Sādhanā Vol. 32, Part 3, June 2007, pp. 253–275. © Printed in India

TCP with header checksum option for wireless links: An analytical approach towards performance evaluation

PAWAN KUMAR GUPTA and JOY KURI

Centre for Electronics Design and Technology, Indian Institute of Science, Bangalore 560 012
e-mail: pawankumargupta@hotmail.com, kuri@cedt.iisc.ernet.in

MS received 9 June 2004; revised 26 May 2006

Abstract. TCP performs poorly in wireless mobile networks due to large bit error rates. Basically, the TCP sender responds to these losses as if they were due to congestion in the network, and reduces the congestion window unnecessarily. In earlier work, it has been shown that adding a *TCP header checksum* is very useful in differentiating between congestion loss and corruption loss. With the modified TCP, receivers can explicitly indicate corruption of received packets by generating "Explicit Loss Notifications (ELNs)." This paper focuses on an analytical study of this modified TCP protocol. We derive an expression for the probability of a receiver generating successful ELN, assuming a generic link layer protocol for data transfer over wireless links. Next, we develop an analytical approach for TCP throughput evaluation under the modified scheme. We compare the throughput results obtained by analysis and simulation, and find very close agreement between the two sets. We also compare the performance of the modified scheme with the standard NewReno TCP, and find considerable improvement in data throughput over wireless links.

Keywords. ELN; NewReno; NewRenoEln; TCP; throughput; wireless links.

1. Introduction

Transmission Control Protocol (TCP) is a reliable, end-to-end, transport protocol widely used to access the Internet and to support applications like telnet, ftp and http (Stevens 1994, Postel). TCP was designed for wired networks, where packet losses are mainly due to congestion. In today's world, more and more people use their mobile devices to access the Internet either for work or entertainment, where TCP runs over wireless links and is subjected to more corruption losses as compared to congestion losses. Current TCP implementations respond to any such loss of packet in the traditional way by reducing the congestion window. This unnecessary reduction in the congestion window reduces throughput and wastes precious wireless resources.

A lot of research has been done in this area, and many studies have proposed different ways of increasing throughput for TCP transmission over wireless networks (Balakrishnan *et al* 1997, Goff *et al* 2000, Chen & Lee 2000, Caceres & Iftode 1995, Sinha *et al* 1999, Bakshi

et al 1997). These proposals mainly focus on hiding corruption losses from the TCP sender by performing re-transmissions of any lost data to avoid coarse timeouts. Re-transmissions may be performed either at the link level when a reliable link layer protocol is used (Balakrishnan *et al* 1997) or at the packet level when split connection protocol is used (Chen & Lee 2000). Balakrishnan & Katz (1998), Peng & Ma, Vaidya (1997), Parsa and Aceves (2000) have discussed mechanisms to successfully differentiate congestion loss from corruption loss in various scenarios. The schemes suggested in (Balakrishnan & Katz 1998, Peng & Ma) rely on intermediate nodes to provide sufficient information for distinguishing corruption loss from congestion loss. Such mechanisms are not useful when the IP traffic is encrypted, or when incorporating TCP-awareness in intermediate nodes is not feasible. Vaidya (1997), Parsa & Aceves (2000) suggest the use of successive inter-arrival times of packets to heuristically distinguish corruption loss from congestion loss. These mechanisms work on end-to-end basis but are not very reliable.

As shown in Balan *et al* (2001) and Gupta & Kuri (2002), the use of a new TCP option— the TCP *header checksum* option— is an attractive approach for distinguishing between congestion loss and corruption loss of TCP segments over wireless links. If TCP is enhanced to generate and process this header checksum option, then explicit indications of loss on wireless links can be obtained. The enhanced TCP is called "TCP HACK" in Balan *et al* (2001) and "TCP NewRenoELN" in Gupta & Kuri (2002)—essentially, these are identical schemes.

In this paper, we are interested in an analytical study of the proposals in Balan *et al* (2001) and Gupta & Kuri (2002). The first question of interest is: Given that a TCP segment is in error, what is the probability that an "Explicit Loss Notification" (ELN) can be generated? The issue here is that a TCP segment may be fragmented into several pieces because of a small MTU (maximum transmission unit) on the wireless link, and an unfortunate pattern of link-level frame corruption may make it impossible for the TCP receiver to generate ELN. In § 3, we analyse the ELN generation process and answer this question.

Secondly, we are interested in obtaining analytical expressions for the TCP *throughput* that can be achieved when the modified TCP is used. For this, we extend the analysis developed in Zorzi *et al* (2000) and obtain formulae for computing the throughput. Berkeley's network simulator ns is used to simulate the modified TCP, and the simulation results are in excellent agreement with the formulae. The analytical expressions provide a clear, quantitative picture of the benefit that the modified TCP brings, when compared to standard NewReno TCP.

This paper is organized as follows. In § 2, we start with a brief explanation of TCP NewReno and then describe the enhancements to TCP proposed in Balan *et al* (2001) and Gupta & Kuri (2002). For ease of discussion, we refer to the modified TCP proposed in Balan *et al* (2001) and Gupta & Kuri (2002) as "NewReno ELN"—NewReno with ELN. In § 3, we develop an expression for the conditional probability of a receiver generating successful ELN, given that a TCP segment is received in error. In § 4, we develop a detailed analytical model for the performance evaluation of the NewRenoELN TCP protocol. In § 5, we show that simulation results match the analysis very closely, and also note that NewRenoELN performs appreciably better than NewReno.

2. The NewRenoEln TCP protocol

In this section, we start with a brief description of the TCP receiver and transmit processes specific to NewReno, (Postel Floyd & Henderson, Kumar 1998). The TCP receiver can receive packets out of sequence, but will only deliver them in sequence to the TCP user. The receiver

advertises a maximum window size of $W_{\max}$, so that the transmitter does not allow more than $W_{\max}$ unacknowledged packets outstanding at any given time. The receiver sends back an acknowledgement (ACK) packet for every data packet that it receives correctly. The ACKs are cumulative, i.e., an ACK carrying the acknowledgement number m acknowledges all data packets up to, and including, the data packet with sequence number $(m - 1)$. If a packet is lost (after a stream of correctly received packets), then the transmitter keeps receiving ACKs with the sequence number of the first packet lost (called duplicate ACKs), even if the packets transmitted after the lost packet are correctly received at the receiver.

The TCP transmitter operates on a window-based transmission strategy as follows. At any given time t, there is a lower window edge $A(t)$, which means that all TCP packets numbered up to, and including, $A(t)-1$ have been transmitted and acknowledged, and that the transmitter can send data packets from $A(t)$ onwards. The receipt of an ACK that acknowledges some new data will cause an increase in $A(t)$ by an amount equal to the amount of data acknowledged. The transmitter's congestion window, $W(t)$, defines the maximum amount of unacknowledged data packets the transmitter is permitted to send, starting from $A(t)$. The slow start threshold, $W_{th}(t)$, controls the increments in $W(t)$. Each time a new packet is transmitted, the sender starts a timer. If the timer reaches the round trip timeout value (derived from a round trip time estimation procedure Stevens (1994)) before the packet is acknowledged, *timeout* timer expiration occurs. The evolution of $W(t)$ and $W_{th}(t)$ are triggered by ACKs and timeouts as described below.

(1) If $W(t) < W_{th}(t)$ (*slow start phase*), each ACK causes $W(t)$ to be incremented by 1.
(2) If $W(t) \geq W_{th}(t)$, each ACK causes $W(t)$ to be incremented by $1/W(t)$. This is the *congestion avoidance phase*.
(3) If *timeout* occurs at the transmitter at time t, $W(t^+)$ is set to 1, $W_{th}(t^+)$ is set to $\lceil W(t)/2 \rceil$, and the transmitter begins retransmission from the next packet after the last acknowledged packet.
(4) When the K^{th} (dupack threshold) duplicate ACK is received, $W_{th}(t^+)$ is set to one half of the current "*flightsize*" and $W(t^+)$ is set to $W_{th}(t^+) + K$. Addition of K inflates the congestion window by the number of packets that have left the network and which the receiver has buffered. The *flightsize* is defined as the number of packets that are transmitted and not yet acknowledged. The sender will also record the highest sequence number transmitted in the variable "*recover*". The missing packet indicated by the duplicate ACK is re-transmitted at this stage. This is the *fast re-transmission phase*.
(5) In the *fast recovery phase*, each time another duplicate ACK arrives, $W(t)$ is incremented by one and a new packet is transmitted (if allowed by the new value of congestion window). Increment of one inflates the congestion window to reflect the additional packet that has left the network.
(6) When an ACK packet arrives that acknowledges all the data up to and including "*recover*", then the ACK acknowledges all the intermediate packets sent between the original transmission of the lost packet and the receipt of the K^{th} duplicate ACK. The congestion window is now set to the current value of slow start threshold, i.e., $W_{th}(t^+)$. The sender exits the *fast recovery phase* and transmission of packets resumes in *congestion avoidance phase*.
(7) If this new ACK does not acknowledge all of the data up to and including "*recover*", then this is a *partial* ACK. In this case, the sender re-transmits the first unacknowledged segment indicated by the ACK packet. It deflates the congestion window $W(t)$ by the amount of new data acknowledged, then adds back one, and sends a new segment if permitted by the new value of congestion window.

In the NewReno protocol, the receiver will respond in the same manner to all packet losses, which may occur either due to congestion or corruption on the wireless link. For packet losses due to corruption, there is no need for the TCP sender to reduce the congestion window. The reduction in congestion window reduces the rate of flow of packets, thereby degrading the throughput. On the other hand, if the packet loss is due to congestion, then the TCP sender should reduce the congestion window to avoid congestion collapse in the network. Thus, an effective mechanism is required to reliably distinguish congestion losses from corruption losses. With the modifications suggested in Balan *et al* (2001) and Gupta & Kuri (2002), it is possible for the TCP sender and receiver to effectively distinguish congestion losses from corruption losses and take action accordingly. These modifications are described here in brief.

(1) During connection establishment, sender (server) and receiver (client) exchange information on whether to use this new scheme or not, using SYN packets. While requesting for a connection, the receiver (mobile host) will add an option field (figure 1) to the SYN packet asking for permission to send ELN based on TCP header checksum along with ACK packets. The sender (correspondent node) will acknowledge the use of this option using the same option field along with the SYN packet in the reverse direction. This option will not be sent in non-SYN packets. If the sender does not receive this option in the SYN segment then it must not send header checksum on the data packets. If the receiver does not receive this option in the SYN segment, then it must not generate ACK packets with Explicit Loss Notification.
(2) During the data transfer phase, the TCP header and the pseudo-header are protected by their own checksum, which is different from the complete checksum that is currently calculated for the TCP header and data. The checksum will be calculated on all the fields of the TCP header and pseudo-header except the TCP checksum (original checksum that is calculated for both header and data). This header checksum will be carried in the option field (figure 2) that will be attached to the TCP header of each data packet transferred during data transfer phase.
(3) The TCP/IP protocol is modified in the receiver to separately calculate the checksum on the header portion. For packets that are successfully received, the checksum check on the complete TCP packet will pass and the receiver will generate the ACK packet according to the normal TCP protocol. In case the receiver gets a corrupted packet and finds that the checksum check on the complete packet as well as the header checksum fail, then it will drop the packet without further processing. Finally, if the receiver finds that the checksum check on the whole packet fails but the check on the header is successful, then surely this is a case of corruption of packet on the wireless link. In this case, the receiver can generate reliable ELN for the sender, as it now has the complete socket address information as well as the sequence number of the corrupted packet. To indicate the sequence number of the corrupted packet to the sender, the receiver will generate a duplicate ACK and it will also attach ELN ACK option field (figure 3) to the packet. These packets will be called ELN ACK packets.

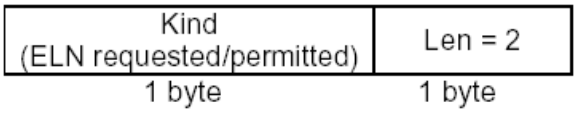

Figure 1. Explicit loss notification requested option, along with SYN packet from receiver to sender and ELN permitted option, along with SYN packet from sender to receiver.

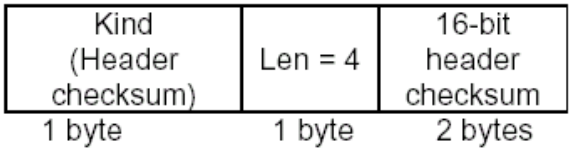

Figure 2. TCP header checksum option, along with data packets from sender to receiver.

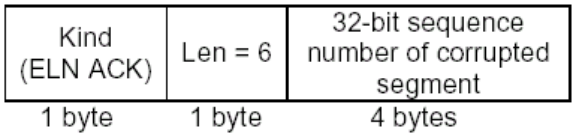

Figure 3. ELN ACK option along with duplicate ACK packets.

(4) When the sender receives an ACK packet without the ELN ACK option field, it will process it according to the NewReno protocol. If the sender receives a duplicate ACK with ELN ACK option attached, then it re-transmits the corrupted packet, as indicated by the ELN ACK option field. It also re-starts the timeout timer for this packet. The sender makes no changes to the congestion window parameters and the dupack counter (indicating number of duplicate ACKs received). Figure 4 shows the NewRenoEln connection establishment and data transfer phase.
(5) In case the re-transmitted packet is also corrupted, the TCP receiver will not generate ELN. The loss of successive packets indicates that the wireless link is in a bad state and further re-transmissions may also get corrupted.

3. Analysis of ELN generation process

In this section we start with a brief description of the system model that is considered for our analysis. We assume a system where the sender (fixed host) is connected to the receiver (mobile host) through an intermediate system feeding the wireless link (figure 5). This wireless link is the main source of corruption errors for the packets transmitted by the sender. For the sake of analysis we assume that congestion losses on the wired link are very few and can be ignored

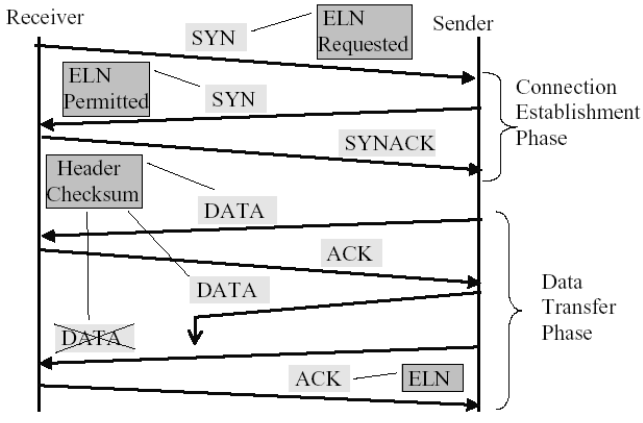

Figure 4. Timeline indicating connection establishment and data transfer phase for TCP NewRenoEln.

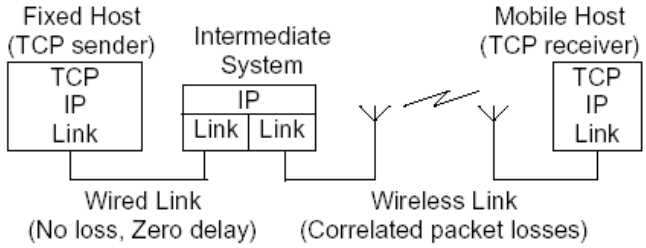

Figure 5. Mobile host connected to fixed host over wireless link.

when compared to wireless losses. We also assume very low bandwidth-delay product and an instantaneous and perfect feedback channel without any errors.

We assume a generic link layer protocol between intermediate system and mobile host. IP fragments the TCP packet into multiple frames depending on the wireless link's MTU and the link layer protocol adds some error control bits before transmitting the packet on the wireless link. Due to the high bit error rate on the wireless link, some of these frames are lost or are received in error by the receiver. Normally, the link layer protocol discards such corrupted frames and the TCP/IP layer does not receive the complete packet. We assume a modified link layer protocol that does not discard the frames received in error. The frames received by the wireless link receiver are simply passed up to the IP layer; a corrupted frame is accompanied by an indication that it was received in error.

We will evaluate the probability that the receiver is able to provide reliable ELN to the sender. Following Zorzi *et al* (1995), we model the wireless link as a discrete time Markov chain with two states: good and bad. Time is slotted in units of frame transmission time (figure 6). In § 4, figure 6 will again be used, with the modification that slot times will represent packet transmission times.

We assume that packet/frame transmission starts only at slot boundaries t_k, t_{k+1}, etc. If the channel is in the good state at time t_k^-, then the transmission starting at time t_k will be successful with probability 1 and when the channel is in the bad state at time t_k^-, then the packet/frame transmission starting at time t_k will fail with probability 1. Also, it is assumed that the channel state changes only at the slot boundary. For such a channel the transition probability matrix is given by

$$M_{CP}(x) = \begin{pmatrix} P_{BB}(x) & P_{BG}(x) \\ P_{GB}(x) & P_{GG}(x) \end{pmatrix} \tag{1}$$

$$M_{CP} = \begin{pmatrix} P_{BB} & P_{BG} \\ P_{GB} & P_{GG} \end{pmatrix}, \quad \text{where} \quad M_{CP}(x) = (M_{CP})^x. \tag{2}$$

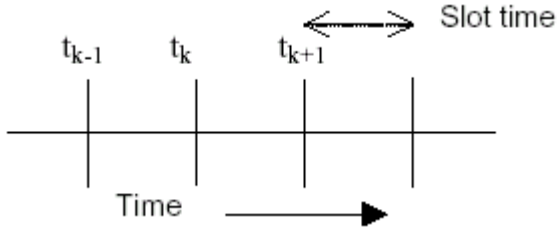

Figure 6. Slotted timeline for packets/ frames over the wireless link.

Here $M_{CP}(x)$ denotes the x-step transition probability matrix and M_{CP} denotes the single-step transition probability matrix. Thus $P_{GG}(x)$ denotes the probability that transmission in slot i is successful given that the transmission in slot $i - x$ was successful. Given the matrix M_{CP}, the channel properties are completely characterized. In particular, it is possible to find the steady state distribution of the chain. The steady state probability that a packet error occurs, P_E, is

$$P_E = \frac{1 - P_{GG}}{2 - P_{GG} - P_{BB}}. \tag{3}$$

Also, for a Rayleigh fading channel with a fading margin F, the average packet error rate can be found as

$$P\_E = 1 - \exp(-1/F). \tag{4}$$

The Markov parameter P_{BB} is given as

$$P_{BB} = 1 - \left(\frac{Q(\theta, \rho\theta) - Q(\rho\theta, \theta)}{e^{1/F} - 1} \right), \text{ where } \quad \theta = \sqrt{\frac{2/F}{1 - \rho^2}}$$

$$\text{and} \quad \rho = J_0 (2\pi f_d T) \tag{5}$$

ρ is the Gaussian correlation coefficient of two successive samples of the complex amplitude of a fading channel with Doppler frequency f_d, taken T seconds apart. $f_d T$ is the normalized Doppler bandwidth and is the indication of correlation in the wireless channel. Lower value of $f_d T (= 0{\cdot}01)$ indicates high correlation in the channel and larger value $(= 0{\cdot}5)$ indicates low correlation approaching (independent identically distributed) IID characteristics for the channel. Table 1 indicates the calculated values of $f_d T$ for various values of mobile speed and transmission rate. For the numerical calculations, we assume a TCP packet size of 1400 bytes and carrier frequency of 900 MHz. T is the packet transmission time in seconds on the channel and is calculated as

$$T = \frac{K_P}{\text{Transmission rate in bits/sec}}, \tag{6}$$

Table 1. Normalized Doppler bandwidth f_dT for various values of mobile speed and transmission rate on wireless link.

S. No.	Mobile Speed (Km/hr)	Transmission rate on wireless link	$f_d T$
1.		100 Kbps	0·168
2.	1·8	1 Mbps	0·0168
3.		2 Mbps	0·0084
4.		100 Kbps	3·36
5.	36	1 Mbps	0·336
6.		2 Mbps	0·168
7.		100 Kbps	8·4
8.	90	1 Mbps	0·84
9.		2 Mbps	0·42

where K_P is the packet size in bits. $J_0(.)$ is the Bessel function of the first kind and zero order. $Q(.,.)$ is the Marcum Q function given by

$$Q(x, y) = \int_y^\infty e^{\frac{-(x^2+w^2)}{2}} I_0(xw)wdw. \tag{7}$$

$I_0(.)$ is the modified Bessel function of first kind and zero order. Now we can calculate the channel parameters for transmission of TCP packets of different sizes and for different values of packet error probability, speed of mobile and transmission rate. We can easily adapt the same approach to frames. The slot time is defined as the frame transmission time in this case. Here, we can calculate the Markov channel parameters for a given F_E (frame error probability), mobile speed, transmission rate and size of frame K_F. Here

$$K_F = K_P/N. \tag{8}$$

when a packet of size K_P is fragmented into N (fragmentation factor) frames, each of size K_F. We can denote the transition probability matrix for frames as

$$M_{CF}(x) = \begin{pmatrix} F_{BB}(x) & F_{BG}(x) \\ F_{GB}(x) & F_{GG}(x) \end{pmatrix}. \tag{9}$$

The steady state probability that a frame error occurs is given by

$$F_E = \frac{1 - F_{GG}}{2 - F_{GG} - F_{BB}}. \tag{10}$$

We will now calculate packet error probability given the transition probability matrix for frames. A packet will be received in error if any of the frames is received in error. This can be written as

$$P_E = 1 - [(1 - F_E)(F_{GG}^N) + F_E F_{BG} F_{GG}^{(N-1)}]. \tag{11}$$

Here we are first calculating the probability that all the N frames of the packet are received without error, when the previous frame was received in either good or bad condition, and then subtracting it from one.

Normally, IP will reassemble packets based on start of packet indication in the first fragment and end of packet indication in the last fragment. In case any of these fragments is lost, the IP layer at the receiver will not be able to identify the boundaries of the packet from the fragments. Thus, in our analysis we assume that the receiver will be able to correctly decode header information only when both the first and the last frame of the packet are received correctly. If the first and/or last frame of the packet is found in error, we will conclude that the header is found in error. We will denote this term by header error probability H_E. This is given by

$$\begin{aligned} H_E = {} & ((1 - F_E)F_{GB} + F_E F_{BB}) \\ & +((1 - F_E)F_{GG} + F_E F_{BG})(F_{GB}(N - 1)). \end{aligned} \tag{12}$$

In the above expression the first term denotes the case where the first frame is received in error. The second term denotes the case where the first frame is received successfully, but the last frame is received in error. Let us denote the probability that the receiver will send

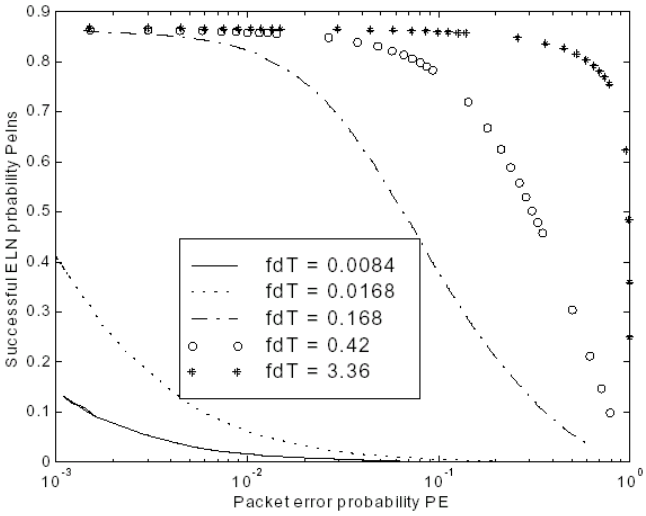

Figure 7. Successful explicit loss notification probability versus packet error probability, for fragmentation factor $N = 15$ and different values of normalized Doppler bandwidth f_dT.

successful ELN information to the sender, given that a packet is received in error, by P_{ELNS}. Also P_{ELNF} denotes the probability that the receiver will not send ELN information to the sender. Now P_{ELNF} is just the probability that first or last frame is received in error given that packet was received in error. This is given by

$$P_{ELNF} = H_E/P_E \quad \text{and} \quad P_{ELNS} = 1 - H_E/P_E. \tag{13}$$

Using the expressions we have derived above, we can now calculate the probability that the sender will receive ELN information from the receiver, given the packet error rate. In figure 7, we have plotted P_{ELNS} versus P_E for various values of f_dT. It is obvious from figure 7 that there is high probability that the receiver will be able to distinguish corruption loss from congestion loss when the channel is not highly correlated (for high values of f_dT). In case the channel is highly correlated (for low values of f_dT), then it is likely that either the first or last frame of the packet will be in error and thus the receiver will not be able to differentiate corruption loss from congestion loss.

4. Analytical model for performance evaluation of NewRenoEln

In this section, we will develop the analytical model for performance evaluation of NewRenoEln protocol. Using this model we can calculate the throughput of data transfer under various channel conditions. We have assumed that a large data file is to be transferred from the TCP sender to a mobile host. We also assume very low bandwidth-delay product and instantaneous and perfect feedback channel without any errors.

4.1 *Joint Window/Channel Evolution*

The analysis for performance evaluation of the TCP protocol is based on a Markov renewal reward approach (Zorzi *et al* 2000). The joint evolution of the window parameters and the channel state can be tracked by a random process $\mathcal{X}(t) = (C(t-1), W_{th}(t), W(t))$, where

$W(t)$ and $W_{th}(t)$ are the window size and slow start threshold in slot t, respectively, and $C(t-1)$ is the channel state (bad, B, or good, G, corresponding to an erroneous or correct transmission, respectively) in slot $t-1$. This process is not a Markov process as its evolution from a certain state not only depends on the parameters specified by $\mathcal{X}(t)$, but also on some other parameters like number of oustanding packets not yet acknowledged and the time at which these packets were transmitted. If we incorporate these parameters into the process description then the state space of the process would become very large and impractical to evaluate. To make the process Markov we will sample it at appropriate instants t_k such that $X(k) := \mathcal{X}(t_k)$ is a Markov process.

Considering t_k's as the slots immediately following those in which either a timeout timer expires or a loss recovery phase is successfully completed, we obtain a process $X(k) = \mathcal{X}(t_k)$ which is Markov. At such instants, there are no outstanding packets and knowledge of $(C(t-1), W_{th}(t), W(t))$ is enough to characterize the window/channel evolution in the future. Also from the protocol rules, the value of $W(t_k)$ can only be equal to one (timeout case) or to $W_{th}(t)$ (successful loss recovery). Note that the channel state at time $t_k - 1$ can be either bad or good in the timeout case, but can only be good for a successful loss recovery, which must be ended by successful transmission. Therefore, the state space of the process $X(k)$ is given by

$$\begin{aligned}\Omega_X = {} & \{(C, W_{th}, 1), C = B, G, 1 \le W_{th} \le \lceil W_{\max}/2 \rceil\} \\ & \cup \{(G, W_{th}, W_{th}), 1 \le W_{th} \le \lceil W_{\max}/2 \rceil\},\end{aligned} \tag{14}$$

where the first set corresponds to timeout and the second set corresponds to successful recovery phase. Note that the total number of states in this case is $3\lceil W_{\max}/2 \rceil - 1$.

For evaluating metrics of interest, such as throughput, we need to track transmission attempts, successful transmissions and time between successive sampling instants. In order to be able to characterize these quantities, we consider a semi-Markov process which admits $X(k)$ as its embedded Markov chain. That is, we label transitions of the chain $X(k)$ with transition metrics, which track the (possibly random) events which determine time delay, transmissions, and successes. For a given transition, let N_d be the associated number of slots, N_t the number of transmissions, and N_s the number of successful transmissions.

4.2 *Semi-Markov Analysis*

Let t_k be the k^{th} sampling instant determined according to the above rules (without loss of generality, assume $t_1 = 1$). We define cycle k as the time evolution of the system between the two consecutive sampling instants t_k and t_{k+1}. The statistical behavior of a cycle only depends on the channel state at time $t_k - 1$ and on the slow start threshold and window size at time t_k. Also, we assume that the process is stationary, so that everything is independent of k. When we look at the evolution of the process during a generic cycle k, it is implicit in what follows that system variables are conditioned on the pair of states $X(k), X(k+1) \in \Omega_X$, where $X(k) = (C(t_k - 1), W_{th}(t_k), W(t_k))$ and Ω_X is the set of all possible values of $X(k)$ (state space of the sampled process). For simplicity of notation, let $C(t_k - 1) = C$.

Let the n^{th} packet (where $n \ge 1$) be the first packet of the cycle that was either erroneously re-transmitted (for which no ELN is sent by the receiver) or erroneously transmitted for which receiver could not generate ELN. Thus in this part of the cycle there will be $n-1$ successes and these will only affect the window size evolution. The failures that get recovered by successful retransmission based on ELN will be counted as single successes and affect

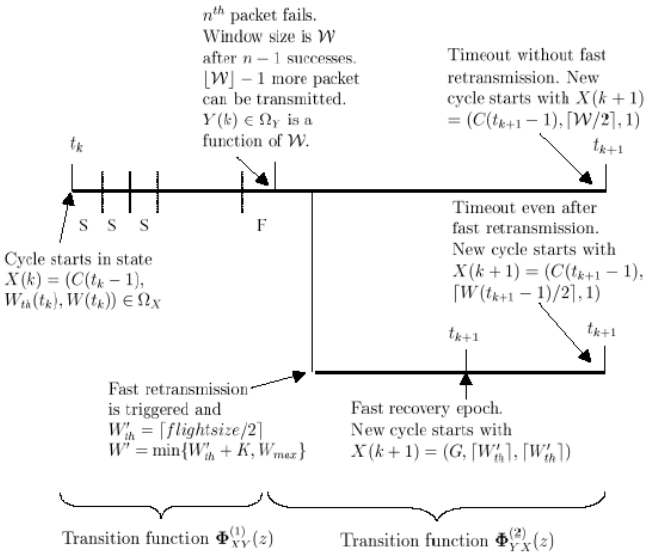

Figure 8. Evolution of NewRenoEln transmission cycle; two parts of the cycle are shown.

window evolution as if one successful transmission had occurred. Let $Y(k)$ be the system state and $\mathcal{W}$ be the congestion window size after $n-1$ successes in cycle k. At this point there are no outstanding packets except for the one transmitted last. Also, by definition we know that the channel state is B. In the next cycle, the value of the slow start threshold will only depend on the congestion window size in the previous cycle. Therefore, the state space Ω_Y only consists of the possible values of the congestion window size $\mathcal{W}$ after $n-1$ successes, which is the only quantity to be tracked. Following the argument in Zorzi *et al* (2000), the size of Ω_Y is $3W_{\max}/2 - 1$ (for $W_{\max}$ even).

The system evolution during a cycle can then be separated into two parts. In figure 8, we have shown the evolution of cycle in two parts. In the first part, the system makes a transition from a state $X(k) \in \Omega_X$ to a state $Y(k) \in \Omega_Y$; whereas in the second part, the system makes a transition from a state $Y(k) \in \Omega_Y$ to a state $X(k+1) \in \Omega_X$. Due to the feed forward structure of the transitions, we can consider the two parts separately and then combine the results to obtain the complete description of a full cycle. More specifically, let $\Phi^{(1)}(z)$ be a matrix whose ik^{th} entry is the transition function associated with the transition from state $i \in \Omega_X$ to state $k \in \Omega_Y$, and analogously let $\Phi^{(2)}(z)$ be a matrix whose kj^{th} entry is the transition function associated with the transition from state $k \in \Omega_Y$ to state $j \in \Omega_X$. The statistics of the system evolution during a cycle is fully characterized by the matrix

$$\Phi(z) = \Phi^{(1)}(z)\Phi^{(2)}(z) \tag{15}$$

whose entries are the transition functions associated with transitions from Ω_X to itself. The variable z is in general a vector of transform variables, each of which tracks a quantity of interest. In our case, we set $z = (z_d, z_t, z_s)$, to track the delay, number of transmissions and number of successes. More precisely, let $\xi_{ij}(N_d, N_t, N_s)$ be the probability that the system makes a transition to state j in exactly N_d slots, and that in $\{1, 2, \ldots, N_d\}$ slots, N_t transmission attempts are performed and N_s transmission successes are counted, given that the

system was in state i at time 0. Then, we have

$$\Phi_{ij}(z_d, z_t, z_s) = \sum_{n_d, n_t, n_s} \xi_{ij}(n_d, n_t, n_s) z_d^{n_d} z_t^{n_t} z_s^{n_s}. \tag{16}$$

In particular, the transition matrix of the embedded Markov chain is just given by

$$\mathbf{P} = \Phi(1, 1, 1). \tag{17}$$

The matrix of average delays can be found as

$$\mathbf{D} = \left. \frac{\partial \Phi(z_d, z_t, z_s)}{\partial z_d} \right|_{z_d, z_t, z_s = 1} = \mathbf{D_1} \Phi^{(2)}(1, 1, 1) + \Phi^{(1)}(1, 1, 1) \mathbf{D_2}, \tag{18}$$

where,

$$\mathbf{D_1} = \left. \frac{\partial \Phi^{(1)}(z_d, z_t, z_s)}{\partial z_d} \right|_{z_d, z_t, z_s = 1} \quad \mathbf{D}_2 = \left. \frac{\partial \Phi^{(2)}(z_d, z_t, z_s)}{\partial z_d} \right|_{z_d, z_t, z_s = 1} \tag{19}$$

The averages of other quantities like number of transmission attempts **T** and number of successes **S** can be similarly found. From the above quantities we can compute a number of steady-state performance parameters. In particular, we can evaluate the average throughput as

$$\text{Throughput} = \frac{\sum_{i \in \Omega_X} \pi_i \sum_{j \in \Omega_X} S_{ij}}{\sum_{i \in \Omega_X} \pi_i \sum_{j \in \Omega_X} D_{ij}}, \tag{20}$$

where π_i, $i \in \Omega_X$, are the steady-state probabilities of the Markov chain with transition matrix **P** and can be computed using the expression $\pi = \pi \mathbf{P}$. Note that, in order to compute steady-state performance from this analysis, knowledge of $\mathbf{P} = \Phi(1, 1, 1)$ and of **D**, **T** and **S** is sufficient. Detailed derivation for throughput is provided in Appendix I.

Now we will evaluate the transition functions $\Phi^{(1)}(z)$ and $\Phi^{(2)}(z)$. The major task here is to correctly identify all possible transitions and the associated transition functions. To derive these, we proceed as follows for each possible system state (transition origin): (i) a set of mutually exclusive events is identified, exhausting all possibilities; (ii) for each of those events, based on the origin state and on the NewRenoEln protocol rules, the destination of the corresponding transition is identified and the transition function is computed; (iii) transitions corresponding to distinct events but leading to the same destination state are combined (i.e., the corresponding transition functions are added), to obtain $\Phi^{(1)}(z)$ and $\Phi^{(2)}(z)$.

4.3 *Computation of* $\Phi^{(1)}(z)$ *for NewRenoEln*

The first part of the cycle consists of either error-free transmissions or successful retransmissions based on ELN information. Let $X = X(k) = (C, W_{th}, W)$. The first part of the cycle has $N_s = n - 1$ successes. Conditioned on the value of n, the window size after $n - 1$ successes is a deterministic function $\omega(n, W, W_{th})$ of W_{th} and W that can be tabulated and is assumed to be known. Therefore, the window size after $n - 1$ successes is denoted by

$$\mathcal{W} = W(n) = \omega(n, W, W_{th}). \tag{21}$$

We will define the transition function $\alpha_C(n)$ based on the value of n and the channel state C at the beginning of the first part of the cycle.

$$\alpha_C(n) = \begin{cases} P_{CB}P_{ELNF}z_d^1 z_t^1 z_s^0 + P_{CB}P_{ELNS}P_{BB}z_d^2 z_t^2 z_s^0 & n = 1 \\ (P_{CG}z_d^1 z_t^1 z_s^1 + P_{CB}P_{ELNS}P_{BG}z_d^2 z_t^2 z_s^1)\alpha_G(n-1) & n > 1. \end{cases} \tag{22}$$

The symbols used in the above equation are defined in previous section. Here $n = 1$ means that the first transmission attempt itself failed. The first term indicates the case where the receiver failed to generate ELN information. The second term indicates the case where the sender received ELN information and attempted re-transmission in the subsequent slot. This re-transmission also failed due to bad channel condition. The number of successes in both cases is 0 but the number of slots used and number of transmissions is 2 in the second case as it includes the retransmission attempt also.

Let us consider $n = 2$ for the case when $n > 1$. In this case there is one success and then a failure. We use recursion to evaluate this case. We have already evaluated the expression for $n = 1$ and this is used again, as indicated here by the term $\alpha_G(n-1)$. The success can be due to two cases. In the first case, the transmission was successful in the first attempt as the channel was good. In this case the number of successes, number of transmission attempts and number of slots all are one. If the first transmission attempt is unsuccessful then with probability P_{ELNS}, the sender will receive ELN information and will attempt re-transmission. This re-transmission will be successful if the channel is good in the next slot. In this case the number of successes is still one, but the number of transmission attempts and number of slots is two. The same explanation can be easily extended to the cases for $n > 2$ also. With this we can now write the expression for $\Phi^{(1)}(z)$ as

$$\Phi_{XY}^1(z_d, z_t, z_s) = \sum_{n \in \mathcal{C}(X,Y)} \alpha_C(n)$$
$$\text{where} \quad \mathcal{C}(X, Y) = \{n : \omega(n, W, W_{th}) = \mathcal{W}\} \tag{23}$$

4.4 *Computation of* $\Phi^{(2)}(z)$ *for NewRenoEln*

The second part of the cycle can also be fully characterized by appropriately labelling transitions and counting events. Let $\varepsilon(Y)$ be an exhaustive set of mutually exclusive events associated with transitions from state $Y(k) = Y$. Also, let $d_Y(.) : \varepsilon(Y) \to \Omega_X$ be a well defined function, giving the destination state corresponding to each event in $\varepsilon(Y)$. Note that this requires that events be defined so that the destination is uniquely specified. However, distinct events may lead to the same destination. Also, let the function Ψ map each event to the corresponding transition function. We can then formally find the YX^{th} entry of the transition function matrix $\Phi^{(2)}(z)$ as

$$\Phi_{YX}^2(z) = \sum_{\mathcal{A} \in \mathcal{D}(X,Y)} \psi(\mathcal{A}), \tag{24}$$

where $\mathcal{D}(X, Y) = \{\mathcal{A} \in \varepsilon(Y) : d_Y(\mathcal{A}) = X\}$ is the set of all events leading from $Y \in \Omega_Y$ to $X \in \Omega_X$ during phase two. We evaluate all such events for the second phase of the cycle below.

For simplicity of notation, in what follows we denote by 0 the slot where the first phase ends, so that the first slot in the second phase corresponds to time 1. Define $\varphi_{ij}(k, x, y)$ as

the probability that there are k successes in slots 1 through y, when the sender has exactly x packets to send and it transmits all of them, and the channel is in state j at y, given that the channel was in state i at time 0. Recursive solution for this is as follows:

$$\varphi_{ij}(k,x,y) = \begin{cases} 0, & \text{for} \quad k,x,y < 0 \\ 0, & \text{for} \quad k > x, \quad k > y, \quad x > y \\ 0, & \text{for} \quad (k = x = y = 0 \quad \text{and} \quad i \neq j) \\ 1, & \text{for} \quad (k = x = y = 0 \quad \text{and} \quad i = j) \\ P_{iG}\varphi_{Gj}(k-1,x-1,y-1) & \\ \quad + P_{iB}P_{ELNS}P_{BG}\varphi_{Gj}(k-1,x-1,y-2) & \\ \quad + P_{iB}P_{ELNF}\varphi_{Bj}(k,x-1,y-1) & \\ \quad + P_{iB}P_{ELNS}P_{BB}\varphi_{Bj}(k,x-1,y-2), & \text{otherwise.} \end{cases} \tag{25}$$

The first term in the recursive expression given above indicates that the first transmission attempt is successful. The second term indicates that the first transmission attempt was unsuccessful but the sender received ELN information and the re-transmission was successful. The third term indicates that the first transmission attempt was unsuccessful and the sender did not get any ELN information. The fourth term indicates that the first transmission attempt was unsuccessful, the sender received ELN information but the re-transmission was unsuccessful.

We will consider two cases: one in which fast re-transmission is not triggered and the other in which fast re-transmission is triggered.

4.4a *Fast re-transmission is not triggered:* In the second part of the cycle the congestion window is denoted by $\mathcal{W}$ and the sender can now send $\{1, 2, \ldots, \lfloor \mathcal{W} \rfloor - 1\}$ more packets. Let $N_{ps} < K$ denote the number of packets that are successfully transmitted. In this case the timeout timer will expire and the lost packet will be re-transmitted in slot $t_{k+1} = T_0$ (Timeout). We assume in following sections that the timeout value is always very large as compared to the congestion window size and the sender will exhaust its window of packets before the timeout timer expires. Note that the value of the window size at timeout will still be equal to $\mathcal{W}$ (recall that duplicate ACKs before fast re-transmission do not advance the window), so that after timeout the algorithm will set $W_{th} = \lceil \mathcal{W}/2 \rceil$, $W = 1$. This event will therefore, lead to state $X(k+1) = (C, \lceil \mathcal{W}/2 \rceil, 1)$ with transition function

$$z_d^{(T_0-1)} \sum_{y=\lfloor \mathcal{W} \rfloor - 1}^{2(\lfloor \mathcal{W} \rfloor - 1)} \sum_{N_{ps}=0}^{K-1} \varphi_{BB}(N_{ps}, \lfloor \mathcal{W} \rfloor - 1, y) P_{BC}(T_0 - y - 1) z_s^{N_s(B,N_{ps})} z_t^y$$

$$+ z_d^{(T_0-1)} \sum_{y=\lfloor \mathcal{W} \rfloor - 1}^{2(\lfloor \mathcal{W} \rfloor - 1)} \sum_{N_{ps}=1}^{K-1} \varphi_{BG}(N_{ps}, \lfloor \mathcal{W} \rfloor - 1, y) P_{GC}(T_0 - y - 1) z_s^{N_s(G,N_{ps})} z_t^y \tag{26}$$

where $0 \leq N_s(B, N_{ps}), N_s(G, N_{ps}) \leq N_{ps}$. The expression here is calculated for $N_{ps} \leq K-1$ successes when the sender has $\lfloor \mathcal{W} \rfloor - 1$ packets to transmit and it uses y slots to transmit them. Clearly y can range from $\lfloor \mathcal{W} \rfloor - 1$ to $2(\lfloor \mathcal{W} \rfloor - 1)$ as each packet transmission will utilize a minimum of one and a maximum of two slots, irrespective of its success or failure.

The two terms account for the two possibilities for the channel state after $\lfloor \mathcal{W} \rfloor - 1$ packets are transmitted. $N_s(B, N_{ps})$ and $N_s(G, N_{ps})$ are random variables and their exact values will depend on further evolution of the chain. This is because we may or may not count certain packets as successes, depending on their possible retransmission in the next cycle. We will only consider bounds for these variables for the best and worst cases as shown above.

4.4b *Fast re-transmission is triggered:* In this case, the sender receives the K^{th} duplicate ACK and fast re-transmit is triggered right after the K^{th} successful transmission in $\{1, 2, \ldots, \lfloor \mathcal{W} \rfloor - 1\}$. Clearly in this case, $\lfloor \mathcal{W} \rfloor > K$. There are many sub-cases possible here and we will consider them one by one.

Single loss before the K^{th} duplicate ACK and re-transmission successful: In this case a single loss occurs in the cycle and it is the loss that resulted in the state transition from state X to state Y. Since there is no other loss, the sender will receive the K^{th} duplicate ACK after K packet transmissions. Upon receiving the K^{th} duplicate ACK, the slow start threshold will be updated to $W'_{\text{th}} = \lceil flightsize/2 \rceil$ and the congestion window size will be updated to $W' = \min\{W'_{\text{th}} + K, W_{\max}\}$. In this particular case, the flightsize is $K + 1$. The lost packet that resulted in the K duplicate ACKs will now be re-transmitted in the next slot. We consider that re-transmission is successful and loss recovery phase is completed and a new cycle starts with the state $X(k+1) = (G, \lceil W'_{th} \rceil, \lceil W'_{th} \rceil)$ and the transition function is given by

$$z_s^{K+1} \sum_{y=K+1}^{2(K+1)} \varphi_{BG}(K+1, K+1, y) z_d^y z_t^y. \tag{27}$$

Single loss before the K^{th} duplicate ACK and re-transmission failure: In this case we consider that the retransmitted packet also fails. The sender can continue to transmit more packets if allowed by the congestion window and flightsize. In this particular case the sender can send more packets only if $W' > K + 1$. We will consider these two cases separately.

$\mathbf{W' \leq K + 1}$:

Here the sender will stall after re-transmission and wait for timeout and the new cycle will start after timeout with the state $X(k+1) = (C, \lceil W'/2 \rceil, 1)$ and the transition function is given by

$$z_s^K \sum_{y=K}^{2K} \varphi_{BG}(K, K, y) \left[\begin{array}{l} P_{GB} P_{ELNF} P_{BC} (T_0 - 1) z_t^{y+1} z_d^{(T_0+y)} \\ \quad + P_{GB} P_{ELNS} P_{BB} P_{BC} (T_0 - 1) z_t^{y+2} z_d^{(T_0+y+1)} \end{array} \right] \tag{28}$$

$\mathbf{W' > K + 1}$:

In this case the sender can send $W' - (K + 1)$ more packets and some of these packets may be successfully received by the receiver, thereby generating more duplicate ACKs. Each of these duplicate ACKs will increment the congestion window by one, thereby allowing one more packet transmission. This will continue until the number of packets in flight equals the congestion window or the congestion window reaches $W_{\max}$. We will assume that the sender

stalls when the congestion window size is M; then the next cycle starts after timeout with state $X(k+1) = (C, \lceil M/2 \rceil, 1)$, $M = W', W'+1, \ldots, W_{\max}$. Let us denote by N_{pt} the number of packets transmitted after the failure of re-transmission and before the sender stalls, waiting for a timeout. Then $N_{pt} = M - (K+1)$. Also let w denote the number of slots used to transmit these N_{pt} packets. Then the transition function when $M < W_{\max}$ is given by

$$\sum_{y=K}^{2K} \varphi_{BG}(K, K, y) \left[\begin{array}{l} P_{GB} P_{ELNF} \displaystyle\sum_{w=N_{pt}}^{2N_{pt}} \left\{ \begin{array}{l} \varphi_{BB}(M - W', N_{pt}, w) \\ \quad \times P_{BC}\,(T_0 - 1 - w) \\ \quad \times z_t^{y+1+w} z_s^{N_s+K} z_d^{(T_0+y)} \end{array} \right\} \\ \\ \quad + P_{GB} P_{ELNS} P_{BB} \displaystyle\sum_{w=N_{pt}}^{2N_{pt}} \left\{ \begin{array}{l} \varphi_{BB}(M - W', N_{pt}, w) \\ \quad \times P_{BC}\,(T_0 - 1 - w) \\ \quad \times z_s^{N_s+K} z_t^{y+2+w} z_d^{(T_0+y+1)} \end{array} \right\} \end{array} \right] \tag{29}$$

Note that the total number of successful transmissions to be counted in this case is $N_s + K$ where K is a constant and N_s is a random variable with range $0 \le N_s \le M - W'$.

The case where $M = W_{\max}$ is more complicated. Here the number of successful packet transmissions on the channel after the re-transmission fails could range from $M - W'$ to N_{pt}. We will approximate the expression in this case taking the maximum value. This approximation hardly introduces any error in the analytical results shown later. We can write the transition function in this case as

$$\sum_{y=K}^{2K} \varphi_{BG}(K, K, y) \left[\begin{array}{l} P_{GB} P_{ELNF} \displaystyle\sum_{w=N_{pt}}^{2N_{pt}} \left\{ \begin{array}{l} \varphi_{BG}(N_{pt}, N_{pt}, w) \\ \quad \times P_{GC}\,(T_0 - 1 - w) \\ \quad \times z_t^{y+1+w} z_s^{N_s+K} z_d^{(T_0+y)} \end{array} \right\} \\ \\ \quad + P_{GB} P_{ELNS} P_{BB} \displaystyle\sum_{w=N_{pt}}^{2N_{pt}} \left\{ \begin{array}{l} \varphi_{BG}(N_{pt}, N_{pt}, w) \\ \quad \times P_{GC}\,(T_0 - 1 - w) \\ \quad \times z_t^{y+2+w} z_s^{N_s+K} z_d^{(T_0+y+1)} \end{array} \right\} \end{array} \right]. \tag{30}$$

In this case too N_s is a random variable with range $0 \le N_s \le N_{pt}$.

Multiple losses before the K^{th} duplicate ACK and successful loss recovery: Before proceeding further we will define $B(N_k, \ell_1, y)$, $K < N_k < \lfloor \mathcal{W} \rfloor$, $0 < \ell_1 \le K$, $N_k \le y \le 2N_k$, to be the event that the K^{th} success occurs at N_k^{th} packet transmission and y^{th} slot and the first loss after the loss in slot 0 occurred at the ℓ_1^{th} packet transmission. (Note that since $N_k > K$, there must be a packet loss before the K^{th} success). The probability of this event is given as

follows:

$$P\left[B\left(N_k, \ell_1, y\right)\right] = \begin{cases} P_{BB}P_{ELNF}\varphi_{BG}\left(K, N_k - 1, y - 1\right) \\ \quad + P_{BB}P_{ELNS}P_{BB}\varphi_{BG}\left(K, N_k - 1, y - 2\right), \\ \quad \text{for} \quad \ell_1 = 1; N_k = K + 1, ..., \lfloor \mathcal{W} \rfloor - 1 \\ \sum\limits_{w=\ell_1}^{2\ell_1} \{\varphi_{BB}\left(\ell_1 - 1, \ell_1, w\right) \\ \quad \times \varphi_{BG}(K - \ell_1 + 1, N_k - \ell_1, y - w)\} \\ \quad \text{for} \quad \ell_1 = 2, \ldots, K; N_k = K + 1, \ldots, \lfloor \mathcal{W} \rfloor - 1. \end{cases} \tag{31}$$

We are considering the case where multiple losses occur but the sender is able to recover from the losses successfully without waiting for timeout. Upon receiving the K^{th} duplicate ACK the slow start threshold will be updated to $W'_{th} = \lceil flightsize/2 \rceil$ and the window size will be updated to $W' = \min\{W'_{th} + K, W_{\max}\}$. In this particular case the flightsize is $N_k + 1$. Now the sender will re-transmit the lost packet. Here we consider that this retransmission succeeds and also the next $N_k - K$ packets are successfully re-transmitted (note that the N_k^{th} packet transmission resulted in the K^{th} duplicate ACK and thus a total of $N_k - K$ more packets need to be transmitted for loss recovery to be complete) and thus loss recovery is complete. In this case the next cycle starts in the state $X(k+1) = (G, \lceil (N_k + 1)/2 \rceil, \lceil (N_k + 1)/2 \rceil)$ and the transition function is given by

$$\sum_{y=N_k}^{2N_k} \sum_{\ell_1=1}^{K} \sum_{w=N_k+1-K}^{2(N_k+1-K)} \left[\begin{array}{l} P\left[B\left(N_k, \ell_1, y\right)\right] \varphi_{GG}\left(N_k + 1 - K, N_k + 1 - K, w\right) \\ \quad \times z_d^{y+w+1} z_t^{y+w+1} z_s^{N_k+1} \end{array} \right] \tag{32}$$

Multiple losses before the K^{th} duplicate ACK and unsuccessful loss recovery: Many sub-cases are possible here depending on the number of successful re-transmissions after the K^{th} duplicate ACK is received. The analysis becomes very complicated if we track each loss and each retransmission attempt. Instead, some approximations can be made without affecting the overall analysis depending on the desired accuracy of the results. Here we have shown the transition function for one case where first re-transmission fails. The sender will stall after re-transmitting the original lost packet and wait for timeout and the new cycle will start after timeout with the state $X(k+1) = (C, \lceil W'/2 \rceil, 1)$ and the transition function is given by

$$z_s^{N_s} \sum_{y=N_k}^{2N_k} \sum_{\ell_1=1}^{K} P\left[B\left(N_k, \ell_1, y\right)\right] \left[\begin{array}{l} P_{GB}P_{ELNF}P_{BC}\left(T_0 - 1\right) z_t^{y+1} z_d^{(T_0+y)} \\ \quad + P_{GB}P_{ELNS}P_{BB}P_{BC}\left(T_0 - 1\right) z_t^{y+2} z_d^{(T_0+y+1)} \end{array} \right] \tag{33}$$

where $\ell_1 - 1 \leq N_s \leq K$.

Similarly, other sub-cases can be evaluated. Using these expressions we can evaluate the throughput of data transfer when the NewRenoEln protocol is used.

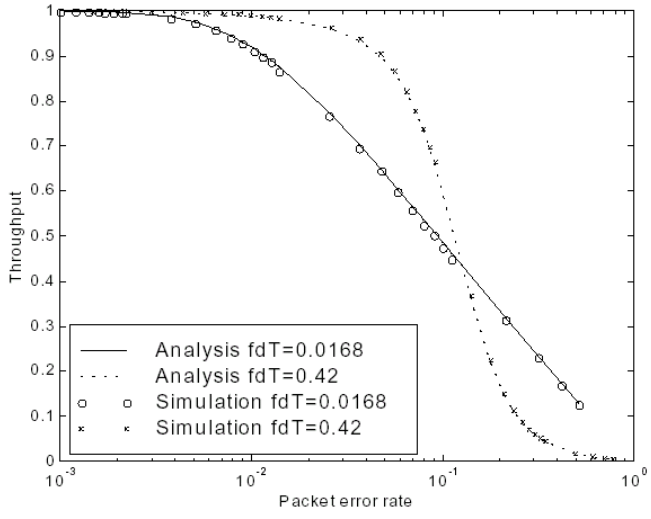

Figure 9. Comparison of analysis and simulation results for NewRenoEln throughput versus packet error rate. $W_{\max} = 6$.

5. Results and discussion

We have done extensive calculations for throughput evaluation under various scenarios that include different channel conditions, mobile host speeds and transmission rate on wireless link. For the throughput evaluation of NewReno we have used the expressions derived in Zorzi *et al* (2000). Some of these results are presented here. In all the graphs we assume dupack threshold $K = 3$, timeout is fixed and is equal to 100 packet transmission times on the wireless link. The TCP packet size is fixed and is 1400 bytes. The sender always has data to send, and the user at the mobile host receives the data as soon as the TCP receiver provides it. We also assume that the sender receives ACKs instantaneously and without any loss.

Figures 9 and 10 present the comparison of analysis and simulation results for the NewRenoEln protocol for various channel conditions and window sizes. Simulation results are obtained by modelling the NewRenoEln protocol in Berkeley Network simulator ns UCB/LBNL/VINT. The simulation results match the results obtained by the analysis very well, showing the accuracy of the analytical approach.

If the channel is highly correlated ($f_d T = 0{\cdot}0168$), then a given packet error rate translates into small number of error events (bad channel condition) but with each error event persisting for a long time. On the other hand, if the channel is IID ($f_d T = 0{\cdot}42$), then the same packet error rate translates into larger number of error events but with each error event persisting for a smaller time. For low values of packet error rate, throughput of NewRenoEln is higher when channel is IID, but at higher values of packet error rate throughput is higher for correlated channel (see figure 9). This shows that NewRenoEln is able to recover from IID packet losses more reliably as compared to correlated losses. However, when the packet error rate is high, the number of error events has a dominant effect on the throughput. This results in lower throughput for IID channel at higher packet rate.

We have also compared the performance of NewRenoEln with NewReno protocol in figures 11–12 for various conditions. For very high correlation in the channel ($f_d T = 0{\cdot}0168$), NewRenoEln performs marginally better than NewReno protocol (figure 11). With high correlation in the channel there is high probability that the header portion of the packet will be

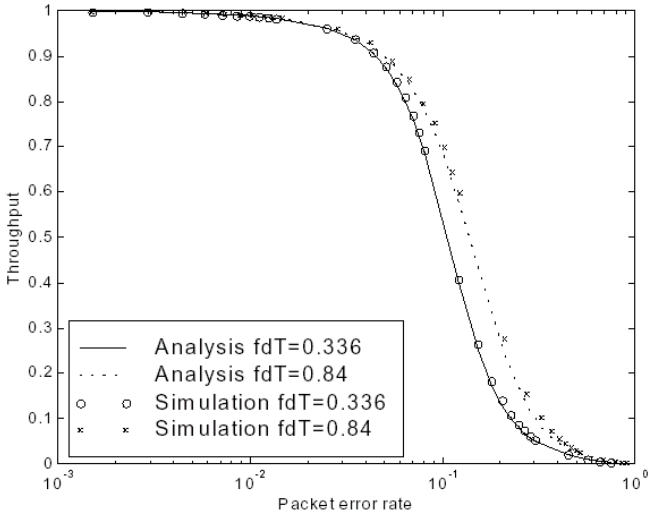

Figure 10. Comparison of analysis and simulation results for NewRenoEln throughput versus packet error rate. $W_{\max} = 24$.

corrupted and the receiver will not be able to distinguish between corruption and congestion loss. When the correlation in the channel is low ($f_d T = 0{\cdot}84$), NewRenoEln performs substantially better than NewReno as is visible in figure 11.

In figure 12, we have shown the effect of the receiver's advertised window size $W_{\max}$ on throughput. The performance of the NewReno protocol improves with increase in window size, as NewReno will be able to recover from losses more often, without going in for coarse timeout. The performance of NewRenoEln, in addition to being better than that of NewReno, is not affected by window size. Even with smaller window size NewRenoEln is able to recover from losses without going in for coarse timeout.

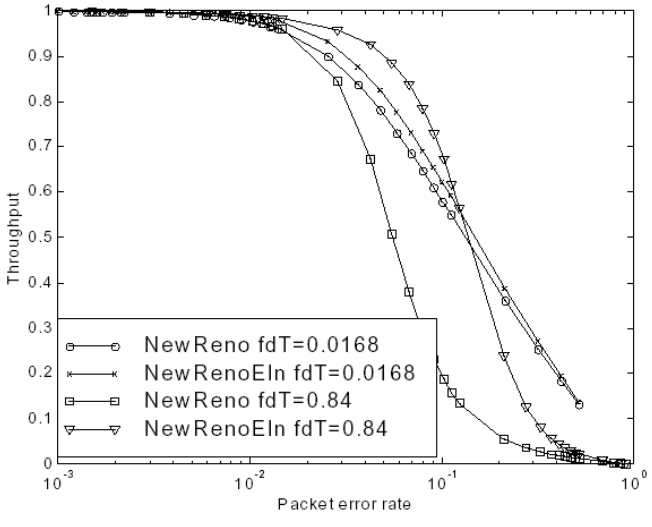

Figure 11. Throughput performance of TCP NewReno and NewRenoEln versus packet error rate. $W_{\max} = 12$.

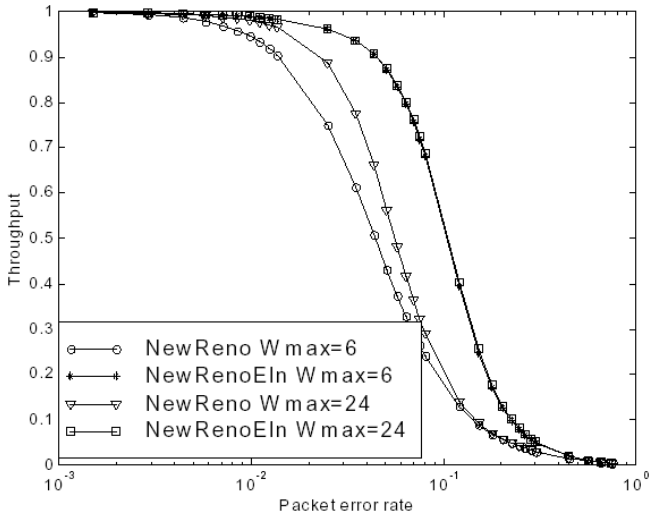

Figure 12. Throughput performance of TCP NewReno and NewRenoEln versus packet error rate. $f_d T = 0{\cdot}336$.

6. Conclusions and future work

We suggested a new protocol called NewRenoEln (NewReno with Explicit Loss Notification) to handle corruption losses in wireless networks. Considering a generic link layer protocol, we have shown that the probability of successful detection of corruption losses can be very high in NewRenoEln, when channel conditions are appropriate. We also developed an analytical model for performance evaluation of the TCP NewRenoEln protocol. We have shown that the results obtained by simulation match and the results obtained by the analytical model, thus showing the accuracy of the analytical approach. We have also shown that the NewRenoEln protocol, performs better than NewReno under various conditions.

There are various possibilities for future work. We need to extend the analysis for delay in the network. We also need to observe and compare the analytical results with the behaviour of the NewRenoEln protocol in the actual network environment.

Appendix I

We will calculate expected cycle length for two different parts of the cycle. We assume that k is an intermediate state after first part of the cycle. For first part of the cycle

$$\xi_{ik}^{1}(\ell, m, n) = P\{N_d = \ell, N_t = m, N_s = n, Y = k/X = i\}$$
$$= P\{N_d = \ell, N_t = m, N_s = n/X = i, Y = k\} P\{Y = k/X = i\} \tag{34}$$

$$\Phi_{ik}^{(1)}(z_d, z_t, z_s) = \sum_{\ell} \sum_{m} \sum_{n} z_d^{\ell} z_t^{m} z_s^{n} \xi_{ik}^{1}(\ell, m, n) \tag{35}$$

and for second part of the cycle

$$\begin{aligned}\xi^2_{kj}(\ell, m, n) &= P\{N_d = \ell, N_t = m, N_s = n, X_f = j/Y = k\} \\ &= P\{N_d = \ell, N_t = m, N_s = n/Y = k, X_f = j\}P\{X_f = j/Y = k\}\end{aligned} \tag{36}$$

X_f is the starting state belonging to Ω_X in the next cycle.

$$\Phi^{(2)}_{kj}(z_d, z_t, z_s) = \sum_{\ell}\sum_{m}\sum_{n} z_d^{\ell} z_t^{m} z_s^{n} \xi^2_{kj}(\ell, m, n) \tag{37}$$

Considering a single element from the matrix of average delays

$$\begin{aligned}D_1[i,k] &:= \left.\frac{\partial \Phi^{(1)}_{ik}(z_d, z_t, z_s)}{\partial z_d}\right|_{z_d, z_t, z_s = 1} \\ &= \sum_{\ell}\sum_{m}\sum_{n} \ell.\xi^1_{ik}(\ell, m, n) \\ &= \sum_{\ell}\sum_{m}\sum_{n} \ell P\{N_d = \ell, N_t = m, N_s = n, /X = i, Y = k\}P\{Y = k/X = i\} \\ &= \sum_{\ell} \ell P\{N_d = \ell/X = i, Y = k\}\, P\{Y = k/X = i\} \\ &= E\{\text{Delay in reaching state k } \in \Omega_Y/X = i, Y = k\}\, P\{Y = k/X = i\} \\ &= E\left\{T^{(1)}/X = i, Y = k\right\} P\{Y = k/X = i\}\end{aligned} \tag{38}$$

Similarly,

$$\begin{aligned}D_2[k, j] &:= \left.\frac{\partial \Phi^{(2)}_{kj}(z_d, z_t, z_s)}{\partial z_d}\right|_{z_d, z_t, z_s = 1} = \sum_{\ell}\sum_{m}\sum_{n} \ell.\xi^2_{kj}(\ell, m, n) \\ &= E\left\{T^{(2)}/Y = k, X_f = j\right\} P\left\{X_f = j/Y = k\right\}\end{aligned} \tag{39}$$

$T^{(1)}, T^{(2)}$ are the cycle lengths for the first and second part of the cycle respectively. Expected delay for the complete cycle for the given initial and final state is obtained as follows. From the definitions of $D_1, \phi^{(2)}, D_2$ and $\phi^{(1)}$, we have

$$\begin{aligned}&(D_1\phi^{(2)}(1,1,1))[i, j] \\ &= \sum_{k} E\{T^{(1)}/X = i, Y = k\}P\{Y = k/X = i\}\, P\{X_f = j/Y = k\} \\ &= \sum_{k} E\{T^{(1)}/X = i, Y = k\}P\{Y = k/X = i\}\, P\{X_f = j/Y = k, X = i\} \\ &= \sum_{k} E\{T^{(1)}/X = i, Y = k\}P\{X_f = j, Y = k/X = i\}\end{aligned} \tag{40}$$

The second step is valid in the equation above as the process is Markov and $P\{X_f = j/Y = k\}$ does not change with inclusion of $X = i$ in the expression. The probability of the next state X_f being j is completely independent of initial state $X = i$, if the intermediate state $Y = k$ is specified.

$$\begin{aligned}
&\sum_j \left(D_1\phi^{(2)}\,(1,1,1)\right)[i,j] \\
&\quad = \sum_j \sum_k E\{T^{(1)}/X = i, Y = k\} P\{X_f = j, Y = k/X = i\} \\
&\quad = \sum_k \sum_j E\{T^{(1)}/X = i, Y = k\} P\{X_f = j, Y = k/X = i\} \\
&\quad = \sum_k E\{T^{(1)}/X = i, Y = k\} P\{Y = k/X = i\}
\end{aligned} \tag{41}$$

The interchange of order of summation in the above equation is valid as the summation is done over finite state space.

$$\begin{aligned}
&\sum_i \pi_i \sum_j \left(D_1\phi^{(2)}\,(1,1,1)\right)[i,j] \\
&\quad = \sum_i P\{X = i\} \sum_k E\{T^{(1)}/X = i, Y = k\} P\{Y = k/X = i\} \\
&\quad = \sum_i \sum_k E\{T^{(1)}/X = i, Y = k\} P\{X = i, Y = k\} = E\{T^{(1)}\}
\end{aligned} \tag{42}$$

Similarly,

$$\sum_i \pi_i \sum_j \left(\phi^{(1)}\,(1,1,1)\,D_2\right)[i,j] = E\{T^{(2)}\}. \tag{43}$$

Thus, using equation (20), the expected cycle length is given by

$$\begin{aligned}
&\sum_i \pi_i \left(\sum_j D\,[i,j]\right) \\
&\quad = \sum_i \pi_i \left\{\sum_j \left(D_1\phi^{(2)}\,(1,1,1)\right)[i,j] + \sum_j \left(\phi^{(1)}\,(1,1,1)\,D_2\right)[i,j]\right\} \\
&\quad = E\{T^{(1)}\} + E\{T^{(2)}\} = E\{T^{(1)} + T^{(2)}\}.
\end{aligned} \tag{44}$$

In a similar manner, we can evaluate the expected number of successes and write the expression for throughput as

$$\text{Throughput} = \frac{\text{Expected number of successes}}{\text{Expected cycle length}} = \frac{\sum\limits_{i\in\Omega_X} \pi_i \sum\limits_{j\in\Omega_X} S_{ij}}{\sum\limits_{i\in\Omega_X} \pi_i \sum\limits_{j\in\Omega_X} D_{ij}} \tag{45}$$

The authors would like to thank Mr Satish Kulkarni and many others from C-DOT and CEDT for their support in this work.

References

Bakshi B S, Krishna P, Vaidya N H, Pradhan D K 1997 Improving performance of TCP over wireless networks. In *Proc. 17th Int. Conf. Distributed Computing Systems*

Balakrishnan H, Padmanabhan V N, Seshan S, Katz R H 1997 A comparison of mechanisms for improving TCP performance over wireless links. *ACM/IEEE Trans. Networking* 56: pp 756–759

Balakrishnan H, Katz R H 1998 Explicit Loss Notification and Wireless Web Performance. In *Proc. IEEE Globecom Internet Mini-Conference*

Balan R K, Lee B P, Kumar K R R, Jacob L, Seah W K G, Ananda A L 2001 TCP HACK: TCP header checksum option to improve performance over lossy links. In *Proc. IEEE INFOCOMM*

Caceres R, Iftode L 1995 Improving the performance of reliable transport protocols in mobile computing environments. *IEEE J. Select. Areas Commun.* 13: pp 850–857

Chen W, Lee J 2000 Some mechanisms to improve TCP/IP performance over wireless and mobile computing environment. In *Proc. 7th Int. Conf. Parallel and Distributed Systems*

Floyd S, Henderson T The NewReno Modification to TCP's Fast Recovery Algorithm, RFC2582

Goff T, Moronski J, Phatak D S, Gupta V 2000 Freeze TCP a true end to end TCP enhancement mechanism for mobile environments. In *Proc. IEEE INFOCOMM*

Gupta Pawan Kumar, Joy Kuri 2002 Reliable ELN to enhance throughput of TCP over wireless links via TCP header checksum. In *Proc. IEEE GLOBECOM*

Kumar A 1998 Comparative performance analysis of versions of TCP in a local network with a lossy link. *ACM/IEEE Trans. Networking* pp 485–498 1998

Parsa C, Aceves J J G L 2000 Differentiating congestion vs random loss: A method for improving TCP performance over wireless links. In *Proc. IEEE WCNC*

Peng F, Ma J A proposal to apply ECN into Wireless and Mobile Networks, draft-fpeng-ecn-03.txt

Postel J Transmission Control Protocol. RFC793

Sinha P, Venkitaraman N, Sivakumar R, Bharghavan V 1999 WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks. In *Proc. ACM MOBICOMM'99*

Stevens W R 1994 *TCP/IP Illustrated*. 1: Reading, MA: Addison Wesley

UCB/LBNL/VINT Network Simulator - ns (version 2). URL: http://www.isi.edu/nsnam/ns

Vaidya N 1997 Discriminating congestion losses from wireless losses using inter arrival times at the receiver. In *Proc. IEEE ASSET*

Zorzi M, Rao R R, Milstein L B 1995 On the accuracy of a first-order Markov model for data transmission on fading channels. In *Proc. IEEE ICUPC'95* pp 211–215

Zorzi M, Chockalingam A, Rao R R 2000 Throughput analysis of TCP on channels with memory. *IEEE J. Select. Areas Commun.* 18: pp 1289–1300