknowledged packet; if this delay is larger than a fine-grained timeout value (updated every ack), the packet is assumed to be lost, and so is retransmitted. It performs the same test for the first two normal ACKs that arrive after a sequence of dupacks in order to recover from two or three losses which happen close together. In our work, we use a Bayesian framework to detect losses, instead of the simple threshold test used by Vegas.

A Bayesian framework similar to ours has been used in evaluating TCP in hybrid wired/wireless networks, with the objective of distinguishing between wireless losses caused by link errors, and losses caused by congestion. Liu *et al.* [4] observed differences in location and scale between the distributions of RTT in the eminence of congestion losses and wireless losses. As a result, they proposed a Hidden Markov Model in which each state is characterized by a Gaussian distribution of RTT with different mean and variance. Each state is then associated with a type of loss during a training phase. Barman *et al.* [5] introduced an explicit loss labeling mechanism in TCP NewReno [6]. The observation in that work is that RTT is highly variable in the presence of congestion loss. Thus they used two exponentially weighted moving averages of delay. One is used only for recent samples which fall within a small range (low variability), the other is used when a sample falls off that range and thus has higher variability. Whenever a packet loss is detected, their method counts how many times the more variable EWMA was used; if it is above a threshold, a congestion loss is inferred, otherwise a wireless loss is inferred.

In contrast, in this work, we are not trying to distinguish the nature of a packet loss, but rather the presence of packet loss per se. Many other papers have looked at this problem from a more practical perspective. A number of papers have proposed new techniques to improve TCP packet loss detection [7], [8], [9], or to recover from a wrong detection [10], [11], [12]. These papers then compare the two versions of TCP with and without the new technique in terms of throughput, fairness, etc., under different network conditions using either simulation or experimentation. In contrast, rather than evaluating our method in simulation, we build an analytic model of the resulting algorithm. This enables us to analyze the impact of different detectors on TCP performance independently of the technique used.

Finally, it is reasonable to wonder whether round-trip delay is a good choice of observable feature on which to base a Bayesian packet loss detector. In fact, the variance of RTT is very high, as first observed by Paxson and Floyd [13], [14] and recently confirmed in a large study conducted by Aikat *et al.* [15]. This may make the distribution of delays for different hypotheses appear too much alike (*e.g.*, loss *versus* reordering, congestion *versus* wireless loss, etc.) Another observation, made recently by Biaz and Vaidya [16], is that the RTT is poorly correlated with the state of a single TCP connection going through a link shared by many flows (*i.e.* in the presence of intense cross-traffic). Notice, however, that in our case, we are exploring the correlation of round trip delay with the network state, not with one single TCP connection state. And our results on the performance achieved by the Bayes detector (described in § IV) suggest that using round-trip delay as a feature for a packet loss detector for TCP is in fact viable.

III. DATASETS

To conduct this study, we use traces of TCP connections that include every packet seen on a link. NLANR has a collection of such traces as part of their *Passive Measure Project*, or simply PMA. We call these traces the PMA datasets. We also collected TCP traces from a link directly in front of a Web server at Boston University (using the Unix utility `snoop`). We call these traces the BU datasets.

The main difference between these two datasets is the location of the measurement point. The PMA datasets are collected in machines which are in the middle of a path between communicating TCP end-systems. In contrast the BU datasets are collected in a proxy host in front of a Web server, and therefore are quite close to the sender. To compute the RTT in the PMA traces, we use the technique described in [17]. That method estimates the congestion

	Dupacks	Fraction	P(loss dupack)
BU-7	3,916	2.0%	77.4%
BU-12	9,856	2.0%	50.4%
BU-14	12,509	1.6%	50.9%
PMA-BUF	2,270	1.8%	28.3%
PMA-MEM	639	3.4%	81.4%
PMA-MRA	4,892	2.1%	63.4%
PMA-TXG	2,909	6.7%	68.1%

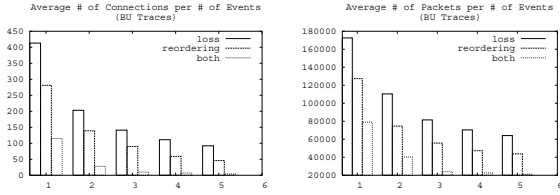
TABLE I

PRIOR PROBABILITY OF DUPACKS CAUSED BY PACKET LOSS

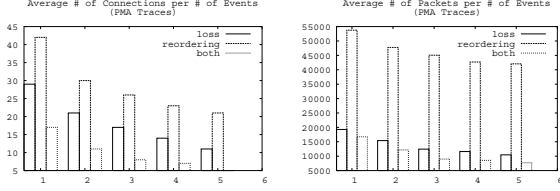
window of the TCP sender, and approximates the RTT by the time difference between the transmission of two data packets, where the transmission of the second is triggered by the reception of the acknowledge for the first. In the case of the BU traces, because the TCP sender is close to the measuring host, we approximate the RTT by the time difference between the transmission of a data packet and the arrival of its corresponding ACK. As a result, the quality of the RTT measurements in the BU traces is better than that for the PMA traces.

We only used symmetric, complete, long and interesting connections in our study. A connection is said to be symmetric if the collecting point sees both sides of the connection, the data packets and the ACK packets. We used the three-way handshake of TCP to identify such connections. A complete connection is one that has all its data packets and corresponding ACKs from its establishment by a SYN packet, to its end by a FIN packet. We considered a connection long and interesting if it had more than fifty packets and at least one dupack. In Table I we show the number of packets that received duplicate acks, their fraction in relation to the total number of packets, the fraction of these duplicate acks which were caused by packet loss. The rest of the dupacks were caused by re-ordered packets.

After connections were filtered, their packets were classified using an approach similar to that described in [17]. We collect both sides of a TCP conversation, and thus we are able to classify their packets into four classes: successfully transmitted, transmitted out-of-order, retransmitted due to a loss and others. In Fig. 2 we show the number of connections we were able to classify loss or reordered events or both, and how many packets these connections had. For instance, in the three BU traces, we were able to find at least one loss event in approximately 400 connections on average; approximately 275 connections had at least one reordering event; and a bit more than 100 connections had simultaneously at least one loss and one re-



(a) BU Traces



(b) PMA Traces

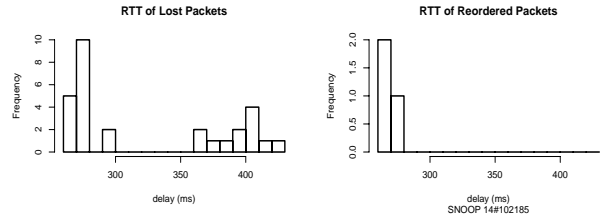
Fig. 2. Breakdown of Dupack into Loss and Reordering Events

ordering event. We observe that whether a connection has a loss event seems approximately independent of whether it has a reordering event. We also observe that for these datasets most connections have a small number of loss or reordering events; for example, about half of the connections that have at least one loss or reordering event have less than four events. Finally, we observe that the PMA dataset had, proportionally, more reordering events than the BU dataset. We attribute this difference to two causes. First, in order to a packet loss be classified for our use it needs to have a valid RTT measure immediately before it. In the PMA traces obtaining a valid RTT measure before a loss is more difficult than it is in the BU dataset, because the measurement point is in the middle of the end-to-end path. Second, it is known that the causes of packet reordering are related to the network’s physical properties [18], [19], so we conjecture that the PMA traces were collected in a network which causes more packet reorderings than is the case in the BU dataset.

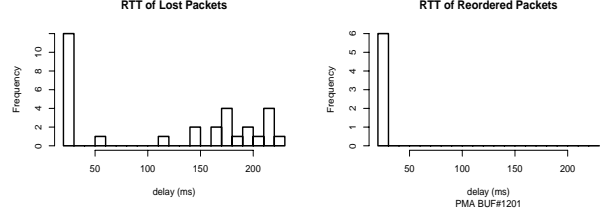
IV. BAYESIAN INFERENCE

In this section we present and evaluate a packet loss inference mechanism for TCP using a Bayesian framework based on the distribution of RTTs for packets which immediately precede the loss event we need to detect.

In order to gauge the existence of the phenomenon depicted in Fig. 1 we investigated the distribution of RTTs. In Fig. 3 we show histograms of RTT distribution of two TCP connections. On the left of Fig. 3 we plot histograms of RTTs for packets which immediately preceded a packet



(a) A Connection from BU Dataset



(b) A Connection from PMA Dataset

Fig. 3. RTT Histograms Conditioned to Loss and Reordering Events

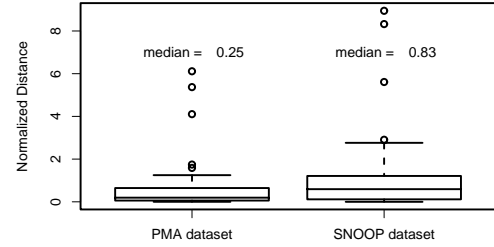


Fig. 4. Normalized Distance between Distribution of RTTs

loss, and on the right, we plot the same for packets which immediately preceded a reordering event. We see that there is a fair amount of mass, in the case of packet loss, that have larger RTT values than in the case of reordering events. This large values correspond to packets that were put in the end of a queue, just before another packet of the same row was dropped due to buffer overflow. In the case of lost packets the small values of RTT may correspond to packets that were intermixed with a burst of packets from other rows, which in turn caused the packet drop.

However, it is evident in Fig. 3 that the respective distributions of RTT in the event of a packet loss and reordering are different. It is possible to quantify this difference with the normalized distance between the distributions of RTTs of the two hypotheses, denoted by

$$d^2 = \frac{(E[RTT|loss] - E[RTT|reordering])^2}{\sqrt{Var(RTT|loss)Var(RTT|reordering)}}$$

The higher d^2 the more distinct are the distributions, and the easier is the problem of detecting them. Some statistics of this quantity are plotted in Fig. 4 for all connections in our datasets. The boxes delimit the 25th and the 75th percentiles, the line inside the box gives the median of the distribution, and the whiskers outside the box mark the minimum and maximum values of d^2 . Small circles are outliers. We notice in Fig. 4 that the d^2 statistic is very low in our datasets, thus we expect the detection process to be difficult.

A. The Bayes Detector

There are three components of the Bayes detector. First is the process that generates the events that we are interested in. We are interested in the causes of dupacks. There are only two possible causes, either a dupack is generated by a packet loss, or by a reordering event.² The only thing that we need to know from this process is the probability of loss, namely P_L . Since there are only two hypotheses, the probability of reordering is simply $P_{\bar{L}} = 1 - P_L$.

Second is the measurement process. We have chosen to use RTT as a measurement because of its distinct distribution under the two hypothesis that we have, as it was illustrated in Fig. 3. Let $R(i)$ be a random variable that describes the RTT of packet i , $L(i)$ represent the event of packet i being lost, and $\bar{L}(i)$ the event of packet i being reordered. From the measurement processes we need the two conditional probability distributions functions as pieces of the Bayes detector, $p(R(i)|L(i+1))$ and $p(R(i)|\bar{L}(i+1))$.

The last piece of the Bayes detector is the decision rule. A decision rule simply maps each value of RTT to one of the hypotheses. In a Bayes detector, an optimal decision rule is one that minimizes the Bayes risk. The Bayes risk is given by the function $C(D(r, i), H(i+1))$, where $D(r, i) \in \{L, \bar{L}\}$ is the decision taken by the detector for a delay value $R(i) = r$ of packet i , and $H(i+1) \in \{L, \bar{L}\}$ is the actual event that happened to packet $i+1$. A typical cost assignment is $C(L, L) = C(\bar{L}, \bar{L}) = 0$ (no penalty for right decisions) and $C(L, \bar{L}) = C(\bar{L}, L) = 1$ (high penalty for wrong decisions), but the Bayes cost can be tunable, to make TCP more or less aggressive, or to adapt it to a network with persistent reordering, for example. Therefore, given a cost assignment, the Bayes cost is given by $E[C] = \sum_{D \in \{L, \bar{L}\}} \sum_{H \in \{L, \bar{L}\}} C(D, H) Prob(D, H)$.

²There are other two causes of dupacks, network duplicates and unnecessary retransmissions. The former is very rare and was ignored in this work; the dupacks generated by the latter can be discarded simply by looking at the last packet sent. A loss or reordering event can not generate a dupack for the last packet sent.

Where $Prob(D, H)$ denotes the probability that the detector assigns label D to an event when the actual event H have happened. $Prob(\cdot)$ is a function over two random variables, the delay $R(i)$ and the actual event $H(i+1)$, thus we rewrite the Bayes risk using conditional densities, $E[C] = E[E[C|R]]$, and minimize $E[C|R]$ instead. Given a fixed value of RTT, the decision rule is a deterministic function, which maps that value of RTT to one of the two hypotheses. Thus the optimal decision rule has to minimize the Bayes risk for each RTT, if it assigns a loss event for a particular RTT, the Bayes risk is $E[C|R] = \sum_{H \in \{L, \bar{L}\}} C(L, H) Prob(D = L, H|R)$ and if it assigns a reordering event to it, instead, the Bayes risk is $E[C|R] = \sum_{H \in \{L, \bar{L}\}} C(\bar{L}, H) Prob(D = \bar{L}, H|R)$. Thus, after applying the Bayes' rule to invert the conditioning and after doing some algebra manipulation, we can write the optimal decision rule as:

$$\frac{p(R|H=L)P_L}{p(R|H=\bar{L})(1-P_L)} \underset{L}{\overset{\bar{L}}{\gtrless}} \frac{(C(L, \bar{L}) - C(\bar{L}, \bar{L}))}{(C(\bar{L}, L) - C(L, L))} = \eta \quad (1)$$

The symbol $\underset{L}{\overset{\bar{L}}{\gtrless}}$ means that the decision rule will as-

sign a loss event to a particular RTT value R if the left side of Equation 1 is larger than its right side, and it will assign a reordering event otherwise. This inequality is known as the likelihood ratio test. The left side of this inequality are quantities that we will need to estimate for each connection, and the right side of this inequality can be seen as the tunable cost parameter of the detector.

The intuition to understand how the detector uses Equation 1 is as follows. We are given estimates of the two conditional densities ($p(R|H = L)$ and $p(R|H = \bar{L})$), an estimate of the probability of loss P_L , and a cost parameter η . Then when a dupack arrives for a packet i , we measure its RTT, evaluate the conditional density functions for that RTT, and compute the likelihood ratio. If the likelihood ratio is larger than the cost ratio, a packet loss is detected and TCP will trigger fast retransmit, otherwise, TCP does nothing. As η increases the likelihood of a packet loss event will need to grow larger compared to the likelihood of a reordering event in order to detect a packet loss. On the other hand, if we decrease η , then we will favor the inference of packet loss events. So, as we change η , the decision rule is changed, and the detector will behave differently. Later in § IV-C we will show how the performance of the detector changes as we change the cost parameter.

B. Probability Density Function Estimation

Before we move on to evaluate the performance of the type of detectors that we can obtain, we need a way to estimate the conditional probability densities for the two events of loss and reordering. We will explore three techniques, a naive approach using discrete histograms, a non-parametric estimation technique, and a parametric estimation technique.

1) *Histogram*: This is the simplest method to estimate a probability mass function. Let n be the total number of samples taken from a distribution, h be the bin size, and define the function $g(r, y, h)$ as

$$g(r, y, h) = \begin{cases} 1 & \text{if } y - h < r \leq y + h \\ 0 & \text{otherwise} \end{cases}$$

Then, $p(r, h, \mathbf{y})$ is simply:

$$p(r) = \frac{\sum_{i=1}^{i=n} g(r, y_i, h)}{n}$$

2) *Parzen Method*: The Parzen method estimates the conditional densities by summing up a common kernel function $K(\cdot)$ placed at each sample point and evaluated at the point of interest, say r . In this paper we use the Gaussian kernel

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

Let \mathbf{S} be a set of N delay measurements, and let b be the Parzen smooth parameter, also known as bandwidth. The computation of the probability density is then:

$$p(r) = \frac{1}{bN\sqrt{(2\pi)}} \sum_{y \in \mathbf{S}} \exp \frac{-(r-y)^2}{2b^2}$$

3) *Bayesian Method*: One of the disadvantages of the previous methods is that they need to *see* some delay samples before they can make a good estimate of the probability density function. Since most of the connections in our datasets that have a loss or reordering event have very few such events, a great number of connections will not benefit much from these methods. However in the Bayesian method, it is possible to encode a prior belief about these events, thus making it possible to estimate the probability of an event even before seeing any data.

Note that in this case, we are using Bayesian statistics to form inferences about the parameters of the detector; this is distinct from the use of Bayes' rule in the detector itself. The key distinction between the Bayesian method and the previous two methods is that in this case, the parameters to be estimated are assumed to be random variables. Thus they are equipped with probability distributions, and one

can assign them some probability distribution (a *prior*) even before any data is observed. Bayes' rule can then be invoked each time new data arrives, allowing an informed progression from the initial prior toward an updated distribution reflecting what is learned from the data.

To use a Bayesian statistical approach, we assume that the distribution of the RTT is Gaussian, with unknown parameters $\theta = (\mu, \sigma)$, the mean and variance respectively. We express this as a *likelihood function* $p(y|\theta) = N(y; \mu, \sigma^2)$, where y is a real variable of the round-trip delay.

Making the Gaussian assumption for the delay is justified in this work for a number of reasons: First, even if the real distribution of delay is something else, the Gaussian assumption can still be used to capture the location of the mass of that distribution — which is what is most important here, because we just seek a way to distinguish the two hypotheses. Second, the Gaussian is the best that can be done using only the simple statistics of sample mean and variance. And finally, the Gaussian has attractive mathematical properties that provide a computationally efficient way to update the estimate as new data becomes available to TCP during a connection. This last point comes from the fact that the Gaussian has conjugate prior distribution functions³.

The true values of μ and σ^2 are not known a priori. However we believe that, under the hypothesis of packet loss, the mean and the variance will be large; whereas under the hypothesis of packet reordering the mean will be small. This belief is expressed in the form of prior distributions on μ and σ . This is a way of encoding and incorporating our engineering knowledge about the typical behavior of RTT under the two hypotheses. We have chosen the conjugate prior distributions as follows: The mean follows a Normal distribution, $\mu|\sigma^2 \sim N(\mu_0, \sigma^2/\kappa_0)$, and the variance follows an Inverse Chi-square distribution, $\sigma^2 \sim Inv - \chi^2(\nu_0, \sigma_0^2)$. The dependence of the mean on the variance in the prior distribution is also justified by observations of variability of RTTs, which have noted that connections with larger delays experience larger delay variance [13], [15].

A strength of the Bayes method is that it provides a way to use samples of RTT to improve the estimation of the parameters of its distribution. We express the dependence of the parameters μ, σ^2 on the data through a conditional distribution, thus using the Bayes' rule we write the *posterior*

³Formally, if \mathcal{F} is a class of likelihood functions $p(y|\theta)$, and \mathcal{P} is a class of prior distributions for θ , then the class \mathcal{P} is conjugate for \mathcal{F} if $p(\theta|y) \in \mathcal{P}$ for all $p(\cdot|\theta) \in \mathcal{F}$ and $p(\cdot) \in \mathcal{P}$. In general, a conjugate function allows treating additional data incrementally, just by replacing the prior function with the posterior function as new data is seen [20].

distribution as

$$\begin{aligned}
p(\mu, \sigma^2|y) &\propto p(y|\mu, \sigma^2)p(\mu, \sigma^2) \\
&= p(y|\mu, \sigma^2)p(\mu|\sigma^2)p(\sigma^2) \\
&= N - Inv - \chi^2(\mu_n, \sigma_n^2/\kappa_n; \nu_n, \sigma_n^2)
\end{aligned}$$

For this model, it can be shown that

$$\begin{aligned}
\mu_n &= \frac{\kappa_0}{\kappa_0 + n}\mu_0 + \frac{n}{\kappa_0 + n}\bar{y} \\
\kappa_n &= \kappa_0 + n \\
\nu_n &= \nu_0 + n \\
\nu_n\sigma_n^2 &= \nu_0\sigma_0^2 + (n-1)s^2 + \frac{\kappa_0 n}{\kappa_0 + n}(\bar{y} - \mu_0)^2
\end{aligned}$$

where n denotes the number of observations of y that have been made, and the subscript n denotes the parameter estimate after the n^{th} observation of y .

The above equations show that the sufficient statistics are the sample mean \bar{y} and the sample variance s^2 , which are already available to TCP. Notice also how easy it is to incorporate new data, for example, ν_{n+1} is simply equal to $\nu_n + 1$.

So far, we have only a way to estimate a distribution for the parameters of the probability density function $p(y)$. But what is needed by the detector is a way to evaluate this function $p(y)$ at particular values of delay \tilde{y} , according to the parameters that we estimate after seeing data. In order to do so we have to evaluate the following expression $p(\tilde{y}|y) = \int \int p(\tilde{y}|\mu, \sigma^2, y)p(\mu, \sigma^2|y)d\mu d\sigma$. The first of the two factors in the integral is just a Normal distribution, given the two unknown parameters (μ, σ^2) and doesn't depend on y at all. The result of the integration has the form of the Student-t distribution, thus $p(\tilde{y}) = t_{\nu_n}(\mu|\mu_n, (\kappa_n + 1)\sigma_n^2/\kappa_n)$. When no data has been seen yet, we evaluate $p(\tilde{y})$ using the prior distribution, instead of the posterior distribution. They both have the same form, because of the conjugacy property. So we only need to substitute for the initial parameters in the above equation, which are summarized in Table II.

4) *Discussion:* All methods presented use the available data to improve their estimates. One advantage of the Bayesian approach over the others is that it is able to form an estimate of the probability density function even before seeing any samples. Another advantage is that it only needs to keep record of the sample mean and the sample variance of the distribution, instead of all the samples. On the other hand, the advantage of the histogram and Parzen methods is that their distributions are more closely matched to the data, and thus their estimates are more precise.

Parameter	Loss Event	Reordering Event
μ_0	500 ms	100 ms
κ_0	1	1
ν_0	1	1
σ_0^2	20	20

TABLE II
INITIAL PARAMETERS OF PRIOR DISTRIBUTIONS

After presenting the three methods we are ready to compare their performance in respect to detection accuracy in terms of probability of detection (P_D) and probability of false alarm (P_F).

C. Detector Performance

In this section we present the performance of detectors obtained using the three techniques described above. Clearly the performance of a detector will be related to the quality of the estimation of the conditional distributions, thus by comparing the detector performance, we will be able to compare the three techniques for density estimation.

The performance of any detector can be evaluated using the quantities bellow:

$$\begin{aligned}
P_D &\equiv Pr(\text{Choose } loss | loss \text{ True}) \\
P_F &\equiv Pr(\text{Choose } loss | reordering \text{ True})
\end{aligned}$$

Clearly a good detector must have high probability of detection (P_D) and low probability of false alarm (P_F).

By varying the Bayes cost parameter η we obtain a range of settings of the detector that trade off detection probability for false alarm probability. The resulting so-called ROC curves for our three detectors are shown in Fig. 5.

In this figure we plot the weighted average of P_D and P_F for all connections of the BU and PMA datasets. The P_D value of each connection was weighted by the number of loss events of that connection, and the P_F value was weighted by the number of reordering events. The extreme values of the cost parameter η are noted in the figures. As expected, small cost parameters lead to high P_D and also P_F , and large cost parameters do the opposite.

Each figure plots a set of curves that correspond to different modes of using the detector. Since the detector accumulates information about the nature of the conditional delay distributions $p(R|H=L)$ and $p(R|H=\bar{L})$ as loss and reordering events occur, it is reasonable to consider only using the detector after a certain number of events

have been observed. These cases are denoted “train > n”, where the detector was trained until at least “n” loss and “n” reordering events have been sampled. For the case of the Bayesian method it is reasonable to invoke the detector without any training at all so those figures show additional curves for “train = 0.”

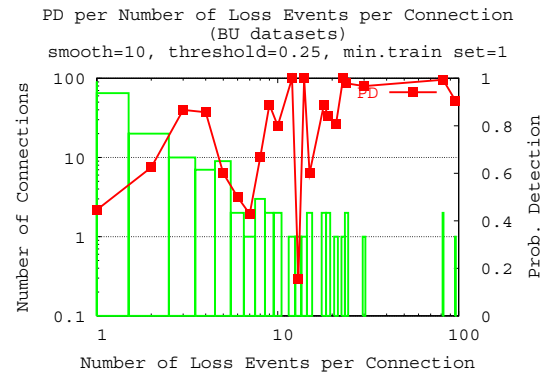
Furthermore, for comparison purposes we plot the case in which the detector has been trained on all events in the connection. This line is denoted “train all.” Note that the “train all” case is not feasible as it uses all data (including future data) in each event classification; however it provides a useful upper bound for the performance of each detector.

The figures show that the detectors work remarkably well. Performance is better on the BU datasets than on the PMA datasets, due to the poor quality of RTT measurements in the PMA datasets (as discussed in § III). Note that the BU datasets, since they are collected immediately in front of a busy server, represent a more realistic set of statistics as seen by a TCP sender.

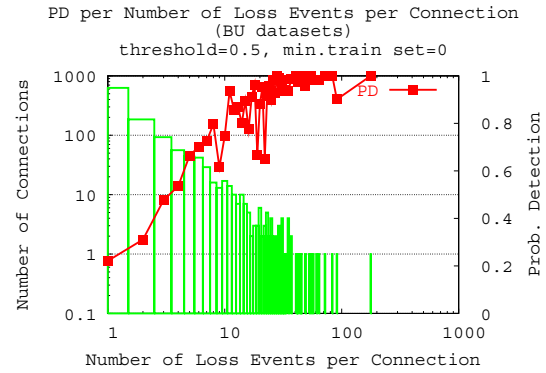
The figures show that the Bayesian method and the Parzen method are roughly comparable, with both methods able (on the BU datasets) to reach a detection probability of 90% with a false alarm probability of only about 20%. Even on the PMA datasets the Parzen method is able to reach a detection probability of 80% with false alarm rate of 40%. The histogram method performs more poorly, as might be expected due to its coarser representation of the conditional probability distributions.

While both the Parzen and Bayesian methods perform approximately equally, it is important to note that the Bayesian method is able to classify many more events since it can be invoked immediately on the first dupack of the connection.

Next, we show a breakdown of how the weighted average P_D values computed above change with the number of samples that a detector uses to estimate the distribution of RTTs. In Fig. 6(a) we break down the connections per number of loss events for which the Parzen detector was invoked. In this plot we used the detector with the Parzen density estimation with smooth parameter $b = 10$ ms, a cost parameter $\eta = 0.25$, and a minimum training set size of 1 sample. We also plot in the second vertical axis what was the average performance in terms of P_D of the detector for those connections. The plots show that the majority of the connections had only one chance to invoke the detector, and P_D was less than 0.5 in those cases. Note that as the number of loss events grows, the performance also improves. This effect is less consistent with Parzen method, which suggests that it may be a less reliable method in practice.



(a) Parzen Method



(b) Bayesian Method

Fig. 6. Break Down of P_D for BU Traces

In Fig. 6(b) we plot the same metrics for the detector using Bayesian density estimation, with initial parameters as given in Table II, with a cost parameter of $\eta = 0.5$ and with no training set. We observe the same kind of improvement in performance as the number of loss events grows. We also notice that the detector is invoked many more times, especially for connections with a low number of loss events, which would not have been enough to train the detector using the Parzen method.

Although we do not show the corresponding plots for P_F , the same type of behavior is observed; as the detector has more data its performance improves, so P_F decreases.

D. Summary

In summary, we find that a Bayes detector can in fact perform packet loss detection with good accuracy. For the Bayesian method, the best performance was obtained was $P_D \sim 0.8$ and $P_F \sim 0.15$, with a cost $\eta = 0.125$, with no training set. For the Parzen method the performance

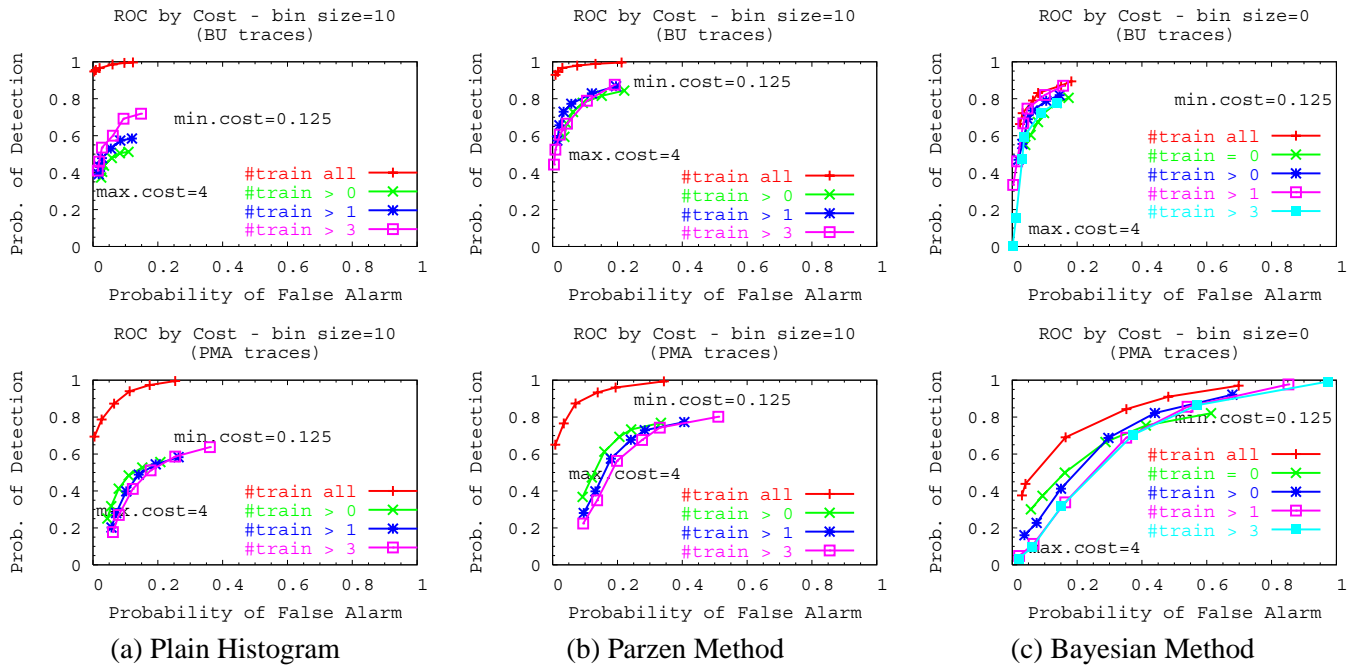


Fig. 5. P_D and P_F of BU Dataset (upper) and PMA Dataset (lower)

was very similar for a cost of $\eta = 0.25$, and bandwidth parameter of 10.

We also observed that the number of events of a connection helps to improve the detector performance, for it has more data to improve the estimation of RTT distribution. On the other hand, connections with lots of events are not as common as connections with as few as 1 or 2 events. For that reason, the Bayesian method may be more suitable for actual deployment in TCP.

The next step is to analyze the improvements possible in TCP performance using such a detector, which we do in the next section.

V. A PERFORMANCE MODEL

In this section we answer our second question, which concerns the impact on TCP performance that is possible when using an early packet loss detector similar to that developed in the last section.

We note that it is not desirable to add too much computation to TCP, so we choose to invoke the detector only when a packet receives its first dupack. As a result, we still have to rely on the timeout mechanism to detect packet losses for which no dupack is generated. Additionally, if three or more dupacks are received by TCP, it will trigger fast retransmit, just as it does currently.

In this setting, the relative significance of P_D and P_F are as follows. A detection will trigger a response to congestion earlier than would have occurred when using

unmodified TCP. Thus it can sometimes avoid the long retransmission timer expiration. A false alarm will trigger a halving of the congestion window that would have not taken place in an unmodified TCP connection. These two effects are in opposition: one increases, the other decreases the throughput of TCP. To properly evaluate this tradeoff requires an accurate model of TCP throughput.

A. A Markov model for TCP with a probabilistic loss detector

To that end we turn to the work described in [2], which created a Markov model for the performance of TCP. The states of the model represent states of TCP, as given by the size of the sending window, the number of unacknowledged packets after a packet loss, etc. The model incorporates TCP's standard congestion avoidance algorithm, with settings for the potential for delayed acks, the number of dupacks that trigger a fast retransmit, and the extent of exponential backoff. Having chosen these TCP characteristics, the model can be used to compute the *sending rate* and the *goodput* of a TCP sender, parameterized by the loss rate (P_L), the maximum receiver window (W), the average round trip delay (RTT), and the base timeout value of the retransmission timer (T_0).

In our work, we have extended this model by integrating a detector of packet loss as described in the previous sections. The added parameters are P_D and P_F , which give the detector performance, and P_R , the reordering

rate, which gives the chance of a packet being reordered.

The key differences between the two models are the following. In our model when TCP receives a dupack sequence, it invokes the detector. If the dupack sequence was caused by a packet loss, then with probability P_D , it will be detected and TCP will trigger fast retransmit. If that dupack sequence was caused by a reordering event, then with probability P_F , the detector will generate a false alarm and TCP will trigger fast retransmit. The other two packet loss inference mechanisms of TCP are still there. If a packet loss does not generate any dupack, the retransmission timer will timeout and TCP will reduce its sending window size to one and retransmit the lost packets, and if three or more dupacks are generated by a packet loss, TCP will trigger fast retransmit. These modifications consist of adding additional transitions to the original Markov chain model. The full specification of our modifications is reported in the following sections.

We first describe the simplifying assumptions of the model, then we give details about the Markov model states and transitions. Finally we present our results.

B. Simplifying Assumptions

Most of the assumptions bellow were extracted from Padye's work[2], but we also add one assumption to the end the list.

- The sender always has data to send, and sends them in back-to-back packets as allowed by its congestion window. Also the corresponding ACKs for the packets which were not lost arrive in a burst within the same RTT. Therefore, after one RTT, TCP has received all ACKs for the packets it had sent and is ready to send another burst of data packets back-to-back. This assumption is supported by simulation and TCP trace analysis evidences found in [21], [13].
- The RTT is fixed and independent of the window size. This assumption was also made in other related works [22], [23].
- Packet losses are independent across rounds, but are correlated within a round. Specifically if a packet is lost in a round, all the remaining packets that fall till the end of the same window are lost. This assumption is justified by the generally adopted drop-tail queuing discipline by many Internet routers, as is discussed in [24], [13], [14].
- Some aspects of TCP were not modeled, namely, slow-start, fast recovery and some subtle differences of specific operating systems implementations. For instance, fast retransmit in Linux is triggered by only two duplicate ACKs, instead of three, as we assume in this paper.

- The receiver uses delayed ACK, that is, the receiver acknowledges every other packet.

We add one assumption more, which has to do with reorderings.

- Packet reordering is independent of packet loss. The justification for this assumption is that these two events have different causes. Losses are caused by buffer overflows as mentioned before; and reorderings are caused by parallel paths and network anomalies, as discussed in [19], [14], [17].

C. States of the Markov Model

We need to introduce the concept of *rounds*. A round starts with the back-to-back transmission of W packets, where W is the current size of the congestion window; and the round ends with the reception of a burst of ACKs in the same RTT. There are two types of rounds, normal and short. In a normal round a full window worth of packets is transmitted; whereas in a short round only a partial window is transmitted. A short round always follows a round in which at least one packet was lost, but not all packets. A round is represented by a state in the Markov model.

Associated with every round there are several random variables that describe its state. The sequence of values taken by these random variables during a TCP conversation is an instance path of a discrete time stochastic process. The transitions from state to state correspond to the actions taken by TCP after each round, during a connection.

The definitions of the sequences of random variables as defined in [2] are reproduced bellow for completeness. We did not add any random variables in our extended model.

- $\{W_i\}_{i=0}^{\infty}$, $1 \leq W_i \leq \mathcal{W}$. W_i is the window size for round i . \mathcal{W} is the maximum receiver window size.
- $\{C_i\}_{i=0}^{\infty}$, $C_i = 0, 1$. C_i allows us to model delayed ACKs. Delayed acks causes the increment of the congestion window by one every two rounds when no packet is lost. $C_i = 0$ indicates first of these two rounds and $C_i = 1$ indicates the second.
- $\{L_i\}_{i=0}^{\infty}$, $0 \leq L_i < W_{i-1}$. L_i is the number of packets lost in the $(i-1)^{st}$ round. If $L_i = 0$ then either no packets were lost in the previous round, or the previous round was a short round and effects of packet loss have already been accounted for via a fast retransmit or a timeout. $L_i > 0$ indicates that this is a short round, as a result of L_i packets being lost in the previous round. For short rounds C_i is always equal to 0.

- $\{T_i\}_{i=0}^{\infty}$, $0 \leq T_i \leq 7$. T_i denotes whether the connection is in a timeout state in round i . If $T_i = 0$ then the transmission of packets in round i was not a result of a retransmission timer expiration. On the other hand, if $T_i = 1, \dots, 7$, then a timeout did occur in the previous round, and the retransmission timer backoff multiplier was $1, 2, 4, \dots, 64$, respectively.
- $\{R_i\}_{i=0}^{\infty}$, $RTT \leq R_i \leq 64 * T_0$. R_i denotes the duration of round i . RTT denotes the (average) round trip time, and T_0 denotes the base timeout value. During a timeout sequence R_i can have one of the values of $T_0, 2T_0, \dots, 64T_0$, as explained earlier. All other rounds have duration equal to RTT .
- $\{N_i\}_{i=0}^{\infty}$, $1 \leq N_i \leq \mathcal{W}$. N_i denotes the number of packets transmitted in round i .
- $\{M_i\}_{i=0}^{\infty}$, $1 \leq M_i \leq \mathcal{W}$. M_i denotes the number of packets successfully received in round i .

In what follows, we will use only the first four variables $(W, C, L, T)_i$ to uniquely define a Markov state. Because these variables have finite values, the number of Markov states is also finite on the order of (\mathcal{W}^2) , more precisely, the number of states is equal to $\mathcal{W} + \mathcal{W} - 1 + \mathcal{W} * (\mathcal{W} - 1)/2 + 7$.

D. State Transitions

Now we need to define the transition probabilities of the Markov chain. We refer the reader to [2] for a detailed description of what Padhye et al. did, although we give here a high level description of their work along with a detailed description of the modifications we did to extend their model with a packet loss detector.

Rounds without loss nor reordering. In this case, a window worth of packets is transmitted successfully in one round trip time, and in every other consecutive round the congestion window is increased by one up to the limit given by \mathcal{W} , the maximum receiver window. More formally

$$\begin{aligned}
W_i &= 1 \dots \mathcal{W} \rightarrow W_{i+1} = \min(\mathcal{W}, W_i + C_i) \\
C_i &= 0, 1 \rightarrow C_{i+1} = 1 - C_i \\
L_i &= 0 \rightarrow L_{i+1} = 0 \\
T_i &= 0 \rightarrow T_{i+1} = 0 \\
R_i &= RTT \rightarrow R_{i+1} = RTT \\
N_i &= 1 \dots \mathcal{W} \rightarrow N_{i+1} = W_i \\
M_i &= 1 \dots \mathcal{W} \rightarrow M_{i+1} = W_i
\end{aligned} \tag{1}$$

Note the way that C_i is updated to give the effect of a receiver with delayed ACKs. We also impose the following constraint on C_i : if $W_i = \mathcal{W}$, then $C_i = 0$. This captures the fact that the window size cannot grow beyond \mathcal{W} .

The probability for this type of transitions is

$$q = ((1 - P_L)(1 - P_R))^{W_i}$$

where P_L is the probability of a packet being lost (the loss rate) and P_R is the probability of a packet being reordered. $P_R = 0$ when $W_i = 1$, since packet reordering is impossible with a window of size one.

Rounds without loss but with reordering. If no packet is lost, but one or more packets are reordered, a window worth of packets is transmitted successfully in one round trip time. However the variables C_i and W_i may be different because TCP would trigger fast retransmit if the detector generated a false alarm. Therefore two transitions are possible. First if the detector does generate a false alarm then we have the same as Transition 1 with probability

$$q = (1 - P_L)^{W_i}(1 - P_F)(1 - (1 - P_R)^{W_i})$$

Second, if the detector generates a false alarm we have the following transition:

$$\begin{aligned}
W_i &= 2 \dots \mathcal{W} \rightarrow W_{i+1} = W_i/2 \\
C_i &= 0, 1 \rightarrow C_{i+1} = 0 \\
L_i &= 0 \rightarrow L_{i+1} = 0 \\
T_i &= 0 \rightarrow T_{i+1} = 0 \\
R_i &= RTT \rightarrow R_{i+1} = RTT \\
N_i &= 1 \dots \mathcal{W} \rightarrow N_{i+1} = W_i \\
M_i &= 1 \dots \mathcal{W} \rightarrow M_{i+1} = W_i
\end{aligned} \tag{2}$$

with the corresponding transition probability:

$$q = (1 - P_L)^{W_i}(P_F)(1 - (1 - P_R)^{W_i})$$

Rounds with packet loss but no reordering. When some packets are lost but not all, and no packets are reordered, there is a transition to a short round. In terms of the state variables we have:

$$\begin{aligned}
W_i &= 2 \dots \mathcal{W} \rightarrow W_{i+1} = W_i - L_{i+1} \\
C_i &= 0, 1 \rightarrow C_{i+1} = 0 \\
L_i &= 0 \rightarrow L_{i+1} = num_Lost \\
T_i &= 0 \rightarrow T_{i+1} = 0 \\
R_i &= RTT \rightarrow R_{i+1} = RTT \\
N_i &= 1 \dots \mathcal{W} \rightarrow N_{i+1} = W_i \\
M_i &= 1 \dots \mathcal{W} \rightarrow M_{i+1} = W_{i+1}
\end{aligned} \tag{3}$$

Where L_i is the random variable that defines the number of lost packets in round i . The corresponding transition probability is

$$q = P_L((1 - P_F)(1 - P_R))^{W_{i+1}}$$

If all packets are lost, TCP will have to wait for a timeout and then will cut window size to 1. The corresponding transition is:

$$\begin{aligned}
W_i &= 1 \dots \mathcal{W} \rightarrow W_{i+1} = 1 \\
C_i &= 0, 1 \rightarrow C_{i+1} = 0 \\
L_i &= 0 \rightarrow L_{i+1} = 0 \\
T_i &= 0 \rightarrow T_{i+1} = 1 \\
R_i &= RTT \rightarrow R_{i+1} = T_0 \\
N_i &= 1 \dots \mathcal{W} \rightarrow N_{i+1} = W_i \\
M_i &= 1 \dots \mathcal{W} \rightarrow M_{i+1} = 0
\end{aligned} \tag{4}$$

And the transition probability is $q = P_L$.

Rounds with packet loss and reordering. Finally, it is possible to have lost and reordered packets in the same round. In this case, the reordered packet must come before the lost packet, because, in our model, all packets from the first lost packet to the end of the congestion window are lost in a loss event. The duration of such round is RTT , and from all packets from the congestion window which were sent, only $W - L$ make it to the receiver. If the detector classify the reordered packet as not lost, then TCP behaves as in case 3 with probability

$$q = P_L(1 - (1 - P_R)^{W_{i+1}})(1 - P_F)(1 - P_L)^{W_{i+1}}$$

But if the detector classify the reordered packet as being lost, then TCP will trigger fast retransmit for the reordered packet:

$$\begin{aligned} W_i &= 3 \dots \mathcal{W} \rightarrow W_{i+1} = \min(W_i - L_{i+1}, \frac{W_i}{2}) \\ C_i &= 0, 1 \rightarrow C_{i+1} = 0 \\ L_i &= 0 \rightarrow L_{i+1} = num_lost \\ T_i &= 0 \rightarrow T_{i+1} = 0 \\ R_i &= RTT \rightarrow R_{i+1} = RTT \\ N_i &= 1 \dots \mathcal{W} \rightarrow N_{i+1} = W_i \\ M_i &= 1 \dots \mathcal{W} \rightarrow M_{i+1} = W_i - L_{i+1} \end{aligned} \quad (5)$$

With transition probability

$$q = P_L(1 - (1 - P_R)^{W_{i+1}})(P_F)(1 - P_L)^{W_{i+1}}$$

If fast retransmit is triggered because the detector generated a false alarm, the window is cut to half, however, if more than half of the packets were lost, the extra losses must be discounted from the next window size as well, as shown in the transition above.

Short rounds. In a short round if at least three packets are sent and make it to the receiver, then TCP triggers fast retransmit. Otherwise, the sender would have to wait for a timeout, unless the detector detects the loss using the first dupack. This feature has the potential to avoid waiting for a timeout, and improve TCP performance, as we show in the results. Note that reordering events have no effect on a short round because duplicate ACKs are being generated for the packet lost in the previous round. Next we show the details of three cases that can happen in a short round.

First, if all packets in a short round are lost, or less than three packets make it to the receiver and the detector fails to detect the packet loss with the first dupack, a timeout is inevitable. In this case

$$\begin{aligned} W_i &= 1 \dots (\mathcal{W} - 1) \rightarrow W_{i+1} = 1 \\ C_i &= 0 \rightarrow C_{i+1} = 0 \\ L_i &= 1 \dots (\mathcal{W} - 1) \rightarrow L_{i+1} = 0 \\ T_i &= 0 \rightarrow T_{i+1} = 1 \\ R_i &= RTT \rightarrow R_{i+1} = T_0 - RTT \\ N_i &= 1 \dots \mathcal{W} \rightarrow N_{i+1} = W_i \\ M_i &= 1 \dots \mathcal{W} \rightarrow M_{i+1} = exp_rcvd \end{aligned} \quad (6)$$

where $L_i + W_i \leq \mathcal{W}$, and exp_rcvd is the expected number of packets that arrive at the receiver in a short round. This value depends on the window size and on the probability of loss, according to following formulas:

$$\begin{aligned} exp_rcvd &= (1 - P_L) && \text{for } W_i = 1 \\ exp_rcvd &= (P_L(1 - P_L) + 2(1 - P_L)^2) && \text{for } W_i = 2 \\ exp_rcvd &= \frac{\sum_{k=1}^{k=2} k P_L(1 - P_L)^k}{\sum_{k=1}^{k=2} P_L(1 - P_L)^k} && \text{for } W_i > 2 \end{aligned}$$

The transition probabilities are

$$\begin{aligned} q &= P_L && \text{for } L_i = W_i \\ q &= (1 - P_D)(1 - P_L) && \text{for } 1 \leq W_i \leq 2 \\ q &= \sum_{k=1}^{k=2} (1 - P_D) P_L (1 - P_L)^k && \text{for } 3 \leq W_i < \mathcal{W} \end{aligned}$$

Next, if only one or two packets arrive at the receiver and are acknowledge, the detector will have a chance to detect the lost packet correctly. In this case, TCP will do a fast retransmit, and cut the window in half.

$$\begin{aligned} W_i &= 3 \dots (\mathcal{W} - 1) \rightarrow W_{i+1} = (W_i + L_i)/2 \\ C_i &= 0 \rightarrow C_{i+1} = 0 \\ L_i &= 1 \dots (\mathcal{W} - 1) \rightarrow L_{i+1} = 0 \\ T_i &= 0 \rightarrow T_{i+1} = 0 \\ R_i &= RTT \rightarrow R_{i+1} = RTT \\ N_i &= 1 \dots \mathcal{W} \rightarrow N_{i+1} = W_i \\ M_i &= 1 \dots \mathcal{W} \rightarrow M_{i+1} = exp_rcvd \end{aligned} \quad (7)$$

Again, the expected number of packets that arrive at the receiver is the same as in the previous case and the transition probabilities are the following

$$\begin{aligned} q &= P_D(1 - P_L) && \text{for } 1 \leq W_i < 3 \\ q &= \sum_{k=1}^{k=2} (P_D) P_L (1 - P_L)^k && \text{for } 3 \leq W_i < \mathcal{W} \end{aligned}$$

Last, if at least three packets arrive at the receiver and are acknowledge, TCP will do a fast retransmit and cut the congestion window to half its size. Therefore

$$\begin{aligned} W_i &= 3 \dots (\mathcal{W} - 1) \rightarrow W_{i+1} = (W_i + L_i)/2 \\ C_i &= 0 \rightarrow C_{i+1} = 0 \\ L_i &= 1 \dots (\mathcal{W} - 1) \rightarrow L_{i+1} = 0 \\ T_i &= 0 \rightarrow T_{i+1} = 0 \\ R_i &= RTT \rightarrow R_{i+1} = RTT \\ N_i &= 1 \dots \mathcal{W} \rightarrow N_{i+1} = W_i \\ M_i &= 1 \dots \mathcal{W} \rightarrow M_{i+1} = exp_rcvd \end{aligned} \quad (8)$$

The expected number of packets that arrive at the receiver is computed as follows:

$$exp_rcvd = \frac{\sum_{k=3}^{k=W_i-1} k P_L (1 - P_L)^k + W_i (1 - P_L)^{W_i}}{\sum_{k=3}^{k=W_i-1} P_L (1 - P_L)^k + (1 - P_L)^{W_i}}$$

And the transition is:

$$q = \sum_{k=3}^{k=W_i-1} P_L (1 - P_L)^k + (1 - P_L)^{W_i} \quad \text{for } 3 \leq W_i < \mathcal{W}$$

Round of exponential backoff. Finally, the last group of transitions captures TCP exponential backoff rounds. In these transitions, only one packet transmission is attempted. If it fails TCP doubles the retransmission timer timeout value, up to 64 times the base value. The state transitions in this case is:

$$\begin{aligned}
W_i &= 1 & \rightarrow & W_{i+1} = 1 \\
C_i &= 0 & \rightarrow & C_{i+1} = 0 \\
L_i &= 0 & \rightarrow & L_{i+1} = 0 \\
T_i &= 1 \dots 7 & \rightarrow & T_{i+1} = \min(7, T_i + 1) \\
R_i &= 2^{(T_i-1)}T_0 & \rightarrow & R_{i+1} = 2^{(T_{i+1}-1)}T_0 \\
N_i &= 1 \dots \mathcal{W} & \rightarrow & N_{i+1} = 1 \\
M_i &= 0 & \rightarrow & M_{i+1} = 0
\end{aligned} \tag{9}$$

With transition probability $q = P_L$.

However if the packet is successfully transmitted:

$$\begin{aligned}
W_i &= 1 & \rightarrow & W_{i+1} = 2 \\
C_i &= 0 & \rightarrow & C_{i+1} = 0 \\
L_i &= 0 & \rightarrow & L_{i+1} = 0 \\
T_i &= 1 \dots 7 & \rightarrow & T_{i+1} = 0 \\
R_i &= 2^{(T_i-1)}T_0 & \rightarrow & R_{i+1} = RTT \\
N_i &= 1 & \rightarrow & N_{i+1} = 1 \\
M_i &= 0 & \rightarrow & M_{i+1} = 1
\end{aligned} \tag{10}$$

with probability $q = 1 - P_L$.

All other transitions not described here should have an associated probability of zero, as they are infeasible according to TCP behavior.

E. Results

We used the model developed above to evaluate the effects of correct detection and false alarm on TCP throughput.

In our analysis, we used $\mathcal{W} = 20$, $T_0 = 2.45s$, and $RTT = 250ms$ in the Markov model, which are the same settings used for validation of the model in [2]. We compared the following four scenarios:

- 1) **“Regular TCP”** TCP Reno with $\text{dupthresh} = 3$. This means that TCP waits for 3 dupacks before triggering fast retransmit.
- 2) **“TCP with dacks=1”** TCP Reno, with $\text{dupthresh} = 1$. This is equivalent to a detector with $P_D = P_F = 1$.
- 3) **“TCPB $P_D = 0.8, P_F = 0.15$ ”** TCP equipped with the Bayes detector, whose performance is $P_D = 0.8, P_F = 0.15$. These values were found to be achievable in practice as described in § IV-D. In this case we set $\text{dupthresh} = 3$,
- 4) **“TCPB perfect”** TCP Bayes with a perfect detector, *i.e.*, $P_D = 1, P_F = 0$, and $\text{dupthresh} = 3$.

The overall performance of the different variants depends on the relative frequency of packet loss vs. packet reordering. Thus we first study the performance under varying loss rates, and then under varying reordering rates.

Figure 7(a) shows the performance of the four variants as a function of loss rate (where reordering probability is fixed at 0.2%). The plot on the left of the figure shows TCP goodput in absolute terms while the plot on the right shows goodput relative to unmodified TCP. We can see that all three variants lead to considerable improvement over TCP. TCP with $\text{dacks}=1$ performs nearly as well as perfect detector. This is because nearly all dupacks are due to packet losses, so the quick response provided by setting dupthresh to 1 is appropriate. However it is also important to note that in this setting, the performance of the realistic detector is almost as good as that of the perfect detector. The plot shows that a realistic detector can achieve an improvement in TCP goodput of up to 25% over standard TCP.

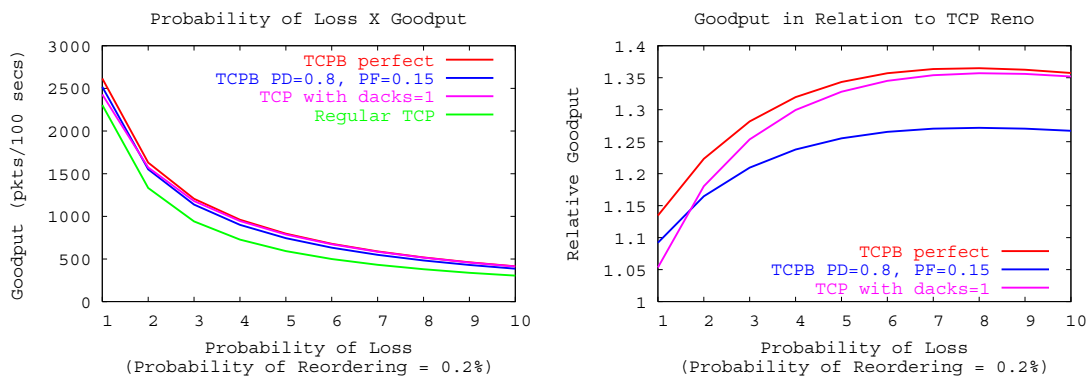
The problem with setting dupthresh to 1, and the reason for preferring the Bayes detector, becomes clear in Figure 7(b). In this figure, we keep the loss rate at 5% and vary the packet reordering rate. We see that the performance of TCP with $\text{dupthresh} = 1$ drops sharply off as the amount of packet reordering increases. This is because TCP is decreasing its congestion window unnecessarily. In contrast, the performance of the practical Bayes detector remains close to its level of 25% improvement over TCP across the range of packet reordering rates.

VI. CONCLUSION

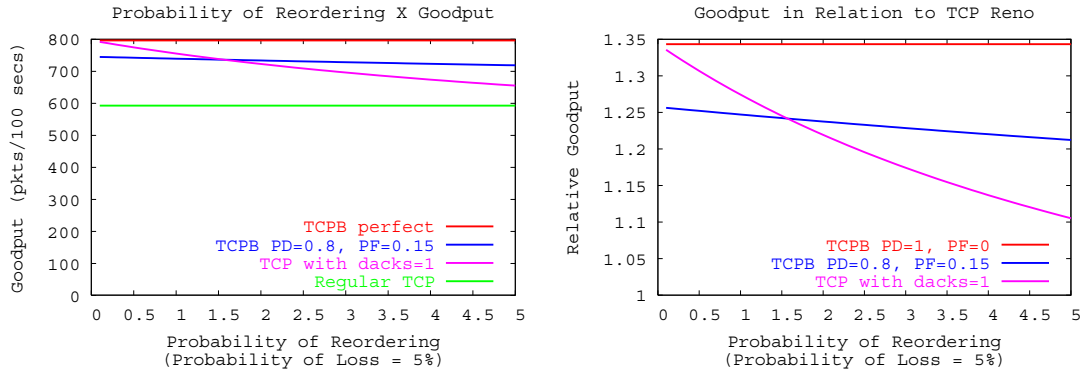
In this paper we start from the basic observation that TCP’s packet loss detection mechanism is in fact solving an inferencing problem. We then ask whether a more effective inferencing procedure is possible, and what the impact of using such a procedure might be.

We develop a packet loss detector using a Bayesian framework. We show that such a detector can profitably make use of round trip time to guide the inferencing process. Using traces taken from a variety of locations, we show that a Bayes detector can achieve a probability of detection above 80% and probability of false alarm below 20%. We evaluate three potential realizations of such a detector and find that one based on Bayesian statistical estimation performs quite well, and has the appealing property that it can be used for each duplicate ACK received, including the first one.

These results encourage us to investigate the improvements to TCP’s performance that can be possible using such a detector. We construct an analytical model of TCP that incorporates a probabilistic loss detector mechanism on top of the existing TCP loss detection mechanisms. Using this model we show that TCP performance can be improved by as much as 25% with realistic detection and false alarm probabilities.



(a) Effect of Loss Rate



(b) Effect of Reordering Rate

Fig. 7. (Relative) Goodput versus Loss and Reordering Rates

Our results suggest that the loss detection problem faced by TCP can in fact be addressed in a formal way, and that the result of doing so can be a significant improvement to TCP’s performance.

REFERENCES

- [1] Vern Paxson and Mark Allman, “Computing tcp’s retransmission timer,” RFC 2988, <http://www.faqs.org/rfcs/rfc2988.html>, November 2000.
- [2] Jitendra Padhye, Victor Firoiu, and Don Towsley, “A stochastic model of tcp reno congestion avoidance and control,” Tech. Rep., U. Massachusetts Amherst Dept. of Computer Science, 1999.
- [3] Lawrence S. Brakmo, Sean W. O’Malley, and Larry L. Peterson, “Tcp vegas: new techniques for congestion detection and avoidance,” in *Proceedings of the conference on Communications architectures, protocols and applications*. 1994, pp. 24–35, ACM Press.
- [4] Jun Liu, Ibrahim Matta, and Mark Crovella, “End-to-end inference of loss nature in a hybrid wired/wireless environment,” in *Proceedings of WiOpt’03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003.
- [5] Dhiman Barman and Ibrahim Matta, “Effectiveness of loss labeling in improving tcp performance in wired/wireless networks,” in *Proceedings of 10th IEEE International Conference on Network Protocols (ICNP’02)*, 2002.
- [6] Sally Floyd, Tom Henderson, and Andrei Gurtov, “The newreno modification to tcp’s fast recovery algorithm,” RFC 3782, <http://www.faqs.org/rfcs/rfc3782.html>, April 2004.
- [7] Stephan Bohacek, Joao P. Hespanha, Junsoo Lee, Chansook Lim, and Katia Obraczka, “TCP-PR: TCP for persistent packet reordering,” in *23rd International Conference on Distributed Computing Systems*, May 2003, pp. 222–231.
- [8] Pasi Sarolahti, Markku Kojo, and Kimmo Raatikainen, “F-rtto: an enhanced recovery algorithm for tcp retransmission timeouts,” *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 2, pp. 51–63, 2003.
- [9] Kostas Pentikousis, “TCP in wired-cum-wireless environments,” *IEEE Communications Surveys*, vol. 3, no. 4, 4th Quarter 2000.
- [10] Reiner Ludwig and Randy H. Katz, “The eifel algorithm: making tcp robust against spurious retransmissions,” *SIGCOMM Comput. Commun. Rev.*, vol. 30, no. 1, pp. 30–36, 2000.
- [11] Feng Wang and Yongguang Zhang, “Improving tcp performance over mobile ad-hoc networks with out-of-order detection and response,” in *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*. 2002, pp. 217–225,

- [12] Ethan Blanton and Mark Allman, "On making tcp more robust to packet reordering," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 1, pp. 20–30, 2002.
- [13] V. Paxson, *Measurements and Analysis of End-to-End Internet Dynamics*, Ph.D. thesis, University of California, Berkeley, April 1997.
- [14] Vern Paxson, "End-to-end internet packet dynamics," *IEEE/ACM Transactions on Networking*, vol. 7, no. 3, pp. 277–292, 1999.
- [15] Jay Aikat, Jasleen Kaur, F. Donelson Smith, and Kevin Jeffay, "Variability in tcp round-trip times," in *Proceedings of the 2003 ACM SIGCOMM conference on Internet measurement*. 2003, pp. 279–284, ACM Press.
- [16] Saad Biaz and Nitin H. Vaidya, "Is the round-trip time correlated with the number of packets in flight?," in *Proceedings of the 2003 ACM SIGCOMM conference on Internet measurement*. 2003, pp. 273–278, ACM Press.
- [17] Sharad Jaiswal, Gianluca Iannaccone, Christophe Diot, Jim Kurose, and Don Towsley, "Measurement and classification of out-of-sequence packets in a tier-1 ip backbone," in *Proceedings of INFOCOM 2003*, March 2003, vol. 2, pp. 1199–1209.
- [18] Renata Teixeira, Keith Marzullo, Stefan Savage, and Geoffrey M. Voelker, "In search of path diversity in isp networks," in *Proceedings of the 2003 ACM SIGCOMM conference on Internet measurement*. 2003, pp. 313–318, ACM Press.
- [19] Jon C. R. Bennett, Craig Partridge, and Nicholas Shectman, "Packet reordering is not pathological network behavior," *IEEE/ACM Trans. Netw.*, vol. 7, no. 6, pp. 789–798, 1999.
- [20] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin, *Bayesian Data Analysis*, Texts in Statistical Science. Chapman & Hall/CRC, 1995.
- [21] Kevin Fall and Sally Floyd, "Simulation-based comparisons of Tahoe, Reno and SACK TCP," *Computer Communication Review*, vol. 26, no. 3, pp. 5–21, July 1996.
- [22] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose, "Modeling TCP throughput: a simple model and its empirical validation," in *Proceedings of SIGCOMM 1998*, 1998.
- [23] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm," *Computer Communication Review*, vol. 27, no. 3, July 1997.
- [24] Jean-Chrysostome Bolot, "End-to-end packet delay and loss behavior in the Internet," in *Proceedings of SIGCOMM 1993*. ACM SIGCOMM, August 1993, vol. 2, pp. 289–298.