This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2005 proceedings.

Throughput Modeling of TCP With Slow-start and Fast Recovery

Kaiyu Zhou, Kwan L. Yeung and Victor O.K. Li
Department of Electrical and Electronic Engineering
The University of Hong Kong
Pokfulam, Hong Kong, China

***Abstract*— Despite the rich literature on modeling TCP, we find two common deficiencies with the existing approaches. First, none of the work gives sufficient treatment to slow-start, although almost all of them show that retransmission timeout events are common. Second, the probability that retransmission timeout occurs has been underestimated, because retransmission timeout is coupled with fast recovery but fast recovery has not been properly modeled in the previous work. In this paper, new analytical models for predicting the steady state throughput of TCP flows are proposed. All major TCP mechanisms, including slow-start, congestion avoidance, fast retransmit, and fast recovery, are jointly considered under both bursty and independent loss models. We show that our proposed throughput models capture TCP performance more accurately.**

***Index Terms* — Bursty Loss, Independent Loss, TCP Performance, Slow-Start, Fast Recovery**

I. INTRODUCTION

Internet traffic is dominated by TCP [1], whose dynamics greatly influence the overall performance of the Internet. In order to have a better understanding of TCP, numerous analytical models have been proposed (e.g. [1-9]). Most of them focus on modeling the steady state TCP throughput, and some investigates the transfer delay for short transactions. Among them, probably the most widely accepted TCP steady state model is introduced in [2], and it is subsequently adopted by TFRC [10]. However, after studying TCP performance under the general stationary ergodic loss model, Altman *et al*. [8] point out that the model of TCP in [2] may not be accurate. They claim that the correctness of the model in [2] is due to error cancellation between the model for TCP and the model for packet loss.

Despite the rich literature on modeling TCP, we find two common deficiencies with the existing approaches. First, none of the work gives sufficient treatment to TCP *slow-start*, although almost all of them show that retransmission timeout events are common. (Note that a TCP flow must enter slow-start if a timeout occurs.) Second, the cause of retransmission timeout is not analyzed with the level of details it deserves, although almost all of the previous efforts notice that timeout greatly degrades TCP performance. (Note that a TCP flow sends little or no packet during the long duration of timeout.) *Fast retransmit* and *fast recovery* algorithms are the major causes of retransmission timeout, but only [3] has analyzed fast recovery under the simplified assumption of no retransmission loss. The work in [11] also pointed out that retransmission timeout for Reno under bursty loss is much more severe than what is shown in [2] with the analysis of fast recovery.

Motivated to find a more accurate model for TCP steady state throughput, we extend the work in [2] to analyze slow-start and fast recovery, and extend the analysis of fast recovery in [3] to include retransmission timeout caused by *loss of retransmitted packets*. Two of the most widely accepted packet loss models are adopted in our study, bursty loss [1-3, 6, 7] and independent loss [3, 8]. Since TCP Sack [12] is the current IETF recommendation of TCP implementation, Sack is assumed in our analysis. Simulations are used to validate our analytical models. We show that without considering slow-start and fast recovery, previous models cannot accurately capture TCP performance.

The rest of the paper is organized as follows. Section II summarizes the key features of TCP Sack. Section III introduces the assumptions, definitions, and notations to be used in the analysis. Section IV presents the TCP throughput model under bursty loss. Section V gives the throughput model under independent loss. Section VI validates our proposed models by simulations. Section VII concludes the paper.

II. TCP SACK

The *Selective Acknowledgment* (Sack) option [12] was introduced to improve TCP performance when multiple packets are lost from a window of data. With Sack, the data receiver can inform the sender about all packets that have arrived successfully, so the sender needs to retransmit only the packets that have actually been lost.

When the sender receives *triple duplicate ACKs* (TD), it responds with *fast retransmit*, i.e. it retransmits the first unacknowledged packet, sets the slow-start threshold (H) to $\max(cwnd/2, 2)$, then halves its *congestion window* ($cwnd$) and activates the *fast recovery* algorithm. Upon receiving an ACK that acknowledges all the outstanding data when fast recovery was triggered ($maxseq$), the sender exits fast recovery. In fast recovery, Sack uses a variable $pipe$ to estimate the number of packets outstanding in the network. The sender sends new or lost packets only if the value of $pipe$ is less than $cwnd$. When TD is received, $pipe$ is set to $cwnd-3$. With each packet transmission or retransmission, $pipe$ increases by one. When a duplicate ACK arrives and reports that new data has successfully arrived at the receiver, $pipe$ is decreased by one. To ensure Sack never recovers more slowly than slow-start, Sack decreases the $pipe$ by two with the receipt of a *partial ACK* (an ACK received in the fast recovery that advances the acknowledgment number field but does not take the sender out of fast recovery). Throughout this paper, we assume the Sack implementation in [12].

This work is supported in part by Hong Kong Research Grants Council Earmarked Grant HKU 7048/02E, and in part by the Areas of Excellence Scheme established under the University Grants Committee of the Hong Kong Special Administrative Region, China (Project No. AoE/E-01/99).

Retransmission timeout [13] is used as the last resort to recover lost packets. Every time a data packet is sent, if the retransmission timer is not running, a new timer is started and counted down with an initial value of t_0 seconds. t_0 is given by

$$t_0 = \min(60, \max(1, RTT + 4 \cdot RTTVAR)) . \quad (1)$$

where RTT is the round trip time and $RTTVAR$ is the variance of the round trip time. If another timeout occurs before receiving an ACK that acknowledges $maxseq$, the sender *backs off* its retransmission timer by setting it to $2t_0$. From (1), the doubling of retransmission timer is only effective when t_0 (before doubling) is smaller than 60 seconds.

When the sender's retransmission timer expires, the sender sets H to $\max(cwnd/2, 2)$ and restarts with *slow-start*, i.e. the sender resets $cwnd$ to 1 and increases it by one with each received ACK, until the slow-start threshold H is reached. From that time onwards, *congestion avoidance* takes over. In congestion avoidance, the sender increases its $cwnd$ linearly by $1/cwnd$ with each received ACK until the sender receives TD or the retransmission timer expires.

III. Assumptions, Definitions, and Notations

We focus only on the transport layer performance, the delays introduced by other layers, such as those caused by scheduling or buffering in the session or application layers and contention in the data-link layer, are not considered. We also do not consider the connection establishment time needed for a TCP flow. We assume that for the duration of the data transfer, the sender always sends full-sized packets as fast as its congestion window allows, and the receiver advertises a consistent flow control window. As the steady state throughput is modeled, we assume the duration of data transfer is long enough such that the flow experiences enough packet losses.

We model the performance of TCP in terms of "*rounds*." A round begins with the transmission of a window of packets and ends upon the receipt of one or more ACKs of these packets, which implies that the time needed to send out all the packets in a window is smaller than the duration of a round. Therefore the duration of a round is independent of the window size, and is determined mainly by the round trip propagation delay.

For the characterization of packet losses, two of the widely investigated loss models are adopted, *bursty loss* [1-3, 6, 7] and *independent loss* [3, 8]. The independent loss model assumes all the packet losses in the network are independent, with stationary *loss ratio* p_I. In the bursty loss model, the loss probability of a packet in a round is independent of any packet loss in other rounds. In the same round, the loss probability of a packet depends on whether the previous packet in the same round is lost. If the previous packet is lost, the current packet is also lost; otherwise the packet is lost with probability p. So if a packet is lost in a round, all subsequent packets in the same round are also lost. Similar to [1-3, 6, 7], we assume that the probability of packet loss is independent of the window size, and we do not consider ACK packet losses. For bursty loss, we treat all the correlated packet losses in the same round as a single *loss event*, and refer to p as the *loss event ratio*. For independent loss, since there is no correlation between packet losses, a loss event has exactly one lost packet, so the value of p_I is exactly the same as p. For simplicity of discussion, we use symbol

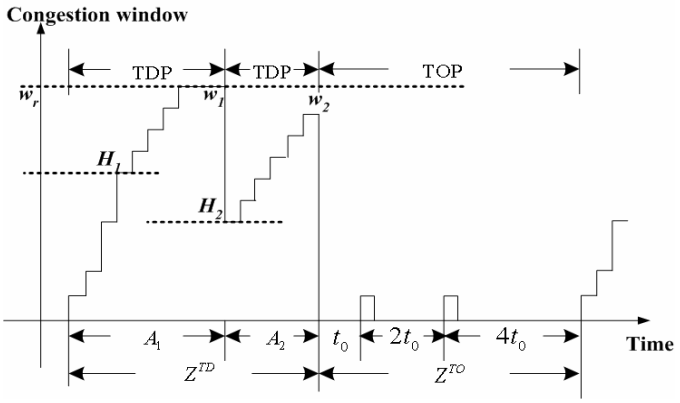

Fig. 1. Evolution of congestion window size

Table I. Notation list.

Z_i^{TO}	the duration of the i-th TOP
Z_i^{TD}	the duration between $(i-1)$-th and i-th TOP
H_{ij}	slow-start threshold when the j-th TDP of Z_i^{TD} begins
w_{ij}	$cwnd$ when the j-th loss event of Z_i^{TD} is detected
w_r	the receiver's advertised window size
R_i	the number of packets sent during Z_i^{TO}
n_i	the number of TDP's in interval Z_i^{TD}
Y_{ij}	packets transmitted in the j-th TDP of Z_i^{TD}
A_{ij}	the duration of the j-th TDP of Z_i^{TD}

p but not p_I when studying the performance under the independent loss model.

We plot the generalized evolution of a TCP sender's congestion window in Fig. 1. We define *timeout period* (TOP) as the duration from when the retransmission timer to be expired is set, to when the packet that successfully recovers the data transmission is sent. We define *triple duplicate period* (TDP) as the duration from when the sender begins to increase its transmission rate (i.e. enters slow-start or congestion avoidance) to the start of the successive TDP or TOP. The list of notations to be used is summarized in the table below.

We define $Q = 1/E[n]$, where $E[n]$ is the expectation of n_i. So Q, or *timeout probability*, is the probability that a timeout occurs given that a loss event happens. The steady state throughput T is thus given by

$$T = \frac{E[n]E[Y] + E[R]}{E[Z^{TD}] + E[Z^{TO}]} = \frac{E[Y] + Q \cdot E[R]}{E[A] + Q \cdot E[Z^{TO}]} . \quad (2)$$

Note $E[Z^{TO}]$ is generally far bigger than $E[A]$, but $E[R]$ is far smaller than $E[Y]$. In the following two sections, we will derive all the variables on the right hand side of (2) under both bursty loss model and independent loss model.

IV. Throughput Under Bursty Loss

A. A closer look on TDP

Without loss of generality, we consider the evolution of the congestion window in a particular TDP in Fig. 2. For simplicity, we drop the subscript i in all the notations defined earlier. To capture the difference in slow-start, congestion avoidance, and fast

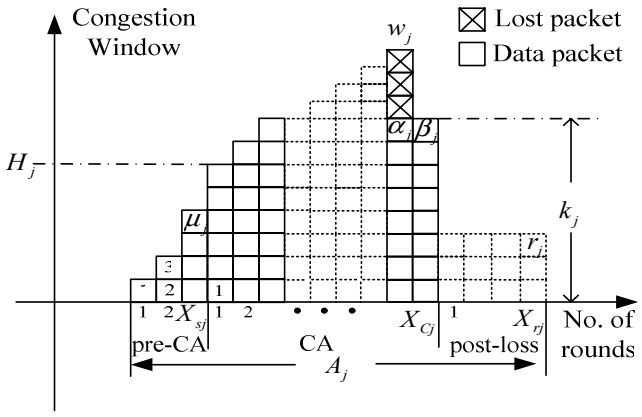

Fig.2. Congestion window and packets sent in a TDP

recovery, A_j in Fig. 2 is further divided into three periods, *Pre-CA*, *CA* and *post-loss*. The CA period is from when the sender enters congestion avoidance to one round after the loss event. The Pre-CA period is from the start of TDP to the start of CA. The post-loss period covers the rest of the TDP.

Respectively, we denote the number of packets sent in Pre-CA, CA and post-loss periods of the j-th TDP as μ_j, β_j, and r_j, and the duration of each period as X_{sj}, X_{cj}, and X_{rj}. From Fig. 2, Y_j and A_j are given by

$$Y_j = \mu_j + \beta_j + r_j, \tag{3}$$

$$A_j = X_{sj} + X_{cj} + X_{rj}. \tag{4}$$

Let packet $\alpha_j + 1$ be the first packet lost in a given loss event. We have

$$E[\alpha] = \frac{1}{p} - 1 = E[\mu] + E[\beta] - w, \tag{5}$$

where $w = E[w_j]$. The problem of TCP performance modeling is then transformed to the analysis of TCP reaction under a given packet loss in each of the timeout period, Pre-CA, CA, and post-loss.

B. Timeout Probability

We call the round where a loss event occurs as the "*loss round*" and all subsequent rounds as the "d-th *post-loss round*", where d is the round-distance from the loss round.

Let $y_j + 1$ be the sequence number of the first packet sent in the loss round, $y_j + k_j + 1$ the sequence number of the first lost packet in the loss round, and $y_j + w_j + m_j + 1$ the sequence number of the first lost packet in the 1st post-loss round. Under the bursty loss model, all packets following $y_j + k_j + 1$ in the loss round are also lost. However, since k_j packets are acknowledged, another k_j packets can be sent in the 1st post-loss round. Assume another loss event may occur among these k_j packets, say at $y_j + w_j + m_j + 1$. m_j is the number of duplicate ACKs received by the sender. If m_j is bigger than three, the sender receives a TD and $y_j + k_j + 1$ is retransmitted. The sender then changes to fast recovery. If m_j is smaller than two, a timeout occurs. We refer to this kind of timeouts as *No Triple Duplicate ACKs* (NTD). We call all the loss events followed by a retransmission timeout as *timeout losses*, and the loss event causing NTD timeout as *NTD timeout loss*.

Let $A(s,l)$ be the probability that the first l packets are acknowledged in a round of s packets sent, given there is a loss event in the round. Then

$$A(s,l) = \frac{(1-p)^l p}{1-(1-p)^s}.$$

Let $C(s,l)$ be the probability that s packets are sent but only the first l packets are acknowledged, and the rest of the packets in the round are lost. Then

$$C(s,l) = \begin{cases} (1-p)^l p, & l < s \\ (1-p)^s, & l = s \\ 0, & l > s \end{cases}.$$

The probability that the sender receives exactly m_j duplicate ACKs in the 2nd post-loss round $P(m_j)$ is given by

$$P(m_j) = \sum_{k_j = m_j}^{w_j - 1} A(w_j, k_j) C(k_j, m_j).$$

Let $P_{NTD}(w_j)$ be the probability that an NTD timeout occurs given that a loss event has happened. $P_{NTD}(w_j)$ is given by

$$P_{NTD}(w_j) = \min\left(1, \frac{(1-(1-p)^3)(1+(1-p)^3(1-(1-p)^{w_j-3}))}{1-(1-p)^{w_j}}\right). \tag{6}$$

Note that in fast recovery, there is no guarantee that the sender will send new or lost packets when the ACK for the first retransmitted packet arrives. The sender sends (new or lost) packets only if the value of *pipe* is less than *cwnd*. When the sender enters fast recovery, *pipe* is set to $w_j - 3$ and *cwnd* is set to $w_j / 2$. With each additional duplicate ACK received, *pipe* is decreased by one. After the sender receives the partial ACK corresponding to the retransmission of $y_j + k_j + 1$, *pipe* is decreased by two. So if $w_j - m_j - 2 \geq w_j / 2$, a timeout occurs. We call it a *Not Recoverable* (NR) timeout, and the corresponding loss event as *NR timeout loss*. Let $P_{NR}(w_j)$ be the probability that an NR timeout occurs given that a loss event has happened. $P_{NR}(w_j)$ is given by

$$P_{NR}(w_j) = P(w_j - m_j - 2 \geq {}^{w_j}\!/_2 \mid m_j \geq 3) = \sum_{m_j = 3}^{w_j/2 - 2} P(m_j). \tag{7}$$

Note that the sender retransmits a packet at most once before retransmission timeout. When $w_j - m_j - 2 < w_j / 2$, suppose one of the retransmitted packets is lost, a timeout occurs. We call it *Retransmission Loss* (RL) timeout, and call the associated loss event *RL timeout loss*. The probability of having an RL timeout given that a loss event happened is $P_{RL}(w_j)$, where

$$P_{RL}(w_j) = \sum_{m_j = \max(3, \frac{w_j - 4}{2})}^{w_j - 1} P(m_j) \sum_{i=0}^{w_j - m_j - 1} C(m_j, i). \tag{8}$$

When an RL timeout loss occurs, the TCP flow sends out packets continuously before the retransmission timer expires. In fact, from

the 4th post-loss round till the retransmission timer expires, the sender receives only duplicate ACKs (instead of partial ACKs). Assuming no more packet loss in this duration, the number of duplicated ACKs received in each round is constant. This is equivalent to having an effective congestion window w_{RL} that controls the transmission rate of the TCP flow in this duration. The value of w_{RL} is two times the number of partial ACKs received in the 3rd post-loss round, and is given by

$$E[w_{RL}] = \frac{2}{P_{RL}(w_j)} \sum_{m_j=\max(3,\frac{w_j-4}{2})}^{w_j-1} P(m_j) \sum_{i=0}^{w_j-m_j-1} i \cdot C(m_j,i) . \quad (9)$$

To account for all the timeout possibilities of Sack, the overall timeout probability Q for Sack connections is given by

$$Q = P_{NTD}(w_j) + P_{NR}(w_j) + P_{RL}(w_j) . \quad (10)$$

From (10), we can see that the timeout probability is a function of both p and w_j.

C. Throughput model

We next derive the throughput model by calculating the expectation for other variables in (2), (3), and (4). Our definition of post-loss period begins with the 2nd post-loss round. From Figs. 3 to 5, the expectations of r_j, the number of packets sent in a post-loss part, and its duration X_{rj}, are given by

$$E[r] = P_{RL} + P_{NR} , \quad (11)$$

$$E[X_r] = P_{NR} - P_{NTD} . \quad (12)$$

From [2], $E[Z^{TO}]$ the expectation of the duration of a timeout period is

$$E[Z^{TO}] = t_0 \frac{1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6}{1-p} . \quad (13)$$

As shown in Part B of this section, if RL timeout occurs, TCP flows send out packets continuously before the retransmission timer expires. With $E[w_{RL}]$ given by (9), we change the representation of $E[R]$, the expectation of the number of packets sent in a timeout period, obtained in [2] to

$$E[R] = \frac{P_{RL}}{Q} \frac{t_0}{RTT} E[w_{RL}] + \frac{1}{1-p} . \quad (14)$$

When NR or RL timeout occurs, the TCP flow experiences two congestion slowdowns. The sender sets the slow-start threshold H to $\max(cwnd/2, 2)$ when the retransmission timer expires, where $cwnd$ has already been halved with fast retransmit. Thus, the expectation of H is

$$E[H] = \max((2 - P_{NR} - P_{RL})w/4, 2) .$$

In the above equation, for simplicity, we have ignored the losses in TOP, and we believe that the error introduced in H should be negligible. Our analysis indicates that w increases inversely with p. When p is small, timeout losses seldom occur. When p is large, timeout losses become common, but H tends to be bounded by 2.

In the Pre-CA period, the expectation of μ_j, the number of packets sent, and its duration X_{sj} are

$$E[\mu] = 2^{\lfloor \log_2[H] \rfloor + 1} , \quad (15)$$

$$E[X_s] = \log_2(H) = \log_2(\max((2 - P_{NR} - P_{RL})w/4, 2)) . \quad (16)$$

In the CA period, the sender's $cwnd$ starts with $E[H]$. Thus we have

$$E[X_c] = w - E[H] + 1 , \quad (18)$$

$$E[\beta] = \sum_{i=0}^{E[X_c]-1} (H + i) + \frac{w}{2} . \quad (17)$$

With (16), μ is shown with w. Similarly β is shown with w in (17), so w can be calculated by solving equation (5). Note that the variables in (3) are functions of p and w, such as μ in (16), β in (17), and r in (11), so $E[Y]$ is determined with a given p. Similarly, with X_s in (16), X_c in (18), X_r and (12), $E[A]$ is also determined with a given p. Hence, with Q, R, and Z^{TO} given by (10), (13), and (14) respectively, Sack throughput of equation (2) under bursty loss can be obtained.

V. Throughput Under Independent Loss

A. Timeout probability

With independent loss, let $B'(s,l)$ be the probability that exactly l packets are acknowledged with s packets sent. Then

$$B'(s,l) = \binom{s}{l} (1-p)^l p^{s-l} .$$

We first investigate how NTD timeout happens with independent loss. Since all packets before the first lost packet are acknowledged, $w_j - 1$ packets are sent following the first lost packet. Among these $w_j - 1$ packets, we define the number of packets that are successfully acknowledged by m'_j. It is easy to see that if $m'_j < 3$, NTD timeout loss occurs. So we have

$$P_{NTD}(w_j) = \sum_{m'_j=0}^{2} B'(w_j - 1, m'_j) . \quad (19)$$

If NR timeout loss occurs, we have $w_j/2 - 2 \ge m'_j \ge 3$. So $P_{NR}(w_j)$ is given by

$$P_{NR}(w_j) = \sum_{m'_j=3}^{w_j/2-2} B'(w_j - 1, m'_j) . \quad (20)$$

The number of packets to be retransmitted is $w_j - m'_j$. Any loss among the $w_j - m'_j$ retransmitted packets causes RL timeout. So $P_{RL}(w_j)$ is given by

$$P_{RL}(w_j) = \sum_{m'_j=\max(3,\frac{w_j-4}{2})}^{w_j-1} B'(w_j - 1, m'_j)(1 - (1-p)^{(w_j - m'_j)}) . \quad (21)$$

Substituting (19) to (21) into (10), timeout probability Q can be found.

B. Throughput model

Compared with the bursty loss model, when RL timeout occurs, the sender receives more duplicate ACKs because not all the packets following the loss in the same round are lost. Accordingly, the sender sends more packets. We approximate this by rewriting the expression of $E[w_{RL}]$ in equation (9) as

$$E[w_{RL}] = \frac{2}{P_{RL}} \sum_{m'_j=\max(3,\frac{w_j-4}{2})}^{w_j-1} B'(w_j - 1, m'_j)(1 - (1-p)^{(w_j - m'_j)}) \frac{w_j - m'_j}{2} . \quad (22)$$

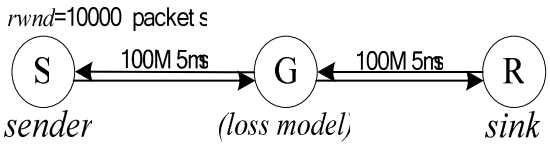

Fig. 3 The simulated network

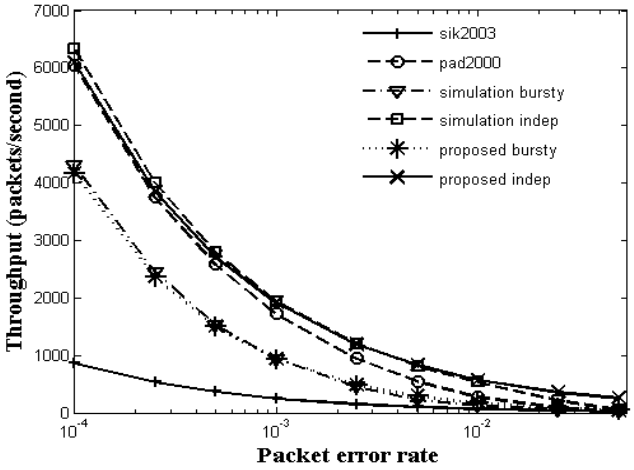

Fig. 4. Comparison of throughput estimations

For other cases studied in Part C of Section VI, Sack flows behave similarly as that under bursty packet loss model. After replacing (6) with (19), (7) with (20), (8) with (21), and (9) with (22), the Sack throughput under independent loss is given similar to that under bursty loss model as shown in Section VI.

VI. Model Validation and Comparison

We validate the proposed analytical models with the ns-2 simulator. The simulated network is shown in Fig. 3. Packets of 1KB are used. For simulations with independent loss, the Error Model implemented in ns-2 is used. Since bursty loss is not implemented in ns-2, we derive a Bursty Loss Model from the basic Error Model class. This Bursty Loss Model drops packets with probability p, the exact loss event ratio. After a packet is selected for dropping, all the subsequent packets in the same rounds are also dropped. This follows the assumption of the bursty loss model in Section II. In the simulations, the (Bursty) Error Model is placed before the queue of the link from nodes G to R.

Fig. 4 compares the simulation results with the analytical results from [2] (denoted "pad2000"), from [3] (denoted "sik2003"), and from our model (denoted "proposed bursty" and "proposed indep"). Each of the points in Fig. 4 is an average of 2000 simulation runs, with each simulation run lasting for at least 200 timeouts. Each of our proposed models is clearly a better match to the simulation results. Although the model of [2] is for TCP performance under bursty loss, it is in fact closer to the performance under independent loss. We agree with Altman's [8] opinion that although the model in [2] may be close to the real throughput in some cases, its accuracy is due to error cancellation between the model for TCP and the model for packet loss.

The accuracy of our proposed model is mainly due to two improvements over previous efforts, the analysis of fast recovery which results in a more accurate representation of timeout probability, and the analysis of slow-start which provides a more reasonable TCP window evolution model.

VII. Conclusion

Despite the rich literature on modeling TCP, we find two common deficiencies with the existing approaches. First, none of the work gives sufficient treatment to slow-start, although almost all of them show that retransmission timeout events are common. Second, the probability that retransmission timeout occurs may have been underestimated, because retransmission timeout is coupled with fast retransmit and fast recovery but fast recovery has not been properly modeled in the previous efforts. In this paper, new analytical models for predicting the steady state throughput of TCP flows are proposed. All major TCP mechanisms, including slow-start, congestion avoidance, fast retransmit, and fast recovery, are jointly considered under both bursty and independent losses. We show that our proposed models can capture the TCP performance more accurately.

References

[1]. N. Cardwell, S. Savage, and T. Anderson, "Modeling TCP latency," in Proceeding of Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, (INFOCOM '00), vol. 3, pp. 1742-1751, Mar. 26-31, 2000, Tel Aviv, Israel, 2000.

[2]. J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation," IEEE/ACM Transactions on Networking, vol. 8, pp. 133-145, 2000.

[3]. B. Sikdar, S. Kalyanaraman, and K. S. Vastola, "Analytic Models for the Latency and Steady-State Throughput of TCP Tahoe, Reno, and SACK," IEEE/ACM Transactions on Networking, vol. 11, pp. 959-971, 2003.

[4]. I. Khalifa and L. Trajkovic, "An overview and comparison of analytical TCP models," in Proceeding of IEEE. the 2004 International Symposium on Circuits and Systems, 2004. ISCAS '04, vol. 5, pp. 469-472 Vol.5, 2004.

[5]. S. Fortin and B. Sericola, "A model of TCP in wide area networks," in Proceeding of 10th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems, 2002, (MASCOTS'02), pp. 453-462, Oct. 12-16, 2002, Fort Worth, Texas, USA, 2002.

[6]. D. Zheng, G.Y. Lazarou, and R. Hu, "A stochastic model for short-lived TCP flows," in Proceeding of IEEE International Conference on Communications, 2003, (ICC'03), vol. 1, pp. 76-81, May 11-15, 2003, Anchorage, AK, USA, 2003.

[7]. T. J. Ott, T. V. Lakshman, and L. H. Wong, "SRED: stabilized RED," in Proceeding of Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '99), vol. 3, pp. 1346-1355, Mar. 21-25, 1999, New York, NY, USA, 1999.

[8]. E. Altman, K. Avrachenkov, and C. Barakat, "A stochastic model of TCP/IP with stationary random losses," ACM Computer Communication Review, vol. 30, pp. 231-242, 2000.

[9]. B. Sikdar, S. Kalyanaraman, and K. S. Vastola, "An integrated model for the latency and steady-state throughput of TCP connections," Performance Evaluation, vol. 46, pp. 139-154, 2001.

[10]. H. Handley, S. Floyd, J. Padhye, and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," in RFC3448, 2003.

[11]. K. Zhou, K. L. Yeung, and V. O.K. Li, "On Bursty Packet Loss Model for TCP Performance Analysis," in Proceeding of 2005 IEEE Workshop on High Performance Switching and Routing (HPSR'05), May 12-14, 2005, Hong Kong, China, 2005.

[12]. S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky, "An Extension to the selective acknowledgement (SACK) option for TCP," in RFC 2883, 2000.

[13]. V. Paxson and M. Allman, "Computing TCP's Retransmission Timer," in RFC2988, 2000.