A Simple Model for TCP Loss Recovery Performance over Wireless Networks

Beomjoon Kim and Jaiyong Lee

Abstract: **There have been a lot of approaches to evaluate and predict transmission control protocol (TCP) performance in a numerical way. Especially, under the recent advance in wireless transmission technology, the issue of TCP performance over wireless links has come to surface. It is because TCP responds to all packet losses by invoking congestion control and avoidance algorithms, resulting in degraded end-to-end performance in wireless and lossy systems. By several previous works, although it has been already proved that overall TCP performance is largely dependent on its loss recovery performance, there have been few works to try to analyze TCP loss recovery performance with thoroughness. In this paper, therefore, we focus on analyzing TCP's loss recovery performance and have developed a simple model that facilitates to capture the TCP sender's behaviors during loss recovery period. Based on the developed model, we can derive the conditions that packet losses may be recovered without retransmission timeout (RTO). Especially, we have found that TCP Reno can retransmit three packet losses by fast retransmits in a specific situation. In addition, we have proved that successive three packet losses and more than four packet losses in a window always invoke RTO easily, which is not considered or approximated in the previous works. Through probabilistic works with the conditions derived, the loss recovery performance of TCP Reno can be quantified in terms of the number of packet losses in a window.**

Index Terms: **Congestion control, fast retransmit probability, loss recovery, model validation and analysis, non-congestion packet loss, transmission control protocol (TCP).**

I. INTRODUCTION

Since the specification of transmission control protocol (TCP) was released [1], implementations of TCP have been enhanced with several mechanisms, such as congestion control [2], [3]. For reliable transmission, TCP congestion control provides a function to detect and recover packet losses using two basic algorithms such as fast retransmit and fast recovery, which is called TCP loss recovery in simple. TCP loss recovery performance is very important because it affects overall TCP performance such as end-to-end throughput. It is also the reason for continuous enhancements to the fast recovery algorithm resulting in subsequent TCP implementations such as TCP Tahoe [4], Reno [5], NewReno [6], [7], and selective acknowledgement (SACK) option [8], [9].

Recently, with the advance in wireless technology, TCP performance is gaining more importance. Basically, TCP is tuned to perform well in traditional networks where packet losses occur mostly because of congestion [4]. However, networks with wireless and other lossy links also suffer from significant losses due to bit errors and handovers. TCP responds to all packet losses by invoking congestion control and avoidance algorithms, resulting in degraded end-to-end performance in wireless and lossy systems [10].

There have been several efforts [11]–[19] to analyze TCP performance for non-congestion packet losses. However, although it has already been proven that the performance of TCP is mainly affected by the loss recovery performance [11], the detailed loss recovery behaviors of TCP are not considered or approximated in these works. In [13], although the authors show the effect of packet losses due to buffer overflow at a bottleneck node in terms of bandwidth-delay products (BDP), they do not consider the loss recovery features such as fast retransmit and fast recovery. In [14] and [15], the authors provide a complete analytical description of various TCP versions when packet is lost randomly and in series, respectively. In analytic modeling processes of these works, they assume that over three packet losses in a window always invoke RTO and the congestion window (*cwnd*) [2], [3] is always decreased by half regardless of the number of packets retransmitted. Actually, three packet losses can be recovered by retransmissions in a specific case as will be shown and *cwnd* with which the sender starts in congestion avoidance after successful fast recovery depends on the number of packet losses recovered. In [17], the authors provide a simple closed-form analysis of TCP throughput in terms of packet loss probability. Only the window evolution in congestion avoidance is considered and it is assumed that all packets transmitted after the first lost packet in a window are lost. The assumption may be true for a router using a drop-tail queue, but cannot be applied to wireless environments.

Consequently, the previous works addressed above have a focus on approximating overall TCP performance such as throughput rather than accurate modeling of loss recovery behaviors of TCP, which provides the departure point of our work. It may be because the loss recovery behaviors of TCP are comprised of somewhat complex procedures and produce a lot of states in connection with various factors such as the number of packet losses, window size, the position of packet losses, and so on. Therefore, in this paper, we have developed a simple model which facilitates to model loss recovery behaviors of TCP. We consider TCP Reno implementation, the most prevalent version of TCP, operating in wireless environments where packets are lost at random (i.e., i.i.d.) and correlated (i.e., bursty). Based on

Manuscript received May 16, 2002; approved for publication by Dong In Kim, Division II Division Editor, June 5, 2004.

B. Kim is with the Standardiztion & System Research Group, Mobile Communication Technology Research Lab., LG Electronics Inc., LG R&D Complex, 533, Hogye-1Dong, Dongan-Gu, Anyang-City, Kyongki-Do, 431-749, Korea, email: beom@lge.com.

J. Lee is with the School of Electrical and Electronic Engineering, Yonsei University, 134, Shinchon-Dong, Seodaemun-Gu, Seoul, 120-749, Korea, email: jyl@nasla.yonsei.ac.kr.

1229-2370/04/$10.00

the developed model, we can derive the accurate conditions for successful loss recovery of TCP Reno in terms of the number of packet losses in a window and the window size. Under the condition derived, the loss recovery performance of TCP can be quantified by the fast retransmit probability. The difficulty to obtain the complete distribution of the congestion window makes us deploy the fact that the evolution of TCP congestion window is of a cyclical structure. It can be analyzed with a Markov chain as in the related references [14], [15], [17]. Thus, we can compute the stationary distribution of the window process numerically.

The remainder of the paper is organized as follows. In Section II, we briefly describe the data transmission and loss recovery of TCP Reno. In Sections III and IV, we describe our model and derive the conditions for fast retransmit. We present the derived conditions probabilistically in Section V. Section VI contains the numerical results and their discussion. Finally, some conclusions are summarized in Section VII.

II. DESCRIPTION OF TCP RENO

TCP Reno implementation has modified the loss recovery to include fast recovery [2], [3], [5]. The fast recovery algorithm enables the TCP connection to maintain so-called *self-clocked* state after fast retransmit, thereby, the sender can continue to transmit packets in congestion avoidance if all packet losses in a window may be recovered by fast retransmits. In this section, we illustrate the operation of TCP Reno from the aspect of the sender, receiver, and loss recovery operation. The readers can refer to [11], [12], and [13] for details of the TCP Reno.

A. The Sender

After a connection is established, a sender initializes *cwnd* to one packet size and starts to transmit packets in slow start. For simplicity, all packets are assumed to have the same size. In slow start, every normal (i.e., non-duplicate) acknowledgement (ACK) makes *cwnd* to increase by one. When *cwnd* is equal to slow-start threshold (*ssthresh*) [2], [3], congestion avoidance begins. The initial value of *ssthresh* is determined at the connection setup phase. In congestion avoidance, *cwnd* increases by one divided by *cwnd*. In order to explain the evolution of *cwnd*, we define two parameters, the current *cwnd* by W and *ssthresh* by W_{th}. We consider the source has infinite packets to send so that the congestion window is always fully incremented. Thus, we have

$$W = \begin{cases} W+1, & \text{if } W < W_{th} \\ W + 1/\lfloor W \rfloor, & \text{otherwise.} \end{cases} \quad (1)$$

Every time a normal ACK is received, the congestion window slides not to include the packets that have been already transmitted. In this process, the sender is allowed to transmit the packets that are newly included in the congestion window.

B. The Receiver

The receiver delivers an ACK when a good packet is received. We do not consider the effect of 'delayed acknowledgement' [20] so that the number of packets received is always equal to the number of ACKs generated. As the size of an ACK packet is considerably small compared to a data packet size, an ACK packet is assumed not to be lost. Since the receiver has a finite buffer, it advertised a maximum window size $W_{\max}$ at the connection setup time. The size of the congestion window cannot be larger than $W_{\max}$.[1]

Actual TCP uses a sequence number expressed with the unit of bytes to discriminate the boundary between the packets. However, in this paper, for an easier description, each packet is assumed to have an assigned number (i.e., packet 1, packet 2) instead of the sequence number. An ACK from the receiver always delivers the *next expected* packet number. An ACK for packet n acknowledges all data packets up to and including the packet $(n-1)$, which is known as the cumulative strategy of TCP ACK. We use the term "An ACK for packet B by packet A" to mean that the sender transmits packet A, and the receiver delivers an ACK whose next expected packet number is B according to its reception of the packet A.[2]

If a packet is lost, the receiver delivers ACKs with the same next expected packet number every time it receives a good packet the sender transmits after the lost packet. These are called duplicate ACKs. Suppose that the sender transmits packets 1–8 and packet 5 is lost. The receiver returns the first ACK for packet 5 on receiving packet 4. When packets 6–8 are received, the receiver delivers three more ACKs for packet 5, since it has not received packet 5 yet; in this case, the number of duplicate ACKs is three.

C. The Loss Recovery of TCP Reno

There are two different ways for TCP Reno to recover a lost packet; one is by RTO and the other one is by fast retransmit and fast recovery. To trigger a fast retransmit, the sender should receive at least K duplicate ACKs for a lost packet (K is typically 3). Suppose that a packet is lost and the sender receives the Kth duplicate ACK for the lost packet at $t = t_0$. The lost packet is retransmitted instantly by fast retransmit without waiting RTO. After fast retransmit of the packet, *cwnd* and *ssthresh* are set as follows:

$$W(t_0^+) = \left\lfloor \frac{W(t_0)}{2} \right\rfloor + K \text{ and } W_{th}(t_0^+) = \left\lfloor \frac{W(t_0)}{2} \right\rfloor. \quad (2)$$

The addition of K reflects the fact that K more packets have successfully left the network.

During fast recovery that starts right after the fast retransmission, the sender increases the window by one every time another duplicate ACK arrives. This is also based on the fact that a duplicate ACK means that at least a packet is not lost but transmitted successfully. If the increased window includes a new packet, the sender is allowed to transmit it. If the fast retransmitted packet is not lost, the sender receives a normal ACK which makes the sender exit fast recovery. The sender continues to transmit a packet in congestion avoidance with the congestion window that is equal to W_{th} determined by (2).

[1] According to buffer size in a receiver, the value of $W_{\max}$ may vary during a TCP connection. Also, we assume that $W_{\max}$ keeps a constant value during a connection.

[2] The relation between A and B is $B = A + 1$.

When multiple packets are lost in a window, several fast retransmits and fast recoveries may be repeated. Suppose that two packets are lost in a window. When the Kth duplicate ACK for the first lost packet is received, the sender fast retransmits it. If it is successful, the sender receives the first ACK for the second lost packet[3] and enters congestion avoidance. At this time, *cwnd* is half as much as that before the first fast retransmit. If K more duplicate ACKs for the second lost packet are received, if possible, it is retransmitted by the second fast retransmit and the second fast recovery follows. At this time, *cwnd* is halved again. After receiving a normal ACK by the second retransmitted packet, the sender enters congestion avoidance with *cwnd* whose size is quarter of that before the first fast retransmit. Suppose n lost packets in a window can be recovered by n fast retransmits. If t_1 is the time just before the first fast retransmit and nth lost packet is fast retransmitted at t_2, the relation between $W(t_1)$ and $W(t_2)$ is given by

$$W(t_2) = \left\lfloor \frac{W(t_1)}{2^n} \right\rfloor. \tag{3}$$

By (3), the main reason can be explained that the performance of TCP Reno is poor when multiple packets are lost.[4] Even if all packet losses may be recovered by fast retransmits, *cwnd* has already decreased considerably and it takes long time to restore the size before packet losses. Note that *cwnd* increases at most by one every round-trip time (RTT) in congestion avoidance. Thus, multiple packet losses prevent TCP Reno from utilizing the advantage of fast recovery that it may not slow start after RTO if fast retransmit succeeds. It is the reason that the fast recovery algorithm of TCP Reno is called 'conservative'.

III. MODELING TCP RENO

A. Cyclic Window Evolution

We model the congestion window evolution of TCP Reno in terms of cycles. A cycle means a transmission period of good packets and consists of several rounds defined in [17]. When *cwnd* is equal to W packet, after the sender has transmitted all packets within *cwnd*, no other packet can be transmitted until ACKs by W packets arrive. If RTT is always longer than the transmission time of the packets within a window, no ACK will be received till the sender completes the transmission of W packets. In such a case, *cwnd* maintains a constant value for a while. This time duration is called a round.[5]

In a cycle, *cwnd* increases per every RTT till a packet loss occurs. After detecting a packet loss, the current cycle ends and is followed by loss recovery phase. Whether the next cycle starts in slow start or congestion avoidance is dependent on the result of loss recovery (see Fig. 1).

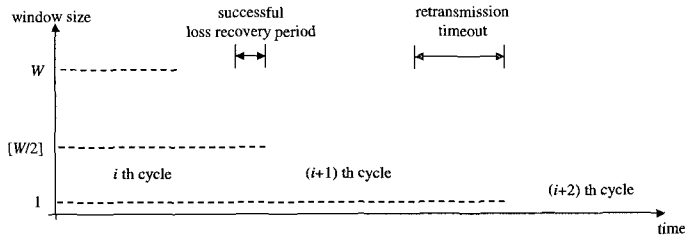

Fig. 1. Cyclic evolution model of the congestion window. If the lost packet in the current cycle can be recovered by fast retransmit, the next cycle starts in congestion avoidance (see the $(i+1)$th cycle). Otherwise, the next cycle should restart in slow start after RTO (see the $(i+2)$th cycle).

B. The Loss Window; Ω

In a cycle, suppose that the first lost packet belongs to the packets that are transmitted in the kth round. The ACKs by good packets transmitted in the kth round are received in the $(k+1)$th round. Let l_n be the nth lost packet in a cycle and $W(k)$ be *cwnd* at kth round. When l_1 is mth packet among $W(k)$ packets, a loss window, Ω, is the congestion window when all normal ACKs by $(m-1)$ packets are received. If $(m-1)$ ACKs do not switch the sender's state from slow start to congestion avoidance, the relation between Ω and $W(k)$ is given by

$$\Omega = \begin{cases} W(k) + (m-1), & \text{slow start} \\ W(k), & \text{congestion avoidance.} \end{cases} \tag{4}$$

Note that *cwnd* in slow start increases by one for every normal ACK while, in congestion avoidance, it increases by one after all ACKs by packets the sender has transmitted in the previous round are received. Consequently, the first packet that Ω includes is always the first lost packet. If we denote the lower (i.e., left) boundary of a by $L[a]$, then we have

$$L[\Omega] = L[l_1]. \tag{5}$$

C. The Condition for Successful Fast Retransmit

When n packets are lost for within Ω of u packets, we denote the number of packets that are not lost or transmitted newly in the kth round of loss recovery period by Φ_k; e.g., Φ_1 is always equal to $u-n$. For $h \geq 2$, Φ_h is equal to the number of packets that are transmitted by the slide and the inflation of the usable window during fast recovery. In order to recover n packet losses by n fast retransmits, each value of $\Phi_1, \Phi_2, \Phi_3, \cdots, \Phi_n$ should be greater than K. However, the condition $\Phi_n \geq K$ includes the conditions that each value of $\Phi_1, \Phi_2, \cdots, \Phi_{n-1}$ should be greater than K. Therefore, the condition that n lost packets in Ω may be recovered by n fast retransmits is simplified by

$$\Phi_n \geq K. \tag{6}$$

[3]This type of ACK is called 'partial ACK' in general. About partial ACK, the readers can refer to [5] and [6].

[4]For multiple packet losses, it is known that TCP Tahoe, an earlier TCP implementation before TCP Reno, outperforms TCP Reno. It is because TCP Tahoe has no fast recovery phase. When multiple packets are lost in a window, the sender of TCP Tahoe recover only the first lost packet by fast retransmit and other lost packets are retransmitted in slow start after the fast retransmission. Thus, the packets that have already been successfully delivered may be retransmitted falsely [11]. However, it can be rather efficient to avoid RTO, even if it should restart in slow start.

[5]Note that the round in this paper is similar to 'mini-cycle' defined in [12] and 'epoch' in [13].

The detailed derivations of Φ_n are done in Section IV.

IV. DERIVATIONS OF Φ_n

A. One Packet Loss ($n = 1$)

For $\Omega = u$, Φ_1 is equal to the number of well-transmitted packets out of u packets, namely, $u - 1$. These Φ_1 packets are going to generate duplicate ACKs for l_1. Thus, we have

$$\Phi_1 = u - 1. \tag{7}$$

B. Two Packet Losses ($n = 2$)

After the fast retransmit of l_1, Φ_2 new packets included by the inflation of the window are transmitted during the first fast recovery period. It means that the number of duplicate ACKs to be received for l_2 is equal to Φ_2. Considering that duplicate ACKs for l_1 inflates window by $u - 2$ and u packets are still outstanding, Φ_2 is given by

$$\Phi_2 = \lfloor u/2 \rfloor + (u - 2) - u = \lfloor u/2 \rfloor - 2. \tag{8}$$

If $\Phi_2 \geq K$, l_2 can be recovered by fast retransmit.

There can be an exception when l_2 is the last packet in Ω. When the sender receives the first ACK for l_2 by fast retransmitted l_1, the window slides till its lower boundary corresponds to l_2. In such a case, if $\lfloor u/2 \rfloor - 1 = K$, l_2 can be recovered by fast retransmit. However, the probability for this case is too low to be considered. For $\Phi_2 \gg K$, even if some packets are lost among Φ_2 packets, l_2 can be fast retransmitted if at least K packets are not lost. However, if these packet losses cannot be recovered by fast retransmit, RTO cannot be avoided in almost all cases. In practice, it is almost impossible for TCP Reno to recover packet losses occurred during loss recovery by fast retransmit due to lack of duplicate ACKs in the next cycle. Therefore, we assume that if a loss occurs on the new packets transmitted during fast recovery, RTO always takes place.

C. Three Packet Losses ($n = 3$)

Fig. 2 shows the loss recovery behavior of TCP Reno for three lost packets in Ω. Each round is explained as follows:

- Loss recovery starts at round i.
- The last duplicate ACK for l_1 is received at round $(i + 1)$.
- The first ACK for l_2 is received by retransmitted l_1 at round $(i + 1.5)$.
- The last duplicate ACK for l_2 is received at round $(i + 2)$.

When the initial loss window is denoted by $\Omega(i)$ and $R[a]$ denotes the right boundary of a, each boundary value is set to have $L[\Omega(i)] = 0$, $R[\Omega(i)] = u$, $R[l_2] = j$, $R[l_3] = k$, $R[\Omega(i+1)] = x$, $R[\Omega(i+1.5)] = y$, and $R[\Omega(i+2)] = z$. For every j and k that respects $2 \leq j \leq u - 1$ and $j + 1 \leq k \leq u$ in respect, the value of x, y, and z are given by

$$\begin{aligned} x &= \lfloor u/2 \rfloor + (u - 3) \\ y &= (j - 1) + \lfloor u/2 \rfloor \\ z &= (j - 1) + \lfloor u/4 \rfloor + \Phi_2. \end{aligned} \tag{9}$$

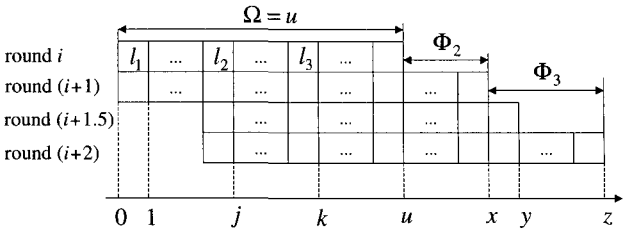

Fig. 2. Model of window behaviors of TCP Reno when three packets are lost in a loss window.

If $y \geq x$ at round $(i + 1.5)$, $y - x$ packets are transmitted. Since l_2 has not been retransmitted yet, they may generate duplicate ACKs for l_2. Considering these two cases, Φ_3 is given by

$$\begin{aligned} \Phi_3 &= z - \max(x, y) \\ &= \begin{cases} (j - 1) + \lfloor u/4 \rfloor - u, & 2 \leq j \leq u - 2 \\ \lfloor u/4 \rfloor - 3, & j = u - 1. \end{cases} \end{aligned} \tag{10}$$

Unlike Φ_1 and Φ_2, Φ_3 has relation with the position of l_2 as well as the value of Ω. When j has a maximum value, z increases most by (9). It is because window slides most when $j = u - 1$. Therefore, the minimum value of Ω for recovery of three lost packets without RTO can be obtained by $\lfloor u/4 \rfloor - 3 = K$. Its value is 24 if K has a typical value, three. For $2 \leq j \leq u - 2$, the condition for l_3 to be recovered is given by

$$(j - 1) + \lfloor u/4 \rfloor - u \geq K. \tag{11}$$

It means that l_3 can be recovered by fast retransmit if there are at least $u - \lfloor u/4 \rfloor + (K - 1)$ between l_1 and l_2.

D. Four Packet Losses ($n = 4$)

Fig. 3 shows the loss recovery behavior of TCP Reno for four lost packets in a window to be recovered. Each round is explained as follows:

- Loss recovery starts at round i.
- The last duplicate ACK for l_1 is received at round $(i + 1)$.
- The last duplicate ACK for l_2 is received at round $(i + 2)$.
- The first ACK for l_3 is received by retransmitted l_2 at round $(i + 2.5)$.
- The last duplicate ACK for l_3 is received at round $(i + 3)$.

Each boundary value is set to have $L[\Omega(i)] = 0$, $R[\Omega(i)] = u$, $R[l_2] = j$, $R[l_3] = k$, $R[l_4] = l$, $R[\Omega(i + 2)] = x$, $R[\Omega(i + 2.5)] = y$, and $R[\Omega(i + 3)] = z$. The values of x, y, and z are given by,

$$\begin{aligned} x &= (j - 1) + \lfloor u/4 \rfloor + \Phi_2 \\ y &= (k - 1) + \lfloor u/4 \rfloor \\ z &= (k - 1) + \lfloor u/8 \rfloor + \Phi_3 \\ \Phi_4 &= z - \max(x, y). \end{aligned} \tag{12}$$

As $x - y = (j - k) + \Phi_2 > 0$, the output of $\max(x, y)$ is always x. In practice, Φ_4 can be greater than zero in no case. In order to prove it, let $\hat{\Phi}_4$ be the maximum value of Φ_4, then we have

$$\hat{\Phi}_4 = z_{\max} - x_{\min}, \tag{13}$$

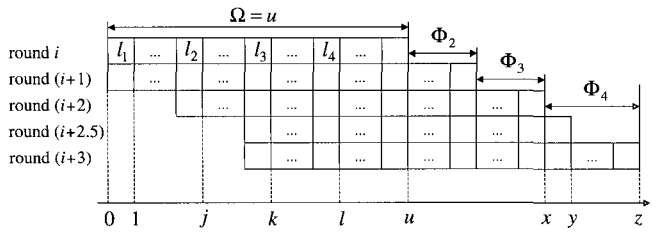

Fig. 3. Model of window behaviors of TCP Reno when four packets are lost in a loss window.

where $x_{\min}$ is the minimum value of x and $z_{\max}$ is the maximum value of z. When the distance between l_2 and l_3 is longest, that is, the window slides most by receiving an ACK for l_3 by retransmitted l_2, z has the maximum value. Thus, we have

$$\begin{aligned} z_{\max} &= z|_{j=j_{\min},k=u-1} \\ &= (j_{\min}-1)+\lfloor u/4\rfloor+\lfloor u/8\rfloor-2 \\ x_{\min} &= x|_{j=j_{\min}} \\ &= (j_{\min}-1)+\lfloor u/4\rfloor+\Phi_2 \\ &= (j_{\min}-1)+\lfloor u/2\rfloor+\lfloor u/4\rfloor-4, \end{aligned} \tag{14}$$

where $j_{\min}$ is the minimum value of j that satisfies (11). From (14), we have

$$\begin{aligned} \hat{\Phi}_4 &= z_{\max}-x_{\min} \\ &= 2-(\lfloor u/2\rfloor-\lfloor u/8\rfloor). \end{aligned} \tag{15}$$

Considering the minimum value for l_3 to be recovered by fast retransmit is given by $\lfloor u/4\rfloor-4=K$, $\hat{\Phi}_4$ can be positive in no case.

V. PROBABILISTIC ANALYSIS

A. The Packet Loss Models

We consider that the characteristic of packet loss is random and correlated. For random packet loss, each packet is lost with probability p and losses are independent as in [14] and [17].

For correlated packet losses, a first-order Markov chain is adopted as in [15], [16], and [18]. This model is commonly adopted for the channel with multi-path fading in a wireless environment. The transition probability matrix, Q_c, of the Markov chain is given by

$$Q_c=\begin{pmatrix} p_{BB} & p_{BG} \\ p_{GB} & p_{GG} \end{pmatrix}. \tag{16}$$

It is assumed that the chain is embedded at the beginnings of packet transmissions. Whether the current packet may be lost or not depends on the transmission result of the previous packet.

The channel has two states that are the 'good' state and the 'bad' state. The packet is lost with probability 1 while in the bad state. We define p_{BG} as the conditional probability that the present packet is transmitted successfully given that the previous packet is lost. The other entries in the matrix can be defined similarly. If we let the values of p_{GB} and p_{BG} to be α and β, respectively, the steady-state probabilities that the channel is in the good state (Π_G) and the bad state (Π_B) are given by

$$\Pi_G=\frac{\beta}{\alpha+\beta} \text{ and } \Pi_B=\frac{\alpha}{\alpha+\beta}. \tag{17}$$

Table 1. Parameters α and β at different values of Π_B and f_dT.

f_dT	Π_B	F (dB)	α	β	$1/\beta$
0.01	10^{-3}	29.99	0.00067	0.67129	1.49
	10^{-2}	19.98	0.00249	0.24610	4.06
	10^{-1}	9.77	0.00815	0.07337	13.63
	$5\cdot10^{-1}$	1.59	0.02090	0.02090	47.83
0.05	10^{-3}	29.99	0.00098	0.97983	1.0206
	10^{-2}	19.98	0.00837	0.82883	1.2066
	10^{-1}	9.77	0.03959	0.35631	2.8066
	$5\cdot10^{-1}$	1.59	0.10408	0.10408	9.61
0.1	10^{-3}	29.99	0.00100	0.99459	1.0054
	10^{-2}	19.98	0.00958	0.94795	1.0549
	10^{-1}	9.77	0.07006	0.63051	1.5860
	$5\cdot10^{-1}$	1.59	0.20534	0.20534	4.87

Further, the average duration of the bad state is given by $1/\beta$ while the average duration of the good state is given by $1/\alpha$. The unit of the duration is the number of packets because we assume that the state transits per every transmission time of a packet.

B. The Parameters for Correlated Packet Loss Model

In order to determine the parameters for correlated packet loss model of (16), we consider a wireless fading channel characterized by *Rayleigh* distribution. We assume that the TCP/IP stack is directly placed at the base station of wireless broadband (WiBro) that is going to be standardized as a specification of wireless MAN by telecommunications technology association (TTA), which is a standard developing organization (SDO) of Korea [21]. In brief, WiBro is considered to operate at the frequency of 2.3 GHz with frequency reuse factor of 1 and frequency efficiency of 6 and 2 (bps/Hz/Cell) in uplink and downlink, respectively.

In order to apply the Markov model at the TCP level, a wireless link of 5 Mbps and a constant packet size of 1,400 (bytes) are assumed in our numerical evaluations. The average packet loss probability (Π_B) depends on the physical characterization of the channel, usually expressed in terms of the *fading margin* (F). By choosing different values of the *normalized Doppler bandwidth* (f_dT), we can establish fading channel models with different degrees of correlation in the fading process. When f_dT is small, the fading process is very correlated, which means long bursts of packet losses; on the other hand, for a large value of f_dT, packet losses are almost dispersed similar to for random packet loss model. For different values of $f_dT(=0.01, 0.05,$ and $0.1)$, Table 1 shows the parameters of α, β, and the average length of a packet loss burst ($1/\beta$). For detailed procedures of determining the parameters, [18] can be referred to.

C. The Fast Retransmit Probability

As described earlier, fast retransmit is the only mean for TCP Reno to recover a packet loss without RTO. Therefore, we have termed *fast retransmit probability* to measure the capability of TCP Reno in handling packet losses by retransmission.

Simply, the fast retransmit probability is defined as the ratio of the number of packet losses retransmitted to the total number of packet losses, and indicates how many packet losses may be recovered by fast retransmit. In order to average the fast retransmit probability, it is needed to normalize the fast retransmit probability derived in terms of a loss window size (u) to the stationary distribution of u. In Section V-D, we discuss about how we can obtain the distribution of u in steady-state.

If we denote the fast retransmit probability of TCP Reno by R_R, then we have

$$R_R = \sum_n \sum_{u=1}^{W_{\max}} R_R^{(n)}(u)\pi_n(u), \tag{18}$$

where $\pi_n(w)$ is the probability when Ω is equal to w in steady-state, and $R_R^{(n)}(u)$ is given by

$$\begin{aligned} R_R^{(n)}(u) = \; & P\{(n-1)\text{packets losses in}(u-1)\text{packets}\} \\ & \times P\{\text{no packet loss during loss recovery}\}. \end{aligned} \tag{19}$$

For random packet loss, $R_R^{(n)}(u)$ in terms of n can be written as follows:[6]

$$\begin{aligned} R_R^{(1)}(u) &= (1-p)^{\Phi_1}(1-p) \\ &= (1-p)^u \\ R_R^{(2)}(u) &= \binom{u-1}{1}p(1-p)^{(u-2)}(1-p)^{\Phi_2}(1-p)^2 \\ &= (u-1)p(1-p)^{u+\Phi_2} \\ R_R^{(3)}(u) &= \binom{\lfloor u/4 \rfloor - K}{2}p^2(1-p)^{(u-3)}(1-p)^3(1-p)^{\Phi_2+\Phi_3} \\ &= \binom{\lfloor u/4 \rfloor - K}{2}p^2(1-p)^{u+\Phi_2+\Phi_3}. \end{aligned} \tag{20}$$

Finally, the total fast retransmit probability for random packet loss is given by

$$\begin{aligned} R_R(u) &= R_R^{(1)}(u) + R_R^{(2)}(u) + R_R^{(3)}(u) \\ &= (1-p)^u \left\{ \begin{array}{l} 1 + (u-1)p(1-p)^{\Phi_2} \\ + \binom{\lfloor u/4 \rfloor}{2}p^2(1-p)^{\Phi_2+\Phi_3} \end{array} \right\}. \end{aligned} \tag{21}$$

For the correlated packet loss model, the sequence of packets should be considered to calculate $R_R^{(n)}(u)$. For $n = 1$, the channel should transit to good state after a packet is lost and stay in the good state till the transmission of u packets that include the retransmitted packet are completed. For $n = 2$, let $R_R^{(2)}(u)_{su}$ be the probability when two lost packets are successive, and $R_R^{(2)}(u)_{ns}$ be the probability when two lost packets are not successive. Then, $R_R^{(2)}(u)$ is sum of $R_R^{(2)}(u)_{su}$ and $R_R^{(2)}(u)_{ns}$. For $n = 3$, l_2 and l_3 may be successive or not for $j_{\min} \le j \le u-2$ while they are always successive for $j = u-1$ by the given condition. Consequently, $R_R^{(n)}(u)$ for correlated packet loss is given by

$$\begin{aligned} R_R^{(1)}(u) &= \beta(1-\alpha)^{\Phi_1} \\ R_R^{(2)}(u) &= \beta(1-\beta)(1-\alpha)^{u+\Phi_2-2} \\ &\quad + \binom{u-2}{1}\alpha\beta^2(1-\alpha)^{u+\Phi_2-3} \\ &= \beta(1-\alpha)^{u+\Phi_2-3}\{(1-\alpha)(1-\beta) + (u-2)\alpha\beta\} \\ R_R^{(3)}(u) &= \left\{ \begin{array}{ll} \alpha\beta^2(1-\beta)(1-\alpha)^{\Phi_2+\Phi_3-2}, & j = u-1 \\ \alpha\beta^2(1-\alpha)^{\Phi_2+\Phi_3-3} & \\ \times\{N_{su}(1-\alpha) + N_{ns}(1-\beta)\}, & \text{else,} \end{array} \right. \end{aligned} \tag{22}$$

where N_{su} is the number of cases that l_2 and l_3 are successive and N_{ns} is the number of cases that they are not. The values of N_{su} and N_{ns} are determined as follows:

$$\begin{aligned} N_{su} &= u - (j_{\min} - 1) - 2 \\ N_{ns} &= \binom{u-(j_{\min}-1)}{2} - (N_{su} - 1). \end{aligned} \tag{23}$$

D. Markov Process

As mentioned earlier, the window evolution of TCP can be analyzed by adopting a Markov chain. The stationary distribution of Markov chain $\{\Omega_i\}$ can be obtained numerically if the exact transition probabilities can be determined. We follow the procedures in [14] and [15] to get the transition probabilities except that the congestion window size after loss recovery does not always decrease to $\lfloor u/2 \rfloor$. The relation between the number of lost packets and the window size in the next cycle is defined by (3).

When RTO occurs, W_{th} of the next cycle can be inferred as follows. For $\Omega = u$, suppose two packets among packets are lost and RTO occurs. If $u \ge K+2$, at least the first lost packet may be recovered by fast retransmit. In this case, we can sure that RTO is invoked by the second lost packet. Therefore, W_{th} in the next cycle is set to $\lfloor u/2 \rfloor$. Depending on the size of Ω and the number of lost packets, W_{th} in the next cycle may be one of $\lfloor u/2 \rfloor$, $\lfloor u/4 \rfloor$, and $\lfloor u/8 \rfloor$.

For correlated packet loss model, it is impossible to know the state of the channel during RTO in which no packet can be transmitted. However, if the first packet is not lost in the next cycle, it assures that the channel is in the good state. That is, regardless of the number of consecutive RTOs, it can be sure that the state of the channel when the next cycle starts is always good. Therefore, we consider the process $\{\Omega_i\}$ over the state space $\{2, 3, 4, \cdots, W_{\max}\}$. Note that the number of successive RTOs does not affect the evolution of the congestion window process.

VI. NUMERICAL RESULTS

In this section, we show the fast retransmit probability defined by (18) for random and correlated packet loss. The x-axis of each graph indicates packet loss probability, which corresponds to p for random packet loss and to Π_B for correlated packet loss. We set K to have the value of three in all cases.

Figs. 4–6 show the fast retransmit probability for different values of $W_{\max}$ of 8, 32, and 128 when packet losses are random and correlated for $f_dT = 0.01, 0.05$, and 0.1. As packet loss probability increases, we can see that the fast retransmit

[6]The derivation of $R_R^{(n)}(u)$ is based on the conditions presented in Section IV. Detailed procedures are not included in this paper due to its insignificance.

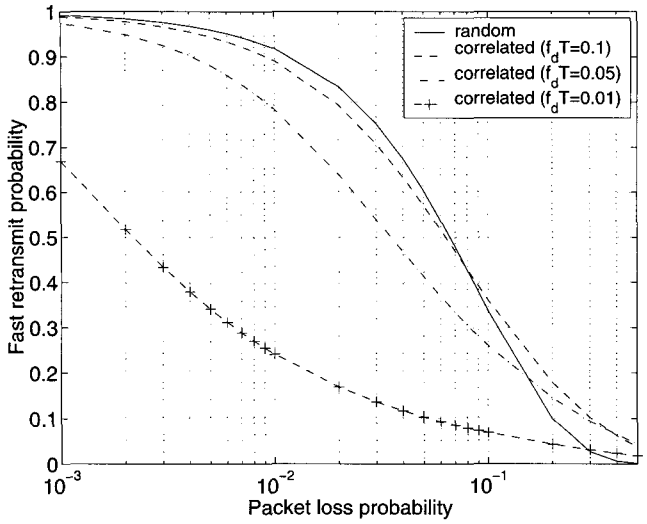

Fig. 4. Fast retransmit probability of TCP Reno for random and correlated packet loss ($W_{\max} = 8$).

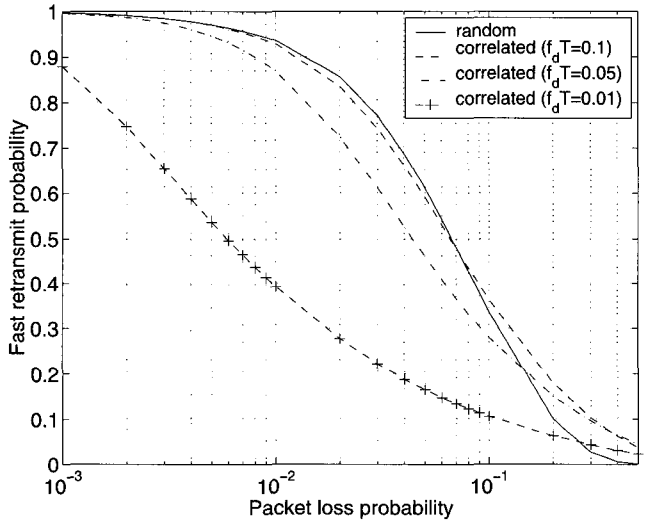

Fig. 5. Fast retransmit probability of TCP Reno for random and correlated packet loss ($W_{\max} = 32$).

probability of all cases decreases as expected. It means that the fast retransmit cannot be triggered well due to lack of duplicate ACKs.

It is very important observation that as f_dT decreases, i.e., as average length of a packet error burst increases, the fast retransmit performance degrades seriously. As addressed in Section IV, more than three successive packet losses can be recovered without RTO in no case. Therefore, for a value of β decreased below $1/3$, almost every packet loss event means invocation of RTO. On the other hand, when packet losses are random (or when average length or a packet error burst is close to 1 for $f_dT = 0.05$ and 0.1), most packet loss events correspond to a single packet loss case so that the fast retransmit probability maintains rather high values compared to the case when $f_dT = 0.01$. Note that a single packet loss in Ω can be recovered by fast retransmit only if the value of Ω is greater than or equal to 4.

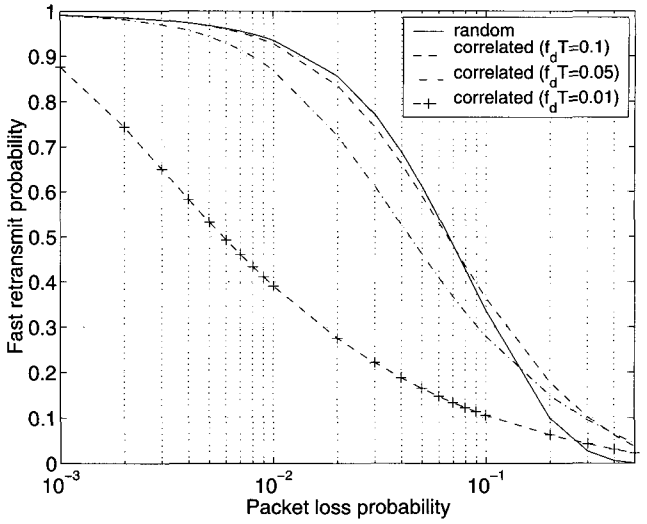

Fig. 6. Fast retransmit probability of TCP Reno for random and correlated packet loss ($W_{\max} = 128$).

Recalling the conditions for successful fast recovery of TCP Reno, the value of $W_{\max}$ should be greater than or equal to 10 to recover two packet losses by two fast retransmits. Therefore, when $W_{\max}$ is equal to 8, two or more packet losses always invoke RTO; i.e., $R_R^{(2)} = R_R^{(3)} = 0$, $R_R = R_R^{(1)}$. For the reason, it can be seen that the fast recovery probability for $f_dT = 0.01$ and $W_{\max} = 8$ is much smaller than the probability for $f_dT = 0.01$ and $W_{\max} = 32$ or 128.

Except in the case for $f_dT = 0.01$ and $W_{\max} = 8$, the increment of $W_{\max}$ makes no significant difference to the fast recovery probability. As mentioned earlier, it is because the number of packet losses in window is one in most cases. As packet loss probability increases, two or more packets tend to be lost in a window. However, at the same time, the sender cannot keep *cwnd* large enough so that, even if $W_{\max}$ is set to a large value, the benefit obtained from the large value of $W_{\max}$ is insignificant. According to the conditions derived in Sections IV-B and IV-C, if K has a typical value of 3, Ω should be greater than or equal to 10 and 24 to retransmit two and three packet losses, respectively. Consequently, we can infer that overall R_R is dominated by $R_R^{(1)}$ in these cases.

In order to investigate the effect of window size on the fast retransmit probability, the average window size compared for different values of $W_{\max}$, n, and f_dT in Figs. 7–9. As mentioned in Section II-C, the window size of TCP Reno is affected by the number of packet losses recovered by fast retransmit. To reflect the effect, we calculate each average window size for three different values of n (= 1, 2, and 3) because over 4 packet losses in a window always invoke RTO. The average window size is defined by

$$W_n = \sum_{w=1}^{W_{\max}} w\pi_n(w) \text{ for } n = 1,\ 2,\ 3, \tag{24}$$

where $\pi_n(w)$ means the probability that $\Omega = u$ when Ω may decrease by $(1/2)^n$; i.e., W_2 means the stationary distribution of Ω that can be decreased by $1/2$ and $1/4$ after successful loss re-

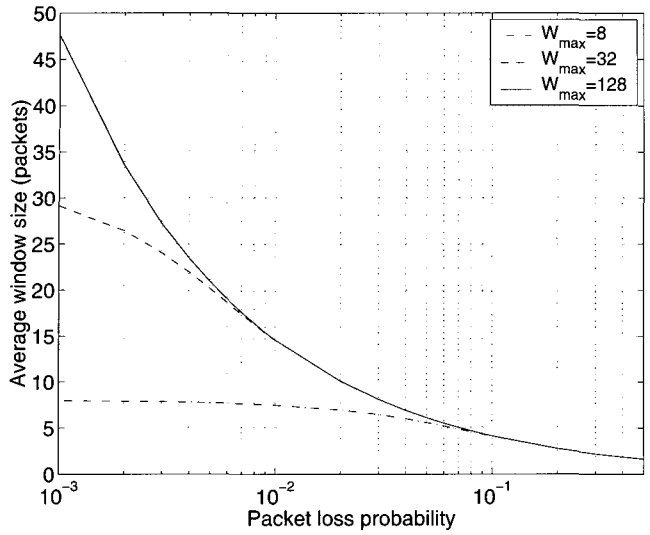

Fig. 7. Average window size for a single packet loss (W_1) when packet losses are random ($W_{\max} = 8, 32$, and 128).

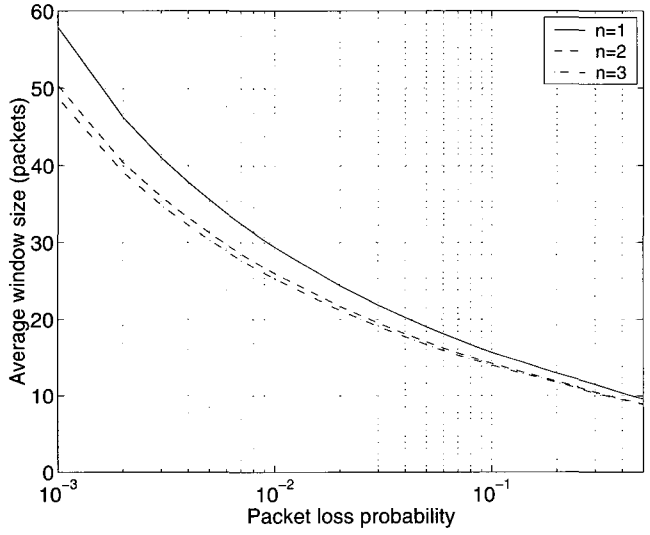

Fig. 8. Average window size for different values of n (W_1, W_2, and W_3) when $f_dT = 0.01$ ($W_{\max} = 128$).

covery (the implication of n is discussed in the later presentation of Fig. 8 in detail).

Fig. 7 corresponds to the case for a single packet loss (W_1) when packet losses are random, and shows the average window size for different values of $W_{\max}$. It can be seen that as p increases, two graphs for $W_{\max} = 32$ and 128 overlap perfectly for p exceeding 10^{-2}, and all three graphs overlap for p exceeding 10^{-1}. That is, a large $W_{\max}$ has no meaning for a packet loss probability exceeding a certain value, which explains the insignificance of $W_{\max}$ to the fast retransmit probability as shown in Fig. 6.

Fig. 8 provides the comparison of average window size for different values of n when $f_dT = 0.01$ and $W_{\max} = 128$.

As n increases, the corresponding window size decreases, which can be expected by (3). Therefore, the consideration of two or three successful retransmissions has an effect of lower-

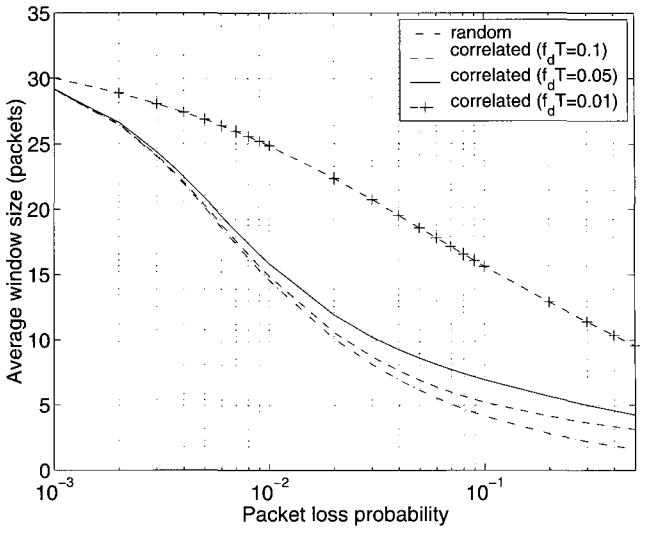

Fig. 9. Average window size for different types of packet loss models ($W_{\max} = 32$).

ing the average window size. The difference between $R_R^{(2)}$ and $R_R^{(3)}$ is much smaller than the difference between $R_R^{(1)}$ and $R_R^{(2)}$. It means that the frequency of decreasing *cwnd* by $1/8$ corresponding to three successful retransmissions is much less than that of decreasing *cwnd* by $1/4$.

We also consider the effect of correlated packet losses on the average window size in Fig. 9.

It is interesting to observe that the longer length of a packet error burst corresponds to the larger size of window size. It can be explained by the fact that, for a certain value of packet loss rate, as average length of a packet error burst increases, the average number of invocations of loss recovery phase rather decreases, even if it is not successful. That is, the sender decreases *cwnd* less frequently for the *grouped* packet losses than for the *scattered* packet losses. Although the larger value of n has an effect to make the window size smaller as can be seen in Fig. 8, the effect of the grouping overweighs the effect. In addition, the effect of the grouped packet losses explains the reason for the reversion of fast retransmit probability for packet loss probability exceeding 10^{-1} shown in Figs. 4–6. When packet losses are grouped, even if packet loss probability is of such a large value, the sender keeps *cwnd* large compared to the values for scattered packet losses so that it is able to have more chances to recover packet losses without RTO.[7]

For more detailed investigation of the loss recovery performance of TCP Reno, we show the fast retransmit probability isolated in terms of n in Figs. 10–13. We consider the case when $W_{\max}$ is equal to 32 and 128 because both $R_R^{(2)}$ and $R_R^{(3)}$ are equal to zero when $W_{\max}$ is equal to 8.

Figs. 10 and 11 show the fast retransmit probability for two packet losses, $R_R^{(2)}$, when $W_{\max}$ is equal to 32 and 128 in respect. Each line in both figures has the same characteristic that it rather increases as packet loss probability increases until packet

[7]Except the results shown in Figs. 7–9, the graphs of the average window size have almost similar characteristics for different values of $W_{\max}$, n, and f_dT so that they are not included in this paper.

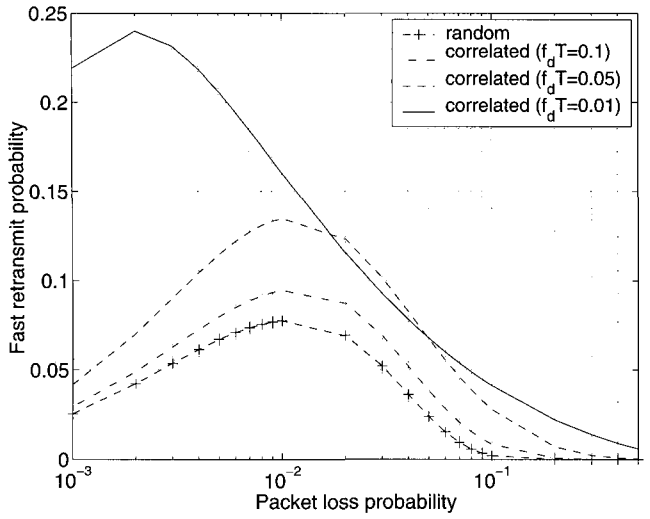

Fig. 10. Fast recovery probability of TCP Reno for two packet losses ($R_R^{(2)}$) and different types of packet losses ($W_{\max} = 32$).

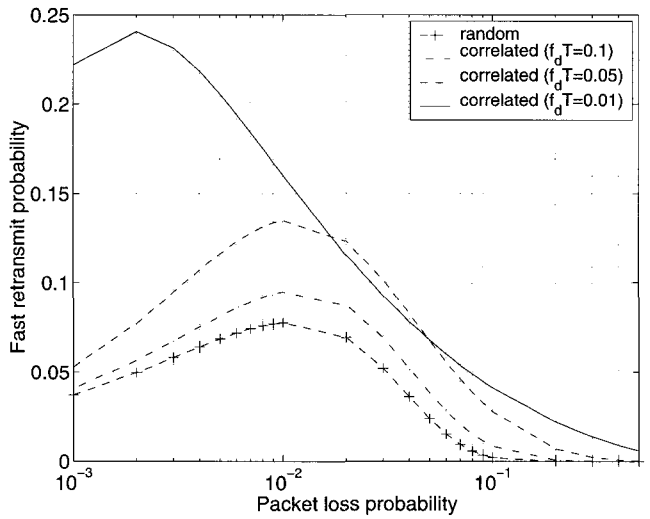

Fig. 11. Fast recovery probability of TCP Reno for two packet losses ($R_R^{(2)}$) and different types of packet losses ($W_{\max} = 128$).

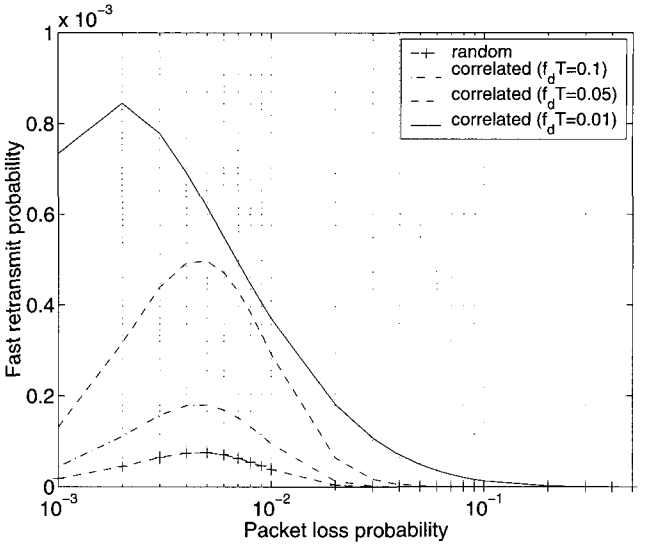

Fig. 12. Fast recovery probability of TCP Reno for three packet losses ($R_R^{(3)}$) and different types of packet losses ($W_{\max} = 32$).

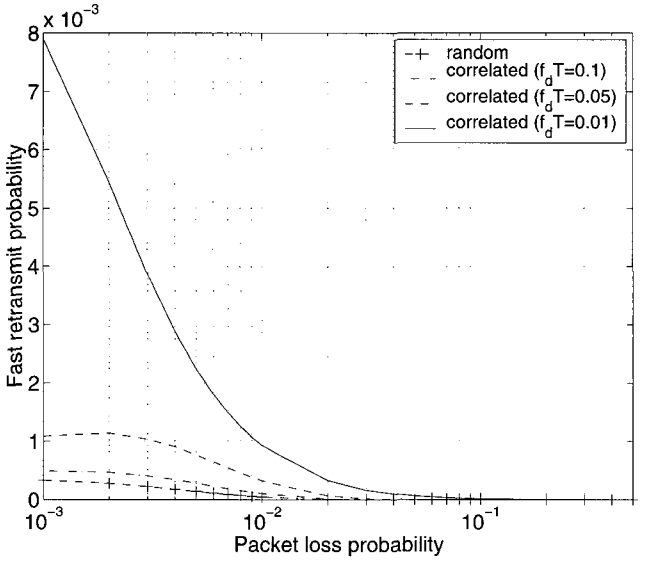

Fig. 13. Fast recovery probability of TCP Reno for three packet losses ($R_R^{(3)}$) and different types of packet losses ($W_{\max} = 128$).

loss recovery probability approaches to a certain value. Another observation is that the fast retransmit probability for most correlated packet losses (when $f_dT = 0.01$) is the highest of the four.

We can explain these two observations based on the frequency of the event that two packet losses are included in a window. As mentioned when we discuss on the result in Fig. 9, for a given packet loss probability, as the larger number of packet losses are grouped, more packet losses are likely to be included in a window. For example, when packet losses are random (or scattered with a large value of f_dT), the likelihood of two packet losses in a window is comparatively smaller than when packet losses are correlated. When packet losses are correlated, as packet loss probability increases, f_dT also increases. Therefore, when packet loss probability is low, the results for correlated packet losses have similar characteristic to the results when packet losses are random. For a large packet loss probability, even if two packet losses are included in a window, they cannot be recovered by fast retransmit due to lack of duplicate ACKs.

The shape of these two figures seems almost similar. We have already discussed about the reason for the insignificance of $W_{\max}$ to the fast retransmit probability. Note that the needed window size for two packet losses is just 10, which is the only matter of concern to the fast retransmit probability.

In Figs. 12 and 13, we also show the fast recovery probability of TCP Reno for three packet losses ($R_R^{(3)}$) when $W_{\max}$ is equal to 32 and 128 in respect. Similarly as the results for $n = 2$ shown in Figs. 10 and 11, the fast retransmit probability when $f_dT = 0.01$ is the highest throughout packet loss probability, which is also related to the frequency of the event that three

packet losses may be included in a window.

It can be seen that a large value of $W_{\max}$ benefits the fast recovery probability as shown in Fig. 13. It is because successful recovery of three packet losses requires a considerably large window size of 24. However, overall value of $R_R^{(3)}$ is extremely small that TCP Reno cannot be considered that it can recover three packet losses in a window. It reflects that the condition for three lost packets to be recovered by fast retransmits is very strict; it is related with the position of the second lost packet as well as the window size.

VII. CONCLUSION

In this paper, we have proposed a simple model which facilitates to capture and evaluate the loss recovery behaviors of TCP. On the basis of the developed model, the loss recovery performance of TCP Reno can be evaluated in a numerical way.

Our contribution can be summarized as following two facts:

- Adopting the proposed model, the rather complex behaviors of TCP loss recovery can be analyzed in a simple way.
- We have quantified the effect of correlated packet losses on the loss recovery performance of TCP.

About the first contribution, the accurate conditions for successful fast retransmit can be derived in terms of the number of packet losses in a window. Under the conditions, we can have revealed that three packets are recovered by three successive fast retransmits under very specific cases, and proved that over four packet losses in a window always invoke RTO as well. Therefore, we need not assume that more than three packet losses may always invoke RTO as in most previous works any more.

We have considered a practical wireless channel where packet losses may be correlated due to fading, and shown that even if the average packet loss probability is very low, the burst of the successive packet losses can degrade TCP performance seriously, which is the second contribution of this paper.

ACKNOWLEDGEMENTS

This work is a partial outcome of the project entitled "Improving TCP Performance over Wireless Networks" that was supported by Qualcomm incorporated Qualcomm Yonsei CDMA Joint Research Center.

REFERENCES

[1] J. Postel, "Transmission control protocol," RFC 793, 1981.
[2] W. Stevens, "TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," RFC 2001, 1997.
[3] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," RFC 2581, 1999.
[4] V. Jacobson, "Congestion avoidance and control," in *Proc. ACM SIGCOMM'88*, 1988, pp. 314–329.
[5] V. Jacobson, "Modified TCP congestion avoidance algorithm," message to end2end-interest mailing list, Apr. 1990, available at ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt.
[6] C. Hoe, "Improving the start-up behavior of a congestion control scheme for TCP," in *Proc. ACM SIGCOMM'96*, 1996.
[7] S. Floyd and T. Henderson, "The NewReno modification to TCP's fast recovery algorithm," RFC 2582, 1999.
[8] M. Mathis *et al.*, "TCP selective acknowledgement options," RFC 2018, 1996.
[9] E. Blanton *et al.*, "A conservative selective acknowledgement (SACK) - based loss recovery algorithm for TCP," RFC 3517, 2003.
[10] H. Balakrishnan *et al.*, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Trans. Networking*, vol. 5, no. 6, pp. 756–769, 1997.
[11] K. Fall and S. Floyd, "Simulation-based comparisons of Tahoe, Reno, and SACK TCP," *ACM Computer Communication Review*, vol. 26, no. 3, pp. 5–21, 1996.
[12] P. P. Mishira, D. Sanghi, and S. K. Tripathi, "TCP flow control in lossy networks: Analysis and enhancements," in *Proc. Computer Networks, Architecture, and Applications*, IFIP Transactions C-13, S. V. Raghavan, G. V. Bochman, and G. Pujolle, Eds. Amsterdam, The Netherlands: Elsevier North-Holland, pp. 181–193, 1993.
[13] T. V. Lakshman and Madhow, "The performance of TCP/IP for networks with high bandwidth-delay products and random loss," *IEEE/ACM Trans. Networking*, vol. 5, no. 3, pp. 336–350, 1997.
[14] A. Kumar, "Comparative performance analysis of versions of TCP in a local network with a lossy link," *IEEE/ACM Trans. Networking*, vol. 6, no. 4, pp. 485–498, 1998.
[15] A. Kumar and J. Holtzman, "Comparative performance analysis of versions of TCP in a local network with a mobile radio link," available at http://ece.iics.ernet.in/ anrug/.
[16] F. Anjum and L. Tassiulas, "On the behavior of different TCP algorithms over a wireless channel with correlated packet losses," in *Proc. ACM SIGMETRICS'99*, 1999, pp. 155–165.
[17] J. Padhye *et al.*, "Modeling TCP Reno performance: A simple model and its empirical validation," *IEEE/ACM Trans. Networking*, vol. 8, no. 2, pp. 133–145, 2000.
[18] M. Zorzi and A. Chockalingam, "Throughput analysis of TCP on channels with memory," *IEEE J. Select. Areas Commun.*, vol. 18, no. 7, pp. 1289–1300, 2000.
[19] A. Abouzeid, S. Roy, and M. Azizoglu, "Stochastic modeling of TCP over lossy links," in *Proc. IEEE INFOCOM 2000*, 2000, pp. 1724–1733.
[20] R. Braden, "Requirements for Internet hosts," RFC 1122, 1989.
[21] http://www.tta.or.kr/English/new/main/index.htm.
[22] B. Kim and J. Lee, "Analytic models of loss recovery of TCP Reno with packet losses," *Lecture Notes in Computer Science (LNCS)*, no. 2662, pp. 938–947, Oct. 2003.

Beomjoon Kim received the the B.S. degree in electronic engineering, the M.S., and Ph.D. degrees in electrical and electronics engineering all from Yonsei University, in 1996, 1998, and 2003, respectively. He joined a project to develop a 4G Wireless System called Y4G System of Center for Information and Telecommunication of Yonsei University in 2003. During the period, he reviewed several paper submitted to IEEE Journal on Selected Areas in Communications and IEEE Communications Letters. Currently, he is working for LG Electronics Inc. as a senior research engineer in the area of standardization including IEEE 802.16, 802.21, and IETF.

Jaiyong Lee received Ph.D. degree in Computer engineering from Iowa State University, USA in 1987. He has been with ADD (Agency for Defense Development) as a research engineer from 1977 to 1982, and with Computer Science Dept. of POSTECH as an associate professor from 1987 to 1994. Since 1994, he has been a professor in School of EE, Yonsei University. He is also actively serving in the IT areas such as Executive Director of KICS (Korea Institute of Communication Sciences), Executive Vice President of OSIA(Open standards and Internet Association), and Editor of JCN and ETRI journal. Recently, he has been working for standard activity, especially in IETF, as an IT professional of MIC(Ministry of Information and Communication Republic of Korea) for 2 years. Now he is performing research works in the areas of QoS and mobility management for supporting 4G architecture, MAC and multicast protocol design for wireless access network, and sensor MAC/routing protocol design.