June 2002

Predicting TCP Throughput From Non-invasive Network Sampling

Mukul Goyal
Ohio State University

Roch A. Guérin
University of Pennsylvania, guerin@acm.org

Raju Rajan
Ipsum Networks

Predicting TCP Throughput From Non-invasive Network Sampling

Abstract

In this paper, we wish to derive analytic models that predict the performance of TCP flows between specified end-points using routinely observed network characteristics such as loss and delay. The ultimate goal of our approach is to convert network observables into representative user and application relevant performance metrics.The main contributions of this paper are in studying which network performance data sources are most reflective of session characteristics, and then in thoroughly investigating a new TCP model based on [1] that uses non-invasive network samples to predict the throughput of representative TCP flows between given end-points.

Predicting TCP Throughput From Non-invasive Network Sampling

Mukul Goyal
Computer & Information Science Dept
The Ohio State University
Columbus, OH 43210
mukul@cis.ohio-state.edu

Roch Guerin
Department of Electrical Engineering
University of Pennsylvania
Philadelphia, PA 19104
guerin@ee.upenn.edu

Raju Rajan
CTO, Ipsum Networks
461 N. 3rd Street
Philadelphia, PA 19123
raju@ipsumnetworks.com

Abstract— **In this paper, we wish to derive analytic models that predict the performance of TCP flows between specified end-points using routinely observed network characteristics such as loss and delay. The ultimate goal of our approach is to convert network observables into representative user and application relevant performance metrics. The main contributions of this paper are in studying which network performance data sources are most reflective of session characteristics, and then in thoroughly investigating a new TCP model based on [1] that uses non-invasive network samples to predict the throughput of representative TCP flows between given end-points.**

I. Introduction

As the Internet continues to evolve into the dominant commercial communications infrastructure, the need for service verification and quality monitoring is also increasing. This is reflected in the emergence of *Service Level Agreements* (SLAs) between providers and their (business) customers, which specify various levels of service guarantees that are to be met. Service guarantees are often specified using aggregate measures, e.g., a minimum bandwidth guarantee between sites, but the performance measures of real interest are usually the level of performance that individual users and applications experience. As TCP [2] traffic represents about 83% of the packets and 91% of the bytes on the Internet[1], monitoring TCP performance is a key step towards predicting, monitoring and analyzing network performance from an end-user perspective.

In this paper, we wish to derive analytic models that predict the performance of TCP flows between specified end-points using routinely observed network characteristics such as loss and delay. The ultimate goal of our approach is to convert network observables into representative user and application relevant performance metrics. In doing so, we seek to draw upon three different approaches to performance characterization - *network sampling*, *application sampling* and *analytic modeling*.

How does our approach relate to each of the above and to previous works on similar topics?

Service providers routinely sample performance and fault data in the network, and many of them advertise delay and loss information between city pairs as one form of feedback to customers. For scalability reasons, carriers use *non-invasive* sources; for example, loss and throughput data available from router SNMP MIBs, or delay estimates obtained from probes exchanged between routers; rather than *invasive* information obtained from stateful inspection of individual flows. In order for this data to be meaningful to individual users, it must be transformed into user/application specific metrics. Undertaking this transformation in the context of TCP is the focus of this paper.

An alternative approach to user relevant monitoring is through application sampling. Several vendors offer SLA assurance software for periodically initiating and measuring web transfers, file transfer, email and other services between predefined end-points. While quite simple to implement and use, application sampling suffers from a number of limitations. Such sampling cannot distinguish between different factors that contribute to application performance. For instance, sampling web download performance composites DNS lookup times, web server latencies, as well as congestion in different network segments. If the hosting provider is different from the network provider, it is difficult to decide which of them is responsible for poor performance. The lack of an underlying model and inability to decompose result in unnecessary over-sampling. For instance, web page download times cannot be re-used to predict FTP performance; nor can information be shared between web downloads passing through the same bottleneck. Rather than treat the network as a "black box", our approach allows information sharing between carrier and customer, with scalability and infrastructure savings as primary advantages.

Of particular relevance to this paper is the extensive literature on analytic modeling of TCP behavior targeting different environments [3], [4], [5], [1], [6], [7], [8], [9], [10]. A direct approach to our goal would be to use network data to estimate model parameters. In doing so, we need to consider two related issues. How should the network be sampled? By polling the MIBs in the routers or by probing the network with end-to-end pings, for instance? The second issue is that of transforming the raw samples into into input parameters of the chosen model. We ruled out several analytic models because of the infeasibility of estimating their input parameters. For instance, some models use parameters such as "loss events" (the event of a TCP flow reducing its congestion window) that cannot be estimated from non-invasive network observables; some others depending on loss correlations between successive packets of a TCP session can work only with a mechanism that inspects packets on a per-session granularity; yet others are tailored to restricted environments. Given such constraints, we tried in-

[1] See http://www.caida.org/outreach/papers/ for recent data.

stead to pick an analytically sound and well-tested model, and see how far we could go in retro-fitting network observables to model inputs. The model proposed in [1] (henceforth called the *Amherst Model*), turned out to be a good starting point for our investigations. While the straightforward approach of directly estimating Amherst Model parameters from network samples produced inaccurate results (see Section III), we were able to produce a more reliable variant based directly on network observables.

A. Contributions

The motivation for this paper is the need for user relevant performance metrics based on observable network data. Our work is a modest but necessary step towards this ambitious goal – we investigate the selection and transformation of network data into metrics based on the predicted throughput of long-lived TCP flows.

The main contributions of this paper are two-fold. First we investigate how to sample the network to obtain reliable and accurate estimates of important network characteristics like Round-trip time (RTT) and loss rates. In this respect, it is important to understand how and when network observables are good estimators of TCP session observables. The importance of this contribution is evident as even the most accurate TCP model will be rendered ineffective if the required input parameter values can not be determined accurately. The second contribution is the development of a model capable of predicting steady state TCP throughput reasonably accurately, using only the input parameters that can be easily obtained in a non-invasive fashion. As mentioned earlier, the new model builds on the Amherst model of [1], but includes a number of non-trivial enhancements that improved overall model accuracy. The model was evaluated for a wide range of configurations using both testbed experiments and simulations and found to predict steady state TCP throughput reasonably well, at least in scenarios where the network monitoring information available to the model was itself reasonably accurate.

The rest of this paper is structured as follows. Section II discusses the feasibility and limitations of sampling network information. It reviews the various non-invasive methods we rely on, as well as their ability to estimate network properties accurately in variety of settings. Section III first examines the feasibility of transforming network observables into parameter estimates in the Amherst model. The resulting inaccuracies in TCP throughput prediction lead us to develop a new model better suited to the use of network information acquired in a non invasive manner. The performance of the new model is investigated for a wide range of network conditions and loss rates in Section IV, while Section V summarizes the findings of the paper.

II. *Non-invasive* ESTIMATION OF NETWORK PROPERTIES

The information needed to predict the throughput of TCP flows consists of *session level* information such as packet size, retransmission timer granularity, maximum congestion window size, use of delayed acks, etc., and of *network level* information such as sequence of successive round trip times and or losses. We are interested in the latter category, as session level information can be regarded as chosen *a priori* while defining representative flows. Further, analytic models of TCP performance use some simplified characteristic of the delay and loss sequences as predictors, rather than the sequences themselves. In this section, we examine loss and delay characteristics that can be derived from non-invasive network sampling, with a view to selecting and developing models that use them.

A. Network based parameters: What loss characteristics to estimate?

In predicting TCP throughput, there are a number of "plausible" loss models one can envision. The simplest one assumes random losses, i.e., packet losses are modeled as a sequence of independent Bernoulli trials with parameter equal to the *loss rate*. Such a parameter is easily estimated from the ratio of the numbers of lost and transmitted packets. More sophisticated models assume that losses are correlated in nature. Generally, these models assume that losses occur with probability p as long as the system is not "congested", and that they occur with probability $p' \geq p$ after the "first" loss, which is used to identify the start of a congestion period. These models require the estimation of two parameters, p and p', as well as the determination of the duration of a congestion period once it has started. Neither of these tasks is straightforward, especially if they are to be performed using non-invasive procedures. Hence in this paper, we have presented only the random loss model. The results indicate that the model provides fairly accurate estimates of long term TCP throughput for a wide range of network conditions and loss rates.

B. Techniques for Non-invasive Estimation: Polling and Probing

This section focuses on non-invasive estimation procedures and their use in estimating parameters of interest in the context of TCP throughput prediction. This means that rather than examining *tcpdump* traces or performing flow level monitoring to estimate loss probabilities, we would like to rely instead on information that is routinely gathered using basic network probing and polling mechanisms. The characteristics as well as the pros and cons of these two mechanisms are reviewed next.

Probing: Probes can easily be implemented using existing mechanisms such as *ping* packets sent from ingress towards egress routers. The ratio of probe packets lost and sent gives an estimate of loss rate while each reply to the ping probe provides a sample for the RTT. RTT samples obtained by ping probes can be used to maintain *smoothed RTT (srtt)* and *smoothed mean deviation in RTT (rttvar)* values [11], [12]. These values along with that of clock granularity (G) can be used to estimate 'first' retransmission timeout value ($T0$) using the well-known formula [11], [13]:

$$T0 = srtt + max(G, 4 \times rttvar) \tag{1}$$

If the retransmission timeout value comes out to be less than 1 second, it is rounded up to 1 second [11]. Hence, it is not crucial to have a very good estimate for *rttvar* unless it will make significant difference to *T0* values.

The main advantage of probing methods is that they sample a complete network segment. As a result, they can provide direct estimates of end-to-end loss probabilities and round-trip times. However, our experiments suggested that while probing methods provide a good estimate of RTT (and timeout durations), the performance is not quite satisfactory for estimating loss rates as observed by the TCP flows [14]. (1) probe packets do not *sample* network queues in the same fashion as TCP flows. We tried to rectify this problem by emulating the bursty or 'ack paced' nature of TCP flows in our probing applications. Specifically, we experimented with a *back-to-back* and an *ack-paced* probing algorithm. *Back-to-back* probing involves sending all the packets in a single round of probing back to back while in *ack-paced* probing the timing between probe replies determines the timing between probe packets in next round of probing. However, these specialized probing applications resulted in only a marginal improvement in the loss rate estimates. (2) Probe loss estimates are significantly affected by the number of packets in a round of probing, the inter-probe delay, the frequency of probing rounds and the packet size used in probes. (3) Loss estimates obtained from probes converge slowly. (4) Probe packets do not always follow the same path or get the same treatment as the data packets.

In figure 1, we show sample simulation results regarding the performance of the probing techniques in estimating loss rates for a set of TCP flows passing through two congested routers with RED [15] buffers. In these simulations, the probing application sent 4 probes (of same size as the packets of TCP flows) in each round of probing and was frequent enough to consume 0.5% of the bottleneck link bandwidth. Other simulation details can be seen later in the paper in section IV-B. It can be observed that *ack-paced* probing provides much better estimates than *back-to-back* probing but still the performance is far from satisfactory. The figure also shows that the loss rate estimates obtained from SNMP MIBs are quite good. These and other similar results led us to conclude that it is not easy to obtain accurate estimates of network loss rates *as observed by TCP flows* using probing methods. As we discuss next, a much better job can be done by polling SNMP MIBs on the routers.

Polling: This refers to the periodic querying of the SNMP MIBs maintained in the routers to retrieve performance data. For example, the *Interfaces* table of SNMP MIB-II [16] can provide information regarding the number of transmitted and lost packets and the length of the output packet queue. The number of transmitted and lost packets can be used to obtain loss rate at the router while the length of the output packet queue can provide an estimate for the queueing delays which can be combined with propagation delays to obtain an estimate for the round trip time. One advantage of router MIBs is that they automatically aggregate statistics over time and do not require the transmission of numerous (probe) packets into the network. The overhead of MIB polling is primarily dictated by the frequency at which the polling occurs, and there is a trade-off between the associated message and processing overhead, and the accuracy with which changes in network parameters are being tracked. The MIB statistics are typically maintained at the interface level, so that the performance measures they track are for the aggregate traffic crossing that interface, and as with

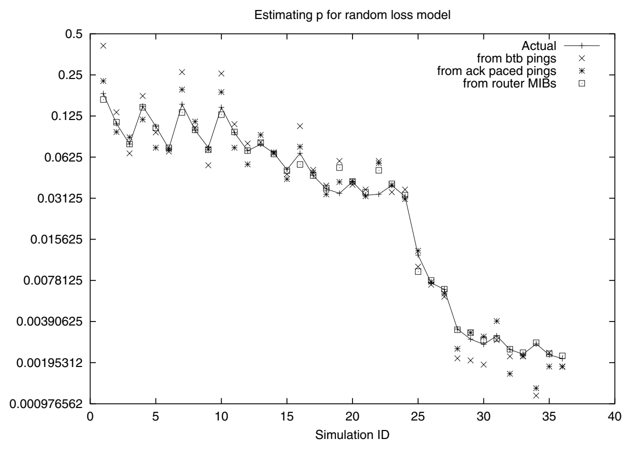
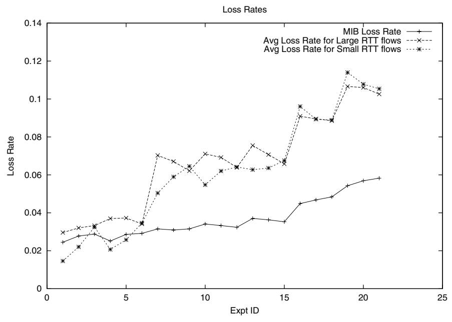


Fig. 1. Loss Rate Estimates for Two Bottleneck RED Router Simulations. MIB losses on individual routers were combined using 'Independent Loss' Assumption.

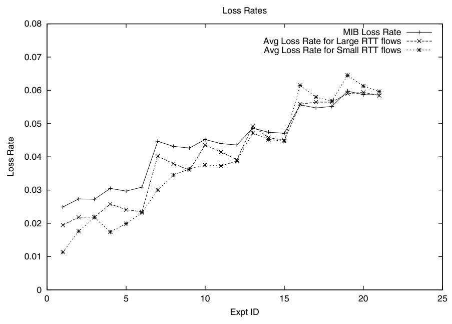
probes, those can differ from what individual TCP flows experience. In this respect, we found that the buffer management policy used at the interfaces can have a significant effect. For RED [15] buffers, the random nature of packet drop tends to minimize the difference between aggregate and individual loss rates. However, for droptail buffers the differences can be significant especially for the small *default* buffer sizes. In figure 2, we present the loss rates experienced by TCP flows with different RTTs and the MIB loss rate for RED and droptail buffers in our testbed experiments[2]. Clearly, with *default* sized droptail buffers, the MIB loss rate is not a good estimate for the loss rates suffered by TCP flows. Our simulation results indicate that for droptail the situation does not improve even with increased buffer sizes [14]. However, for RED buffers, the MIB loss rate can serve as a good estimate for TCP flow loss rates with accuracy improving as the buffer size increases. The improvement in estimate accuracy with increased buffer size can be explained as a result of increased randomness in packet drop.

Since SNMP MIBs provide information specific to a single router (interface), generating end-to-end statistics requires combining MIB information from all bottleneck routers on the path of a flow. Generally a TCP flow's path on the Internet is characterized by just one or two bottlenecks (on access links to high capacity backbone). Hence the problem of obtaining end-to-end loss rate is not that severe. Further, in general, the two bottleneck links on the path of a flow will be quite far away from each other and the traffic causing congestion in these links will be quite unrelated. Thus, it may be assumed that the losses at different bottleneck routers on the path of a flow are independent in nature and can be accordingly combined to obtain end-to-end loss rates. Figure 1 shows sample simulation results in this regard. It can be seen that 'independent loss' assumption provides quite satisfactory estimates of actual end-to-end loss rates.

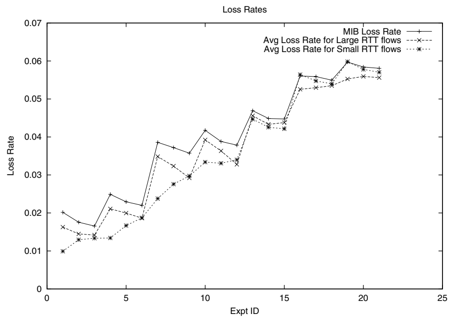
[2]Testbed details are provided in section IV-A.



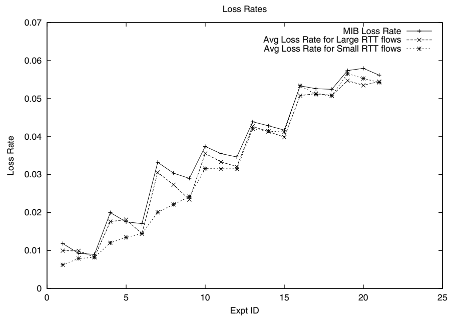
(a) Droptail Experiments (default buffer size: 40 packets)



(b) RED Experiments (buffer 100 packets)



(c) RED Experiments (buffer 500 packets)



(d) RED Experiments (buffer 1000 packets)

Fig. 2. Loss Rates of Flows with Different RTTs (for Testbed Experiments.)

III. TCP Throughput Prediction Models Based on Non-Invasively Obtained Parameters

In this section, we first briefly review why it is difficult to apply popular Amherst model to the setting where network parameters are assumed to be obtained using only non-invasive procedures. This is then followed by the development of several modifications and enhancements to the model, which allow for reasonably accurate predictions of TCP throughput based only on information obtained through non-invasive procedures.

A. Motivation for Developing New Models

We assume that the reader is familiar with the Amherst Model [1]. The model predicts the throughput of a TCP flow based on an expression that involves several parameters, including average RTT, first time-out duration ($T0$), and more important for our purpose, the probability p of "first" packet loss in an epoch[3]. Correctly estimating this parameter is, therefore, key to an accurate throughput prediction. In [1], this estimation was carried out based on the number of "loss events" (triple duplicate acks and time-outs) observed for the flow itself. The identification of loss events requires invasive flow level awareness. Nevertheless, our first attempt was to determine if it was possible to obtain a reasonable estimate for the first packet loss used by the Amherst model, by using only non-invasive procedures such as the ones described in Section II.

The approach we took was to develop a procedure for computing the first packet loss p needed by the Amherst model, from the measured overall loss rate L obtained using non-invasive procedures (Appendix A). Performing such a mapping required relating the observed number of losses to the number of lost packets (after the first loss) given by the loss model used in the Amherst model. Unfortunately, because the Amherst model assumes that, subsequent to the first packet loss, all the remaining packets in the window are also lost, the resulting *inversion* of the observed loss rate L into a first packet loss probability p, is highly inaccurate. As a result and as shown in Figure 3, the resulting throughput estimates are also inaccurate. Note that this assumption regarding packet losses is of very limited consequence[4] in the environment assumed by the Amherst model, i.e., when an accurate estimate is readily available for the first packet loss probability p. This is, however, not true when we need to derive an estimate for p based on the overall observed loss probability. In such a setting, the loss model used for relating these two quantities is of significant importance.

Our next step was, therefore, to determine how to modify the Amherst model, in order to use different loss models, i.e., models that rely on parameters that can be estimated using non-invasive procedures, such as the *random loss* model introduced in Section II.

In the random loss model, packets are lost randomly with a loss probability p. For such a loss model, the probability $P(i, W)$ that after the first packet loss in an epoch, a total of i packets, including the first loss, are lost in the following window of size W is given by:

[3]An epoch corresponds to a period of time during which the TCP flow is in congestion avoidance mode and regularly increasing its window.

[4]As illustrated in [1], the Amherst model gives reasonably accurate TCP throughput estimates.

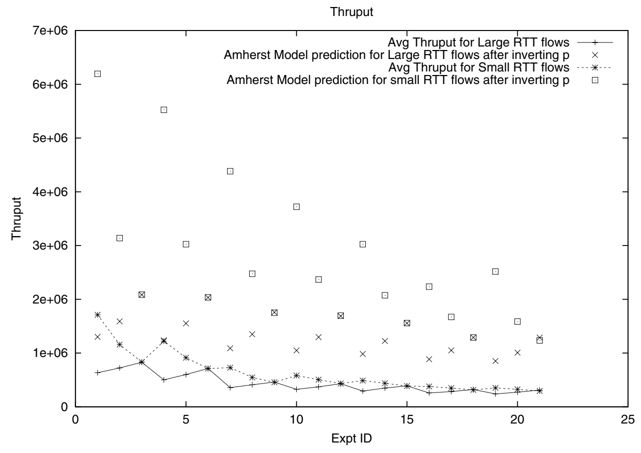


Fig. 3. Performance of the Amherst Model With First Loss Probability Estimated From Overall Loss Rate. (Results for RED experiments with 100 packet long buffers)

$$P(i, W) = \binom{W-1}{i-1} p^{i-1}(1-p)^{W-i}, \tag{2}$$

where the window size W is restricted to integer values.

Before proceeding with the derivation of the model, we briefly point to two important modifications we introduce to the approach used by the Amherst model. The first modification was in the computation of the probability of a retransmission time-out. In the Amherst Model, a retransmission time-out is avoided if three duplicate acks are received after the first lost packet, and the corresponding probability is computed as the probability of being able to successfully transmit 3 or more packets after the first packet loss. Because of the particular loss model used by the Amherst model (after the first packet loss, all remaining packets in the window are assumed lost), this computation is somewhat inaccurate. Our goal was, therefore, to develop a more precise model for computing the probability of retransmission time-out. One that would incorporate both the *size* W of the congestion window at the time of the first loss, and the likelihood of losing a certain number of packets after the first packet loss. This required a careful enumeration of the different possible loss scenarios for a given window size, and this is treated in details in the larger version of the paper [14]. The main result of this investigation is summarized in equation (3), which indicates that a loss of 3 or more packets in the window (typically) leads to a retransmission timeout.

$$P_{TO}(W) = \begin{cases} 1 & \text{if } W < 4 \\ 1 - P(1, W) & \text{if } W < 10 \\ 1 - P(1, W) - P(2, W) & \text{otherwise} \end{cases} \tag{3}$$

where $P_{TO}(W)$ is the probability that a timeout will take place when W is the window size at the time of packet loss. As before, $P(n, W)$ represents the probability of losing n packets out of a window of W packets, starting with the first lost packet.

The second main modification we introduce is in the determination of the initial congestion window at the beginning of an epoch. In the case of the Amherst Model, this initial window is taken to always be half of the window size at the end of the previous epoch, irrespective of the actual number of losses that trigger the end of the epoch. Such an assumption is not likely to be much of a concern for SACK [17] implementations of TCP. However, in TCP Reno implementations, which still comprise about 60% of currently deployed implementations [18], the congestion window can be reduced by more than a half depending on the number of losses. This is significant, as it can be shown that for the *same* total number of packets transmitted in an epoch, the duration of the epoch, i.e., the number of rounds, is larger when the initial window size is smaller. The implication of this result, is that always starting with the largest possible initial window size, as is done in the Amherst Model, can translate in over-estimates of the actual throughput, because it under-estimates the amount of time needed to transmit a given number of packets. Hence, it is desirable to develop a model that relates the initial window size to the final window size in the previous epoch and to the number of losses that ended the epoch.

For that purpose, we assume, as in the Amherst Model, that the steady state of a flow can be characterized by a sequence of similar epochs, with each epoch starting with a window of size W_i and ending with a window of size W_f. Let $Q(m, W_f)$ be the probability that W_i equals $\frac{W_f}{2^m}$. We assume that W_i never goes below $\frac{W_f}{8}$. After an analysis similar to the one performed for retransmission timeout probability (details in [14]), we can obtain the following expressions for $Q(m, W_f), m = 1, 2, 3$:

$$Q(1, W_f) = \begin{cases} 1 & \text{if } W_f < 4 \\ P(1, W_f) & \text{otherwise} \end{cases} \tag{4}$$

$$Q(2, W_f) = \begin{cases} 1 - Q(1, W_f) & \text{if } W_f < 10 \\ P(2, W_f) & \text{otherwise} \end{cases} \tag{5}$$

$$Q(3, W_f) = 1 - Q(1, W_f) - Q(2, W_f) \tag{6}$$

where as before, $P(n, W_f)$ is the probability of losing n packets out of a window of W_f packets starting with the first lost packet. Thus, the initial window size W_i, instead of always being set to $\frac{W_f}{2}$, is given by :

$$W_i = \sum_{n=1}^{3} Q(n, W_f) \frac{W_f}{2^n} \tag{7}$$

In the next section, we bring together the two modifications we have just outlined and the random loss model discussed earlier, to generate a new TCP throughput prediction model. As stated before, our goal is to develop a modified model that can operate on the basis of global network performance parameters that can be estimated using non-invasive procedures.

B. A Modified Model for Bulk TCP Throughput Prediction

In this section, we present a 'cyclical' model for bulk transfer TCP throughput prediction. By 'cyclical' we mean that the steady state of a Reno TCP flow can be characterized as a sequence of *epochs* during which the flow increases its congestion window linearly from an initial value W_i to a final value W_f with a slope of 1 packet per b rounds. Here a *round* corresponds to the time during which the TCP flow sends a congestion window worth of packets and b is the number of packets received before a TCP destination sends an ack back to the source. As in the Amherst Model, we assume that the duration of a round is independent of the congestion window size, and that an epoch consists of a congestion avoidance phase possibly followed by a timeout phase. For simplicity, we ignored packets sent during the slow start and fast retransmit phases, although the latter were taken into consideration when computing time-out probabilities [14].

The number of packets sent in the congestion avoidance phase of an epoch is determined by the packet loss probability p. Starting from the beginning of an epoch, let the αth packet be the first one to be lost. The returning acks of the preceding successfully transmitted packets in the window will allow the TCP flow to send $W_f - 1$ more packets before the packet loss is detected. Thus, the total number of packets sent in the congestion avoidance phase of the epoch is given by $Y = \alpha + W_f - 1$. Now, the probability that $\alpha = k$ is equal to the probability that $k-1$ packets were successfully transmitted before a loss occurs, which for the *random loss* model we consider is given by

$$P[\alpha = k] = (1-p)^{k-1}p, k = 1, 2, \ldots \tag{8}$$

Thus, the expected value of α is

$$E[\alpha] = \sum_{k=1}^{\infty}(1-p)^{k-1}pk = \frac{1}{p} \tag{9}$$

As a result, the expected number of packets sent in the congestion avoidance phase of an epoch is:

$$E[Y] = \frac{1-p}{p} + W_f \tag{10}$$

The number of packets sent in the congestion avoidance phase of an epoch can also be written in terms of W_i and W_f. After the first lost packet, the TCP flow sends $W_f - 1$ more packets before the packet loss is detected. Some of these packets are sent in the same round as the first lost packet, and the remaining β packets constitute what we term another *short round*. Therefore the total number of packets sent in the congestion avoidance phase of an epoch can also be written as:

$$Y = \frac{b}{2}(W_f + W_i)(W_f - W_i + 1) + \beta \tag{11}$$

The expected value of β can be derived as follows. If β packets are sent in the last short round, the position of the first lost packet in the penultimate round was $\beta + 1$. The probability that the first packet is lost at position $\beta + 1$ given that at least one packet was lost in the window of size W_f is equal to $\frac{p(1-p)^\beta}{1-(1-p)^{W_f}}$. Therefore, the expected value of β is given by:

$$E[\beta] = \sum_{i=0}^{W_f-1} i\frac{p(1-p)^i}{1-(1-p)^{W_f}} = \frac{1}{p} - \frac{1+(W_f-1)(1-p)^{W_f}}{1-(1-p)^{W_f}} \tag{12}$$

Thus, the expected number of packets sent in the congestion avoidance phase of an epoch can be expressed as:

$$E[Y] = \frac{b}{2}(W_f + W_i)(W_f - W_i + 1) + \frac{1}{p} - \frac{1+(W_f-1)(1-p)^{W_f}}{1-(1-p)^{W_f}} \tag{13}$$

Equations (10) and (13) provide two different ways for expressing the total number of packets sent in the congestion avoidance phase of an epoch. By equating these two expressions and simplifying we get:

$$W_f = \frac{b}{2}(W_f + W_i)(W_f - W_i + 1)(1-(1-p)^{W_f}) \tag{14}$$

Equation (14) together with equation (7) can be used to eliminate the unknown W_i and express W_f as a function of p. It can be shown that under the *random loss* model, for a given p, the resulting equation admits a unique solution for W_f [14]. Once W_f is known, W_i can be computed from equation (7), and the expected number of packets $E[Y]$ sent during the congestion avoidance phase can, therefore, be computed from equation (13). It now remains to compute the duration of an epoch in order to be able to predict the throughput of a TCP flow.

An epoch consists of a congestion phase possibly followed by a retransmission time-out phase. Let $X(W_i, W_f) = b(W_f - W_i + 1)$ be the number of rounds during which the congestion window increases from its initial value W_i to its final value W_f with a slope of 1 packet per b rounds. Thus, after including the final short round, the duration of the congestion avoidance phase of an epoch is given by $(X(W_i, W_f) + 1)RTT$ where RTT is the average duration of a round.

From equation (3), we know how to compute $P_{TO}(W_f)$, the probability that a congestion avoidance phase is followed by a retransmission timeout phase when the final congestion window is W_f. It, therefore, only remains to compute the number of packets that may be transmitted during this phase, as well as its duration. Using the same approach as suggested in [1], these quantities can be determined to be given by:

$$E[R] = \sum_{k=1}^{\infty} kP[R = k] = \frac{1}{1-p} \tag{15}$$

$$E[Z^{TO}] = T_0\frac{1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6}{1-p} \tag{16}$$

where $E[R]$ is the expected number of packets sent in the timeout phase and $E[Z^{T0}]$ is the expected duration of the timeout phase. Also $P[R = k]$ is the probability that k packets are sent in the timeout phase and is given by $p^{k-1}(1-p)$. T_0 is the average duration of the 'first' timeout period.

Based on the above equations, we are now in a position to compute the steady state throughput of a TCP flow (in packets per second):

$$B(p) = \frac{E[Y] + P_{TO}(W_f)E[R]}{(X(W_i, W_f) + 1)RTT + P_{TO}(W_f)E[Z^{TO}]} \tag{17}$$

In case W_f turns out to be more than W_{max}, the maximum permissible congestion window size, the formula above changes to:

$$B(p) = \frac{E[Y] + P_{TO}(W_{max})E[R]}{RTT(U + V + 1) + P_{TO}(W_{max})E[Z^{TO}]} \tag{18}$$

where U and V are expected number of rounds during which the congestion window increases from its initial value to W_{max} and then remains constant until the round where packet loss occurs. The derivation is simple and details can be seen in [14]. The expressions for U and V can be obtained to be:

$$U = b\sum_{m=1}^{3} Q(m, W_{max})(W_{max} - \frac{W_{max}}{2^m}) \tag{19}$$

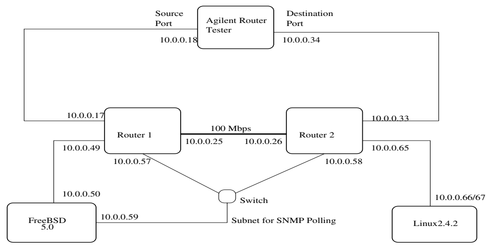


Fig. 4. Testbed Setup

$$
\begin{aligned}
V \quad = \quad & \frac{1}{W_{max}}\left(\frac{1-p}{p} - E[\beta]\right) + 1 \\
& -\frac{b}{2}\sum_{m=1}^{3} Q(m, W_{max})(\frac{W_{max}}{2^m} + W_{max} - 1)(1 - \frac{1}{2^m}) \qquad (20)
\end{aligned}
$$

Before concluding the section, we reiterate the differences between the Amherst Model and the throughput prediction model derived above:

1) The required input loss rate for the new model is simply the average loss rate effective during the lifetime of the TCP Reno flow.
2) The new model calculates the retransmission timeout probability and initial congestion window size in an epoch based on the final window size in the previous epoch and the number of losses that ended the epoch.

Now that we have completed the derivation of a TCP throughput prediction model that relies on network parameters derived using non-invasive estimation procedures, it remains to evaluate the accuracy achieved by this new model.

IV. Performance Evaluation

The throughput prediction model described in the previous section was evaluated thoroughly for a wide range of scenarios using both testbed experiments and NS2 [19] simulations. Testbed experiments allowed us make a good assessment of the non-invasive estimation methods and the performance of our throughput prediction model for the scenario where there is a single congested link in the path of TCP flows. We used NS2 simulations to perform the same assessment for the scenario with two congested links in the path. In the following we describe the testbed and simulation setups and then present the key performance results. A comprehensive presentation of the simulation and testbed results can be found in [14].

A. Testbed Experiments

Figure 4 shows the testbed setup. The setup consisted of two Cisco 3660 routers (running IOS 12.0) connected by a 100 Mbps link, an Agilent Router Tester box for generating background traffic and two PCs running FreeBSD 5.0 and Linux 2.4.2 acting as the source and destination for TCP flows. An interface on Linux machine was assigned two IP addresses (10.0.0.66 and 10.0.0.67) which acted as two destinations for the TCP flows starting from FreeBSD machine. The FreeBSD machine ran *Dummynet* [20] which allowed introduction of delays in the path of packets going to a specific destination. Thus the flows going to 10.0.0.66 and 10.0.0.67 addresses had propagation delays of R1 and R2 ms respectively. The R1/R2 values used in the experiments were 100/20, 80/40, 60/60 ms. The two PCs were kept time synchronized with an NTP server. In each experiment, 10 FTP flows were started from FreeBSD machine to the Linux machine for each destination address. The number of flows was chosen so that the PCs could easily handle the processing overhead of the experiments. Each flow transferred a 180MB long file from FreeBSD machine to different directories on Linux machine. The results shown here correspond to the duration (always greater than 15 minutes) when all the 20 flows were active. *tcpdump* ran on each machine to record all the TCP packets sent and received. The FreeBSD machine used Reno as the TCP protocol and 1460 bytes as the packet size. The Linux machine used delayed ack mechanism and had a *recv window* of 12 packets. While the experiments ran, the FreeBSD machine queried the MIB-2 Interfaces table on two routers after every 3 seconds over a subnet unaffected by congestion. We experimented with both droptail and RED buffer management policies. For buffer size x, the RED min and max thresholds were 0.5x and x with the max drop probability being 0.1 and exponential weighing constant for calculating average buffer occupancy being 1/512. Agilent router tester was used to generate 7 different background traffic mixes to get a good range of the loss rates. Each traffic mix had a bursty component consisting of bursts of 1000 packets every second of 60, 576 and 1460 byte packets each and a CBR component. The CBR component (with load values of 50, 60, 70, 80, 85, 90 and 95 Mbps) distinguished different traffic mixes and consisted of 50 parts each of 576 and 1460 byte packets and 5 parts of 60 byte packets (to represent TCP ACKs).

TABLE I
IDs for Testbed Experiments

Simulation ID	Background CBR Load (in Mbps)	Round-trip Propagation Delay R1/R2
1-3	50	100/20ms, 80/40ms, 60/60ms
4-6	60	
7-9	70	
10-12	80	
13-15	85	
16-18	90	
19-21	95	

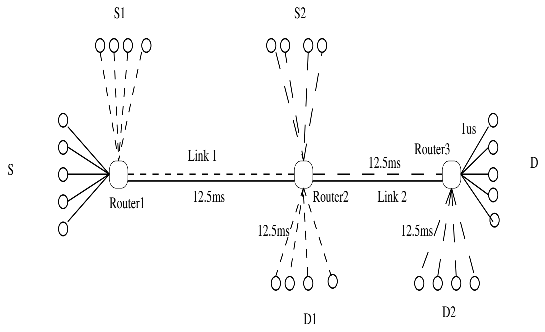


Fig. 5. Two Congested Routers Simulation Configuration

TABLE II

IDs FOR TWO CONGESTED ROUTERS SIMULATIONS

Simulation ID	Bottleneck bandwidth	Round-trip Propagation Delay R1/R2	Buffer at Congested Router
1-12 13-24 25-36	T1 T2 T3	50ms/250ms,100ms/200ms, 125ms/175ms,140ms/160ms	50,100,200 packets

TABLE III

COMMON SIMULATION PARAMETERS

TCP Flavor	Reno
Max CWND, Recv Window	64 packets
Delayed Acks	No
Timer Granularity	100ms
Simulation Time	256ms
Packet Size	500 bytes
Bottleneck Bandwidth	T1(1.544 Mbps),T2(6.132 Mbps),T3(44.736 Mbps)
Round Trip Propagation Delays (R1/R2)	50/250ms, 100/200ms, 125/175ms, 140/160ms
Buffer Size (x) at Congested Router	50,100,200 packets
Buffer Management Policy	RED (0.2x/0.8x/0.1,exp weighing const 0.002)
Access Link Bandwidth	Same as Bottleneck

B. *Simulation Configuration and Parameters*

Figure 5 shows the two congested routers simulation configuration used to evaluate the performance of estimation mechanisms and the throughput prediction model. The common simulation parameters are listed in Table III. In the simulations, a set *S-D* of 24 Reno TCP flows each with same RTT (*R1*) crossed the two congested router links, *link1* and *link2*. Both *link1* and *link2* were congested as a result of traffic generated by flow sets *S1-D1* and *S2-D2*, respectively. Each one of these sets consists of 24 Reno TCP flows with the same RTTs (*R2*). All simulations were run for a period of 256 seconds which was sufficient to allow all flows to achieve their steady state behavior. Further, to avoid synchronization among flows, the flows were started at random times within the first second of simulation run. The simulation results shown in the paper correspond to *S-D* flows.

C. *Results*

In order to facilitate the presentation of results for the different possible variations of parameters, testbed experiments and simulations were numbered according to their combination of parameters (Tables I and II). Based on this numbering, the parameters for a particular experiment/simulation can be ascertained from its ID. e.g. the *simulation* ID 30 in Table II refers to the simulation with bottleneck bandwidth T3, two RTT (R1/R2) values of 100/200ms and buffer size of 200 packets.

The key performance evaluation results are presented in figures 6 and 7. These figures show the average throughput achieved by TCP flows with same RTT and the predictions made by our throughput prediction model using actual loss rates and those obtained from SNMP MIBs. The results shown here correspond to RED simulations and testbed experiments. The loss rate range covered in these experiments can be seen in figures 1 and 2. Our throughput prediction model performed equally well for droptail experiments when actual loss rates were used as input [14]. However, since loss rate estimates obtained from SNMP MIBs were not good in droptail experiments, the performance of the throughput prediction model also suffered in consequence. The results clearly indicate that the new model works pretty well over a large range of loss rates and network conditions when reasonably accurate estimates for the loss rates are available. Note that the loss rate estimates obtained from SNMP MIBs in RED experiments result in quite accurate TCP throughput prediction. Further, the *independent losses* assumption used to combine loss rates at individual bottlenecks into end-to-end loss rate seems to give satisfactory performance.

V. CONCLUSION AND FUTURE WORK

This paper has been concerned with developing and evaluating models for predicting the steady state of representative TCP flows between fixed end-points, based on non-invasively obtained network data. The motivation for this work is to enable the transformation of data readily-available to service providers into user and application specific metrics. As existing TCP models do not readily fit the task at hand, we chose the popular Amherst model as a starting point and developed a modified cyclical model to suit our purposes. We also investigated the appropriateness and limitations of polling and probing as techniques for generating suitable inputs for our model.

Our model has been validated using simulations and as well as testbed experiments under a variety of scenarios. However, there are several limitations of this work, some of which can be remedied with further research.

1) Our work focuses on long-lived TCP Reno flows. Metrics based on short-lived flows, as well as other TCP flavors need to be investigated [9].
2) Our work assumes that one can determine the congestion points affecting flows between two end-user sites. This assumption is valid when network access points are the principal bottlenecks. However, more sophisticated techniques for determining congestion points need to be coupled with our techniques, in order to select network samples relevant to a user.
3) The biggest task for the future is to undertake an assessment of the relevance of simple metrics (DNS lookup times, TCP throughput predictions) or their combinations to application performance (Web downloads, FTP transfer times).

REFERENCES

[1] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Reno performance: A simple model and its empirical validation," *IEEE/ACM Trans. Netw.*, vol. 8, no. 2, pp. 133–145, April 2000.

[2] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," Request For Comments (Standards Track) RFC 2581, Internet Engineering Task Force, April 1999.

[3] T. V. Lakshman and U. Madhow, "The performance of TCP/IP for networks with high delay-bandwidth products," *IEEE/ACM Trans. Netw.*, vol. 5, no. 3, pp. 336–350, June 1997.

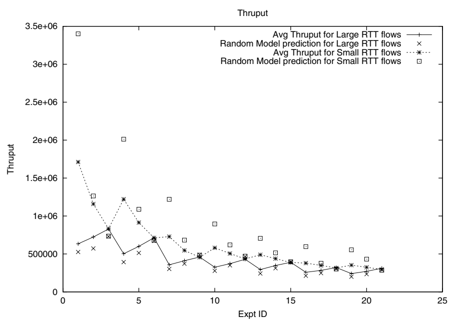
[4] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm," *Computer Commun. Rev.*, vol. 27, no. 3, July 1997.

[5] A. Kumar, "Comparative performance analysis of versions of TCP in a local network with a lossy link," *IEEE/ACM Trans. Netw.*, vol. 6, no. 4, pp. 485–498, August 1998.

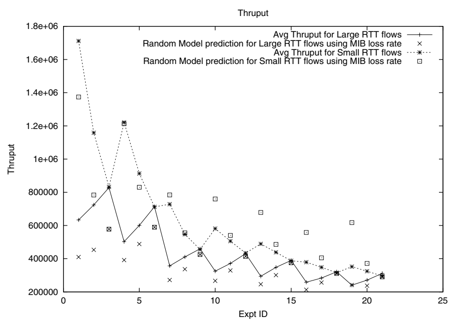
[6] E. Altman K. Avrachenkov and C. Barakat, "A stochastic model of TCP/IP with stationary random losses," in *Proc. SIGCOMM'2000*, 2000.

[7] F. Anjum and L. Tassiulas, "On the behavior of different TCP algorithms over a wireless channel with correlated packet losses," in *Proc. SIGMETRICS'1999*, 1999, pp. 155–165.

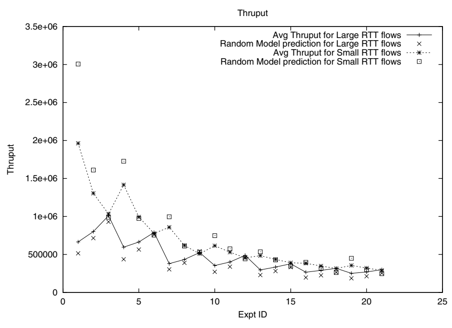
[8] A. Abouzeid S. Roy and M. Azizoglu, "Stochastic modeling of TCP over lossy links," in *Proc. INFOCOM'2000*, 2000, pp. 1724–1733.



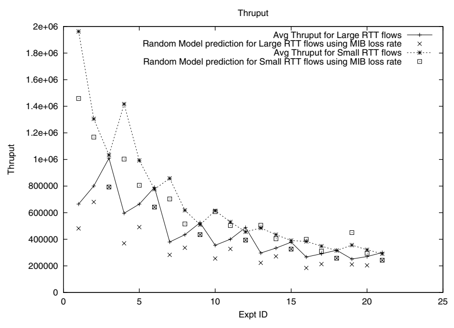
(a) RED (100 pkt buffer) Experiments



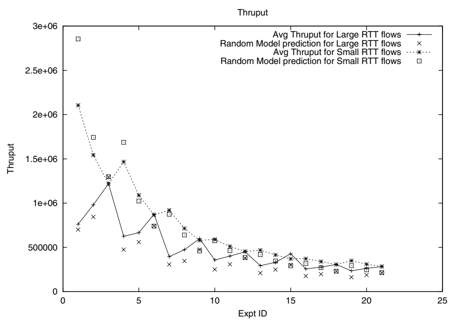
(a) RED (100 pkt buffer) Experiments



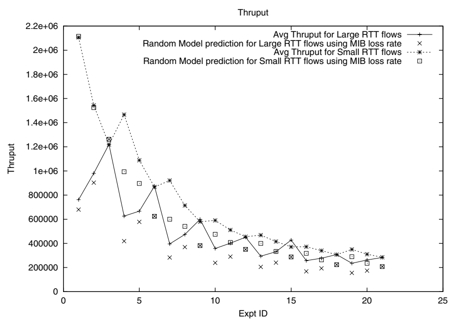
(b) RED (500 pkt buffer) Experiments



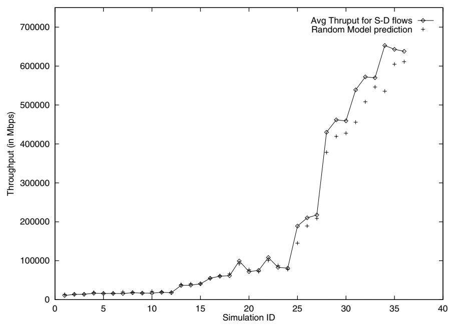
(b) RED (500 pkt buffer) Experiments



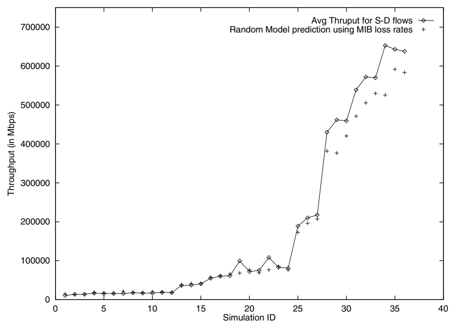
(c) RED (1000 pkt buffer) Experiments



(c) RED (1000 pkt buffer) Experiments



(d) Simulations with Two RED Routers



(d) Simulations with Two RED Routers

Fig. 6. Throughput Prediction with Actual Loss Rates.

Fig. 7. Throughput Prediction with MIB Loss Rates.

[9] N. Cardwell S. Savage and T. Anderson, “Modeling TCP latency,” in *Proc. INFOCOM’2000*, 2000, pp. 1742–1751.
[10] I. Yeom and A.L. Narasimha Reddy, “Modeling TCP behavior in a differentiated services network,” *IEEE/ACM Trans. Netw.*, vol. 9, no. 1, pp. 31–46, February 2001.
[11] V. Paxson and M. Allman, “Computing TCP’s retransmission timer,” Request For Comments (Standards Track) RFC 2988, Internet Engineering Task Force, November 2000.
[12] G. Wright and W. Stevens, *TCP/IP Illustrated, Volume 2: The Implementation*, Addison–Wesley, Reading, Massachusetts, 1996.
[13] V. Jacobson, “Berkeley TCP evolution from 4.3-Tahoe to 4.3-Reno,” in *Proceedings of the Eighteenth Internet Engineering Task Force*, Vancouver, B.C., 1990, University of British Columbia, p. 365.
[14] M. Goyal, R. Guerin, and R. Rajan, “Predicting TCP throughput from non-invasive data,” 2001, http://www.seas.upenn.edu/guerin/.
[15] S. Floyd and V. Jacobson, “Random early detection gateways for congestion avoidance,” *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, August 1993.
[16] K. McCloghrie and M. Rose, “Management information base for network management of TCP/IP-based internets: MIB-II,” Request For Comments (Informational) RFC 1213, Internet Engineering Task Force, March 1991.
[17] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, “TCP selective acknowledgement options,” Request For Comments (Informational) RFC 2018, Internet Engineering Task Force, October 1996.
[18] M. Allman, “A web server’s view of the transport layer,” *Computer Commun. Rev.*, vol. 30, no. 5, October 2000.
[19] “The network simulator - NS2,” http://www.isi.edu/nsnam/ns/.
[20] L. Rizzo, “Dummynet: a simple approach to the evaluation of network protocols,” *Computer Commun. Rev.*, vol. 27, no. 1, January 1997.

APPENDIX

I. ESTIMATING p FOR THE AMHERST MODEL FROM THE OVERALL LOSS RATE L

The goal of this section is to present a method for deriving the “first packet loss probability” p that the Amherst model [1] uses, when what is available from (non-invasive) measurements is the overall loss rate L. The approach we take, is to equate L to the ratio of the expected number of packets lost to the expected number of packets sent during an epoch in Amherst Model.

Let W be the final window size achieved in an epoch and $\widehat{Q}(W)$ be the probability of the epoch ending in a retransmission timeout. The expressions for W and $\widehat{Q}(W)$ are given by [1]:

$$W = \frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \left(\frac{2+b}{3b}\right)^2} \tag{21}$$

$$\widehat{Q}(w) = min\left(1, \frac{(1-(1-p)^3)(1+(1-p)^3(1-(1-p)^{w-3}))}{1-(1-p)^w}\right). \tag{22}$$

Then, the expected number of packets sent during an epoch in Amherst model is given by:

$$E[Y] = \frac{1-p}{p} + W + \widehat{Q}(W)\frac{1}{1-p} \tag{23}$$

The expected number of packets lost, N, in an epoch in the Amherst model is the sum of expected number of packets lost in the congestion avoidance phase plus those lost in the timeout phase. An expression for N can be obtained that involves p and uses the Amherst model assumption that after the first loss the remaining packets in the round are also lost. This gives:

$$N = \begin{array}{l} \left(\sum_{n=1}^{W}\left(W-n+1+p\sum_{i=0}^{n-1} i(1-p)^{n-1-i}\right)\frac{(1-p)^{n-1}p}{1-(1-p)^W}\right) \\ +\widehat{Q}(W)\left(\frac{1}{1-p}-1\right) \end{array} \tag{24}$$

The first expression in parenthesis corresponds to the average number of packets lost at the end of congestion avoidance phase, while the second expression corresponds to packets lost during timeout retransmissions. The above expression can be simplified to yield:

$$N = W - \frac{1-p}{p} + \frac{(1-p)(1+(1-p)^W)}{1-(1-p)^2} + \widehat{Q}(W)\left(\frac{1}{1-p}-1\right) \tag{25}$$

As a result, the measured overall loss rate L and the first packet loss probability p used in Amherst model are related through the following expression $L = \frac{N}{E[Y]}$. It can be shown that the rhs is a monotonically increasing function of p. Hence the value of p can be numerically computed for any given L.