

# USING ECN MARKS TO IMPROVE TCP PERFORMANCE OVER LOSSY LINKS

Haowei Bai

*Honeywell Labs*

*3660 Technology Drive, Minneapolis, MN 55418, USA*

*Email: haowei.bai@honeywell.com*

Mohammed Atiquzzaman

*School of Computer Science*

*University of Oklahoma, Norman, OK 73019-6151, USA*

*Email: atiq@ou.edu*

David Lilja

*Department of Electrical and Computer Engineering*

*University of Minnesota, 200 Union St. SE, Minneapolis, MN 55455, USA*

*Email: lilja@ece.umn.edu*

Keywords: Wireless network, explicit congestion notification, TCP/IP, congestion control

Abstract: TCP was designed for wireline networks, where loss events are mostly caused by network congestion. The congestion control mechanism of current TCP uses loss events as the indicator of congestion, and reduces its congestion window size. However, when a lossy link is involved in a TCP connection, non-congestion random losses should also be considered. The congestion window size should not be decreased if a loss event is caused by link corruptions. To improve TCP performance over lossy links, in this paper, we first present that zero congestion loss could be achieved by appropriately setting the ECN marking threshold in the RED buffer. Secondly, we propose a new TCP algorithm, called Differentiation Capable TCP (Diff-C-TCP). Diff-C-TCP makes an assumption that packet losses are caused by link corruptions, and uses ECN (Explicit Congestion Notification) to determine any loss that may occasionally happen due to network congestion. We have shown that Diff-C-TCP performs very well in the presence of a lossy link.

## 1 INTRODUCTION

The TCP/IP (Transmission Control Protocol/Internet Protocol) protocol was originally designed for wireline networks. Wireless links have a number of fundamentally different characteristics from wired links. Wireless links are dominated by low bandwidth and high error rates, which degrade the performance of TCP (Bai and Atiquzzaman, 2003). In the current TCP, the congestion control mechanism uses packet loss as the indicator of congestion, and reduces its congestion window size. However, when the packet loss is caused by link errors rather than congestion, TCP's reduction of the congestion window size is unnecessary and degrades its throughput.

In most cases, packet losses due to corruption are more significant than congestion losses when a lossy link is involved in a TCP connection. In such a case, TCP may not be able to transmit or receive at the full available bandwidth, because the TCP algorithm is

unnecessarily wasting time in slow-start or congestion avoidance procedures triggered by link errors. Consequently, the current congestion control algorithms in TCP result in very poor performance over lossy links. Significant performance improvements can be achieved (Samaraweera and Fairhurst, 1998) if losses due to network congestion and corruption in lossy wireless links could be appropriately differentiated.

Researchers have applied heuristics, such as loss predictors, to distinguish between congestion and errors due to link errors (Biaz and Vaidya, 1997; Biaz and Vaidya, 1998b), but their simulation results indicated that their loss predictors did not perform well. They proposed a modified TCP-Reno scheme called TCP-Aware (Biaz and Vaidya, 1998a) to differentiate between packet losses due to congestion and losses due to link errors. However, their scheme works well only when the last hop for the connection is wireless, the bandwidth of the wireless link is much smaller than the bandwidth of the wired link, and the overall packet loss rate is small (Biaz and Vaidya, 1998a), all

of which are too limited in the real world Internet.

Without any other additional information, the existing implicit loss feedback mechanisms in TCP does not allow distinguishing between congestion and corruption losses (Dawkins et al., 2000). ECN (Explicit Congestion Notification) (Floyd, 1994) was proposed as an explicit indicator of congestion. It can be used to quickly and unambiguously inform sources of network congestion, without the sources having to wait for either a retransmit timer timeout or three duplicate ACKs (Acknowledgements) to infer a lost packet. For bulk-data connections, this mechanism can avoid unnecessary packet drops for low-bandwidth delay-sensitive TCP connections, and can avoid some unnecessary retransmit timeouts in TCP (Ramakrishnan and Floyd, 1999). Since ECN is an explicit indicator of network congestion, it provides the possibility to differentiate two types of losses. If the buffer in a router is optimally dimensioned and the RED (Random Early Detection) threshold in the router buffer is appropriately set, zero congestion loss could be achieved by appropriately adjusting the source's congestion window size based on feedback from ECN signals. Previous researchers (Kunniyur and Srikant, 2000; Liu and Jain, 2001a; Abouzeid and Roy, 2000) have done some initial work in this direction as described below.

Authors in (Kunniyur and Srikant, 2000) presented a framework for designing end-to-end congestion control schemes in a network where each user may have a different utility function. They considered ECN marks as an alternative to losses for congestion notification. Using this model, they showed that the ECN marking level can be designed to nearly eliminate congestion losses in the network by choosing the marking level independently for each node in the network. However, the drawback of their work is that they achieved zero congestion loss by over-provisioning the network.

Authors in (Liu and Jain, 2001a) analyzed the queue dynamics at the congested router, and derived the closed-form formula and buffer requirements to achieve zero loss and full link utilization. However, as they stated in their work, they did not get the mathematical expression for the *average share of bottleneck link bandwidth* which is the most important parameter of their model. Instead, they used simulation to illustrate the relationship between the average share of bottleneck link bandwidth and the Round Trip Time (RTT). The same difficulty was also encountered by authors in (Abouzeid and Roy, 2000). As a solution, they introduced an unknown constant into the final expression.

Motivated by (Liu and Jain, 2001a; Abouzeid and Roy, 2000), we *derive the exact mathematical model for the average share of bottleneck link bandwidth* by modeling the ECN marking dynamics as a Pois-

son Process. We finally end up with a comprehensive mathematical model for achieving zero congestion loss. The objective of developing the exact mathematical model was that, if we can eliminate all network congestion losses in a heterogeneous network environment involving lossy links, or if the congestion losses are a small fraction of losses due to link error, with negligible error all losses can be attributed to random losses due to link errors. This observation leads to our proposed Diff-C-TCP which is discussed in detail in Section 3.2.

Diff-C-TCP assumes loss events indicate link corruption and uses ECN as congestion indication with the precondition of zero congestion loss to differentiate between congestion and corruption. As mentioned earlier, because of link errors, packet losses due to corruption is more significant in a lossy network. To have a high TCP throughput when a TCP connection traverses a lossy link, the TCP source should persist in the previous utilization of bandwidth instead of reducing the transmission rate when the loss is due to corruption.

The *contributions* of this paper are as follows:

- We develop a comprehensive mathematical model to calculate the value of the average share of bottleneck link bandwidth. The model ensures zero congestion loss in the network.
- Based on the possibility of zero congestion loss, we Propose and evaluate a new TCP algorithm called Diff-C-TCP to improve the performance of TCP over lossy links.

The rest of this paper is organized as follows. In Section 2, we present a comprehensive model to achieve zero congestion loss with multiple competing TCP flows; the model is the basis of our proposed Diff-C-TCP algorithm discussed in Section 3. In Section 4, we describe the simulation methodology that has been used to evaluate the performance of our proposed algorithm. Performance improvements achieved by our proposed algorithm as compared to current TCP are presented in Section 5. Concluding remarks are finally given in Section 6.

## 2 Eliminating Congestion Losses with ECN

If the buffer in the router is optimally dimensioned RED threshold is appropriately set, zero-loss congestion control could be achieved by appropriately adjusting the source's congestion window size based on the notification by ECN.

In this section, we first describe the ECN mechanism and provide the requirements for zero congestion loss in two cases (Liu and Jain, 2001a), viz, one

TCP connection and multiple competing TCP connections, in Sections 2.2 and 2.3 respectively. The difficulty in using the zero congestion model is the calculation of the average share of bottleneck link bandwidth. In this section, we have solved that problem by deriving the exact mathematical expression for the average share of bottleneck link bandwidth, the most important parameter in zero congestion loss model, by modeling the ECN marking dynamics as a Poisson Process in Section 2.4. We finally end up with a comprehensive mathematical model for achieving the zero-loss TCP congestion control.

## 2.1 Analysis Assumptions

In order to set up a proper but not complicated model and analyze the queue dynamics, as well as zero-loss requirements, we make the following assumptions (as used by authors in (Liu and Jain, 2001a; Liu and Jain, 2001b) for a similar model):

- Senders always have data to send and will send as many as their windows allow.
- Receiver windows are large enough.
- There are no delayed acknowledgements.
- All packets have the same length.

## 2.2 Queue Dynamics Analysis with One TCP Flow

Along the path with only one TCP connection, if the bottleneck link bandwidth is  $\mu$  packet/second, then the downstream packet inter-arrival time and the acknowledgement inter-arrival time on the reserve link must be greater than or equal to  $\frac{1}{\mu}$  (Liu and Jain, 2001a).

Based on this statement, we use the analytical model shown in Figure 1 and notations as follows:

$w(t)$  = The sender's window size at time  $t$ .

$r$  = Round Trip Time (RTT).

$\mu$  = Bandwidth of the bottleneck link.

$t_s$  = The time a packet needs to traverse from the sender to the router.

$T$  = The threshold of RED router.

$\hat{T}$  = The optimum threshold of RED router for the purpose of zero congestion loss.

$P$  = The packet which increases the queue length over  $T$ .

From (Liu and Jain, 2001a), we know that if at time  $t$  the bottleneck link has been busy for at least  $r$  seconds, and a packet just arrives at the congested router, the queue length at the congested router is

$$Q(t) = w(t - t_s) - r\mu. \quad (1)$$

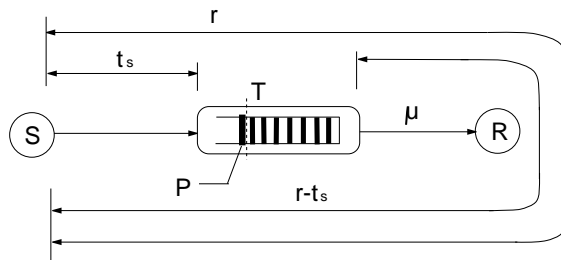


Figure 1: Analytical model of one TCP connection.

Based on this theorem, requirements for achieving zero congestion loss could be described by the relationship between the queue length and the threshold of RED router. Results reported by authors in (Liu and Jain, 2001a) for the two different phases, slow start and congestion avoidance, are given below:

### 2.2.1 Slow Start Phase

The maximum queue length in slow start phase is  $2T + r\mu + 1$ .

### 2.2.2 Congestion Avoidance Phase

The maximum queue length in the congestion avoidance phase is  $T + 1$ ; the minimum queue length is  $\frac{T - r\mu + 1}{2}$ . To derive the optimal threshold for the RED queue, the bottleneck link should be fully utilized, i.e.,

$$\frac{T - r\mu + 1}{2} \geq 0, \quad (2)$$

which means

$$T \geq r\mu - 1. \quad (3)$$

However, if  $T > r\mu - 1$  indicating that the minimum queue length is a positive number, the RED router may have long queueing delays. Thus, the optimal value of threshold ( $\hat{T}$ ) for achieving zero congestion loss, full link utilization and minimum queueing delay in congestion avoidance phase can be calculated from

$$\frac{T - r\mu + 1}{2} = 0. \quad (4)$$

In other words, the optimal value of threshold is

$$\hat{T} = r\mu - 1. \quad (5)$$

Finally, we end up with various cases of possible threshold as shown in Table 1.

## 2.3 Queue Dynamics Analysis with Multiple competing TCP flows

Figure 2 shows the system model considered in this section. We use the following notations in the analysis:

Table 1: Value of threshold  $T$  for the case of one TCP flow.

$T$	Link Status
$T > r\mu - 1$	The link is <i>fully utilized</i> , but packets suffer larger queueing delay.
$\hat{T} = r\mu - 1$	<i>Optimal</i> threshold; zero congestion loss, full link utilization and minimum queueing delay.
$T < r\mu - 1$	The link is <i>under-utilized</i> .

$w_i(t)$  = The  $i^{\text{th}}$  sender's window size at time  $t$ .  
 $r_i$  = Round Trip Time (RTT).  
 $\bar{\mu}_i$  = The average share of the bottleneck link bandwidth.  
 $t_{s_i}$  = The time a packet needs to traverse from the  $i^{\text{th}}$  sender to the router.  
 $T$  = The threshold of RED router.

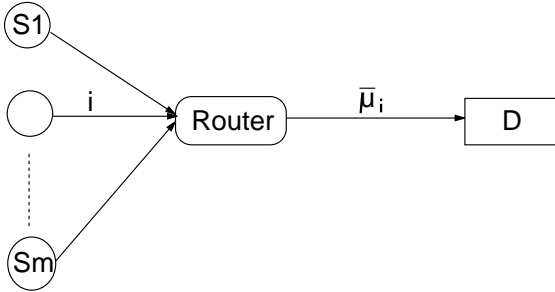


Figure 2: Analytical model of multiple competing TCP connections.

The analysis of the case of multiple competing TCP connections is based on the discussion in Section 2.2. For the analytical model shown in Figure 2, the queue length expression given by (Liu and Jain, 2001a) is shown below.

$$Q(t) = \sum_{i=1}^m (w_i(t - t_{s_i}) - r_i \bar{\mu}_i). \quad (6)$$

As we mentioned in Section 1, the difficulty in using Equation (6) is in calculating  $\bar{\mu}_i$ . We show the calculation of  $\bar{\mu}_i$  in Section 2.4.

Similar to the discussion in Section 2.2, we give the requirements for achieving zero congestion loss for two different cases, slow start phase and congestion avoidance phase. These results are also reported by authors in (Liu and Jain, 2001a).

### 2.3.1 Slow Start Phase

It is almost unlikely for multiple TCP connections to send at the same time. If some connections are in slow

Table 2: The value of threshold  $T$  for the case of multiple TCP flows.

$T$	Description
$T > r\mu - 1$	The link is <i>fully utilized</i> , but packets suffer larger queueing delay.
$\mu - m \leq \hat{T} \leq r\mu - 1$	<i>Optimal</i> threshold; zero congestion loss, full link utilization and minimum queueing delay.
$T < r\mu - m$	The link is <i>under-utilized</i> .

start phase, while others are in congestion avoidance phase, the queue length cannot be increased as fast as all connections are in slow start phase. If all connections start simultaneously, this could be considered as one aggregate flow. Accordingly, the maximum queue length is still  $2T + r_i \bar{\mu}_i + 1$ .

### 2.3.2 Congestion Avoidance Phase

The maximum queue length in congestion avoidance phase is  $T + \alpha$ ,  $\alpha \in [1, m]$ ; the minimum queue length is  $\frac{T - r_i \bar{\mu}_i + \alpha}{2}$ ,  $\alpha \in [1, m]$ . Three different threshold values for the case of multiple TCP flows are shown in Table 2.

## 2.4 Calculating the Average Share of Bottleneck Link Bandwidth with Multiple TCP Flows

In Section 2.2 and Section 2.3, we have summarized the results of RED threshold and buffer size for achieving zero congestion loss at a RED gateway as described in (Liu and Jain, 2001a). However, they did not obtain the quantitative value of the average share of bottleneck link bandwidth ( $\bar{\mu}_i$  in Equation (6) which is the most important parameter in the zero congestion loss model.

In this section, we show the calculation of the average share of the bottleneck link bandwidth. We consider the system model shown in Figure 2 and the Reno version of TCP. To keep the consistence, we use all the assumptions we made for the previous model in Section 2.2 and Section 2.3.

Figure 3 shows the window evolution approximation for TCP-Reno sessions with different round trip delays sharing a bottleneck link with a RED gateway. The loss events are represented by 'x' marks. This approximation has been used by many researchers (Abouzeid and Roy, 2000; Abouzeid et al., 2000) for the analytic understanding of the RED performance. Based on Figure 3, we use the following notations in our model.

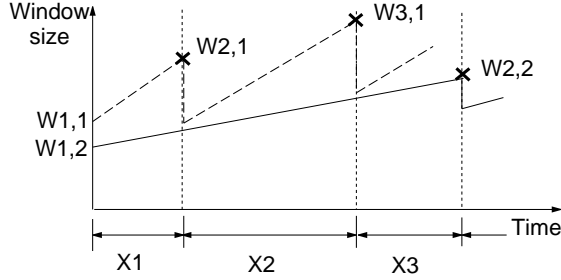


Figure 3: The window evolution approximation with two TCP-Reno flows.

$w_{i,j}(t)$  = The  $j^{\text{th}}$  TCP session's window size right before the previous loss event.

$w_{i+1,j}(t)$  = The  $j^{\text{th}}$  TCP session's window size right before the current loss event.

$\overline{w_{j\text{avg}}(t)}$  = The time-average window size for the  $j^{\text{th}}$  TCP session.

$r_{i,j}$  = Round Trip Time (RTT) of the  $j^{\text{th}}$  TCP session.

$\overline{\mu}_i$  = Average share of the bottleneck link bandwidth.

$L_i$  = The time when  $i^{\text{th}}$  congestion loss event happens.

Consider the scenario in Figure 3;  $X_i = L_i - L_{i-1}$  denotes the inter-loss duration. The window evolution could be expressed by the following equation.

$$w_{i+1,j}(t) = \frac{w_{i,j}(t)}{2} + \frac{X_i}{r_{i,j}}. \quad (7)$$

Since loss events are determined by both the traffic type and the random marking at RED router, it is reasonable to consider  $\{X_i\}$  as an Independent Identical Distributed (i.i.d.) *renewal process*. If we assume in any length of time interval, the number of loss event is Poisson distributed, then the total number of loss events in the interval  $(0, t)$  is a *Poisson process*, denoted by  $N(t)$ . Therefore, the loss time interval  $X_i$  is an i.i.d. *exponential random variable* with the parameter  $\lambda$ . Its probability density function (pdf) is

$$f_{X_i}(t) = \lambda e^{-\lambda t} u(t). \quad (8)$$

In addition, the waiting time  $T[n] = \sum_{k=1}^n X_k$  for a loss event is a *gamma distributed random variable* with parameters  $(n, \lambda)$ . Its pdf can be found as

$$f_T(t) = \frac{\lambda e^{-\lambda t} (\lambda t)^{n-1}}{\Gamma(n)} u(t), \quad (9)$$

which is, in the other form,

$$f_T(t) = \frac{\lambda^n e^{-\lambda t} t^{n-1}}{(n-1)!} u(t). \quad (10)$$

Based on the mathematical nature of the window evolution we analyzed above, we finally calculate the

average share of bottleneck link bandwidth, which has been defined as

$$\overline{\mu}_i = \frac{\overline{w_{j\text{avg}}(t)}}{r_{i,j}}. \quad (11)$$

Taking the expectation for both sides of Equation (7), we have

$$\overline{w_{i+1,j}(t)} = \frac{\overline{w_{i,j}(t)}}{2} + \frac{\overline{X_i}}{r_{i,j}}. \quad (12)$$

Since any two loss events have the same statistical characteristics, it is apparent that  $w_{i+1,j}(t)$  and  $w_{i,j}(t)$  have the same expected value. Thus,

$$\overline{w_j(t)} = 2 \frac{\overline{X_i}}{r_{i,j}}. \quad (13)$$

Recall that the loss time duration is a renewal process and the total number of loss events during any length of time interval is a Poisson process, from Equation (8), we should have

$$\overline{X_i} = E[X_i] = \frac{1}{\lambda}. \quad (14)$$

Therefore,

$$\overline{w_j(t)} = \frac{2}{\lambda r_{i,j}}. \quad (15)$$

Because Poisson process is ergodic in mean (See Appendix A for the proof), using the property of ergodicity, we have

$$\overline{w_{j\text{avg}}(t)} = \overline{w_j(t)} = \frac{2}{\lambda r_{i,j}}. \quad (16)$$

Finally, the average share of the bottleneck link bandwidth is

$$\overline{\mu}_i = \frac{\overline{w_{j\text{avg}}(t)}}{r_{i,j}} = \frac{2}{\lambda r_{i,j}^2}. \quad (17)$$

Remarkably, the above result implies that the connection with the shortest RTT has the largest average share, which is also found by authors in (Liu and Jain, 2001a; Mista et al., 1999).

### 3 Using ECN to Improve the TCP Performance over Lossy Links: Diff-C-TCP

From the previous discussion, we can eliminate all network congestion losses (or the congestion losses are a small fraction of random losses) in a heterogeneous network environment involving lossy links. Therefore, with the negligible error all losses can be attributed to random losses. This leads to our proposed Diff-C-TCP discussed in this section.

### 3.1 Design Assumptions

Before we start to illustrate the principle of our proposed Diff-C-TCP algorithm, we make the following assumptions:

- Our proposed algorithm is used within a WAN or an enterprise network, where it is possible to make all routers and end-systems ECN-capable.
- When we mention wireless links, in order to keep our discussion focused, we do not consider mobility issues such as handoff or power requirements.

### 3.2 The Proposed Diff-C-TCP

Our proposed algorithm assumes that packet losses indicate corruption, and the TCP sender uses ECN as an explicit notification of network congestion. Diff-C-TCP's response to ECN is similar to TCP's response to packet losses. In other words, the receipt of ECN packets should trigger a response to network congestion. Packet losses are treated as link errors unless ECN packets are received.

Figure 4 shows the kernel of our proposed Diff-C-TCP algorithm at the sender's side. A Diff-C-TCP sender treats the situation that the retransmit timer times out without receiving any ECN.ECHO packet and (or) receiving duplicate acknowledgements as the indication of link errors. Most often, this is the case in a network with wireless links (packet losses due to link errors). In this case, the Diff-C-TCP source does not decrease *cwnd*. If the Diff-C-TCP sender receives the ECN.ECHO packet sent by the receiver, the sender treats it as network congestion and triggers the Fast Recovery algorithm (Ramakrishnan and Floyd, 1999) as in the current TCP.

In Diff-C-TCP, the congestion window size is appropriately controlled in the presence of either network congestion or corruption. Congestion window is halved using Fast Recovery algorithm when there is network congestion (explicitly notified by ECN.ECHO packets), and persists at the previous value in the presence of corruption. There are two mechanisms that might be applied to adjust congestion window when Diff-C-TCP sender detects corruption: (i) keep *cwnd* unchanged as the previous value; (ii) use Congestion Avoidance algorithm to slowly increase *cwnd*. In our algorithm, we adapt the first mechanism: make congestion window persist in the previous value.

ECN mechanism will be most effective if it is used with active queue management (such as RED) (Ramakrishnan and Floyd, 1999) as illustrated in Figure 5. In active queue management, when a buffer reaches a certain threshold, the router will send a CE packet to the TCP receiver. Routers send CE packets before their buffers overflow. Therefore, packet drops

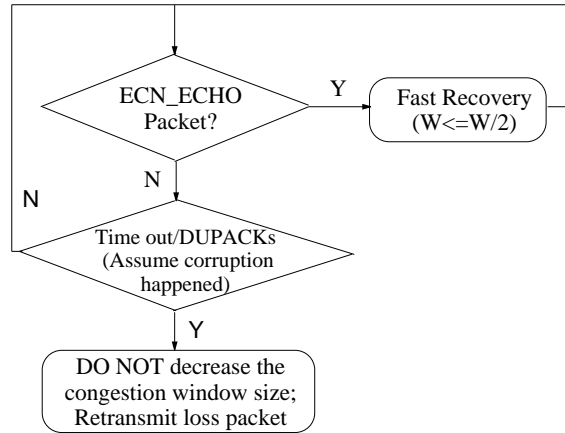


Figure 4: States transition diagram of Diff-C-TCP kernel.

due to congestion happen only after the router has sent CE packets. Upon receiving the CE packet, the TCP

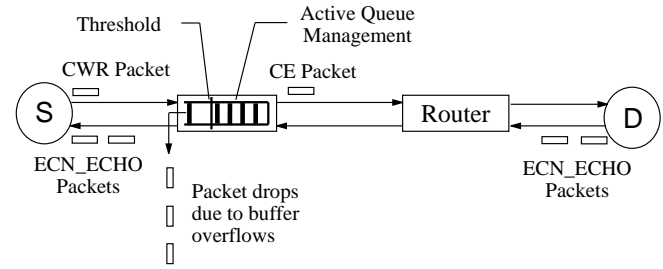


Figure 5: A simple Diff-C-TCP based model.

receiver will keep sending ECN.ECHO packet back to the sender until it receives a CWR packet from the sender, which means the sender has responded to network congestion. The sender only responds to the first ECN.ECHO packet and ignores others up to one RTT.

### 3.3 Illustration of Diff-C-TCP

Depending on the threshold of RED and the level of network congestion, ECN.ECHO packets can arrive at the sender either before or after the retransmit timer times out due to congestion packet losses (caused by buffer overflow). Our proposed Diff-C-TCP is effective in both the above cases as described below.

#### 3.3.1 Case 1: Retransmit timer times out after ECN.ECHO packets are received by the sender.

In this case, the sender will respond to congestion as indicated by the receipt of ECN.ECHO packets. This case is desirable.

### 3.3.2 Case 2: Retransmit timer times out before ECN\_ECHO packets are received by the sender.

In this case, as shown in Figure 6, the retransmit timer timeout happens at time  $t_1$ , and ECN\_ECHO packets are received by the sender at time  $t_2$ . If the difference between  $t_1$  and  $t_2$  is small enough, though the TCP sender does not respond to packet losses indicated by retransmit timer timeout at  $t_1$  (according to our Diff-C-TCP), ECN\_ECHO packets will arrive very quickly, which will trigger Fast Recovery mechanism to relieve the network out of congestion.

The difference between  $t_1$  and  $t_2$  can be decreased by decreasing  $t_2$  as described below. As mentioned above, using active queue management such as RED, when buffer reaches threshold, it will send the CE packet to receiver. Upon receiving the CE packet, the receiver starts to send ECN\_ECHO packets to the sender. Though it is difficult to control the travel time of ECN\_ECHO packets from the receiver to the sender, we can make the receiver send ECN\_ECHO packets earlier by letting the router send the CE packet earlier. The earlier ECN\_ECHO packets are sent, the earlier they arrive at the sender, i.e., the smaller the value of  $t_2$  is. The time when the router sends the CE packet is decided by the value of threshold. Therefore, an optimum value of RED's threshold is very important for the sender to receive congestion notification quickly. Optimal RED threshold is one of our current research topics.

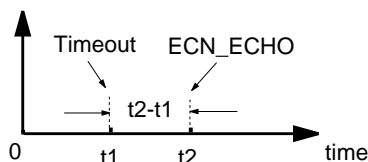


Figure 6: Time sequence of Case 2.

## 4 Simulation Methodology

We have evaluated the performance of our Diff-C-TCP algorithm using the *ns* (*ns* Version 2.1b6) simulation tool from Berkeley (Berkeley/LBNL, 2000). The ECN implementation is based on RFC 2481 (Ramakrishnan and Floyd, 1999). Our network topology for conducting simulations is shown in Figure 7.

Two local area network (10 Mbps) are connected using a lossy wireless link (64Kbps) with a propagation delay of 280ms. Like previous researchers (Balakrishnan et al., 1997; Parsa and Garcia-Luna-Aceves, 1999), a Uniform random error model is used

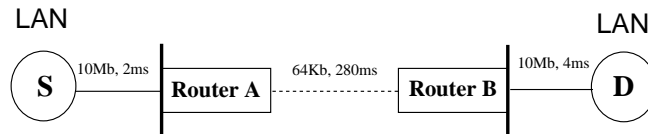


Figure 7: LAN interconnection using a lossy wireless link.

to generate random errors on the wireless link. Instead of dropping packets at routers, RED routers are used in our simulations to set the CE bit in the packet header.

All the links in Figure 7 are labelled with a (bandwidth, propagation delay) pair. The full-duplex link between router A and router B has a BER (bit-error rate), which varies between  $1e^{-7}$  to  $1.2e^{-4}$  in our simulation. The receiver's advertised window size, which is also equal to the initial *ssthresh* at the sender, is set to 30 segments. The packet size is set to 1000 bytes (when BER is below  $5e^{-5}$ ) or 512 bytes (when BER exceeds  $5e^{-5}$ ). Ftp was used in our simulation to transfer data from the source to destination.

## 5 Simulation Results

In this section, we present and compare the *goodput* (bit/s) (the amount of useful information being received by the receiver per second, excluding errors) and the *normalized throughput* of our proposed Diff-C-TCP with the traditional TCP.

Figure 8 compares the goodput in bit/s of both Diff-C-TCP and the current TCP with ECN capability. The normalized throughput of both the TCPs are shown in Figure 9. We see that Diff-C-TCP's throughput is much higher than that of the current TCP with ECN capability. At a BER of  $5e^{-5}$ , the goodput of our Diff-C-TCP is almost **5 times** higher than that of the current TCP. From Figures 8 and 9, this improvement is much higher at higher values of BER. In addition, the throughput of the current TCP with ECN suffers more severely than our Diff-C-TCP as the error rate increases. We can also see that, with the increase of the value of BER, the throughput of the current TCP with ECN capability decreases much faster than the throughput of our Diff-C-TCP. For example, according to Figure 8, when the value of BER increases from  $1e^{-5}$  to  $5e^{-5}$ , the current TCP's goodput decreases by **77 %** in contrast to our Diff-C-TCP whose goodput only decreases by **12 %**. This is also valid for normalized throughput as shown in Figure 9.

As mentioned previously, the current TCP with ECN makes an assumption that every loss event is caused by network congestion, and a congestion control algorithm is triggered. As a result, the congestion

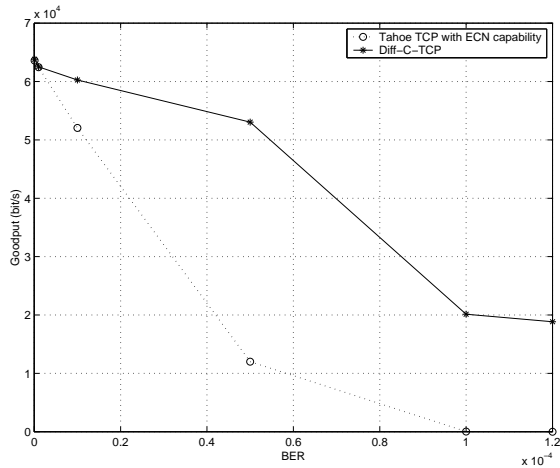


Figure 8: Comparison of goodput (bit/s).

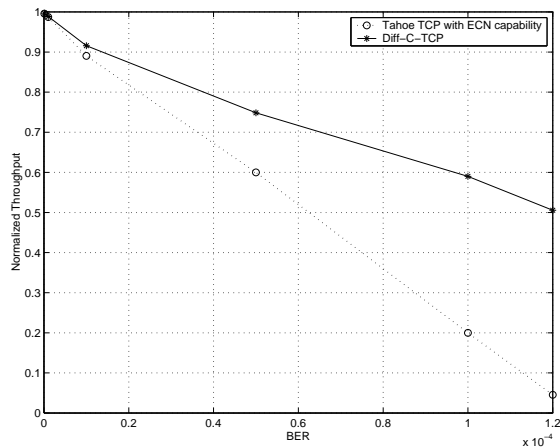


Figure 9: Comparison of normalized throughput.

window size must be reduced. Therefore, each loss event, regardless of whether it is due to congestion or corruption, degrades the throughput. With Diff-C-TCP, all packet losses are assumed to be caused by link errors, and network congestion is indicated by the receipt of ECN\_ECHO packets. The congestion window will not be changed in the presence of corruption losses. Thus, only network congestion can affect its throughput. Furthermore, for the current TCP with ECN, a higher value of BER results in more packet drops and more frequent reduction of the congestion window. Because of this, higher values of BER result in high frequency of reduction of the congestion window size. It is therefore almost impossible for the congestion window size to reach a high value, even during a non-congestion-loss period.

## 6 Conclusion

We have derived a comprehensive model for achieving zero congestion loss. Based on the possibility of zero-loss congestion control, we proposed a new TCP algorithm (named Diff-C-TCP) to improve the TCP throughput in the presence of non-congestion related losses in a lossy wireless environment. Diff-C-TCP improves the network throughput by assuming that all loss events are caused by link errors, unless the network explicitly informs the sources of congestion. Network congestion is explicitly indicated by the receipt of ECN\_ECHO packets.

The proposed Diff-C-TCP has been studied in detail using simulation and has been found to significantly improve TCP performance in lossy wireless links. Our simulation results have shown that we have achieved up to 5 times throughput improvement, over the current TCP with ECN for data transfer across a typical wireless link with high bit-error rates.

## A The Proof of the Ergodicity of Poisson Process

Poisson process can be expressed as (See Figure 10)

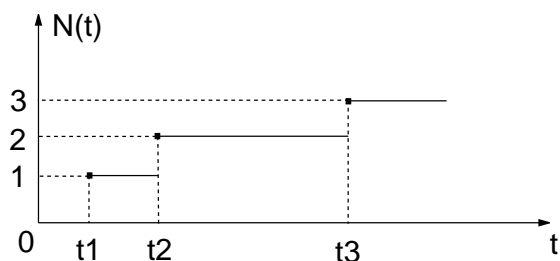


Figure 10: Poisson process.

$$N(t) = \sum_0^{\infty} u(t - T[n]). \quad (18)$$

As mentioned in Section 2.4,  $T[n] = \sum_1^n X_k$  is the arrival time for a loss event. Because  $N(t)$  is a *Poisson process*, during any time interval length of  $t$ , say,  $(0, t)$ , the total number of events is Poisson distributed with *mean*  $\lambda t$ .

Define a random sequence  $Y[n]$  = the total number of events happened in the time interval  $(0, t)$ , according to the above analysis,

$$E[Y[n]] = \lambda t. \quad (19)$$

Then, according to *Strong Law of Large Numbers*,

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=1}^n Y[k] = \lambda t. \quad (20)$$

Thus,

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=1}^n Y[k] = \lambda t = E[N(t)]. \quad (21)$$

In other words, the time average is equal to the expected value. Therefore, Poisson process is ergodic in mean.

## REFERENCES

- Abouzeid, A. and Roy, S. (2000). Analytic understanding of RED gateways with multiple competing TCP flows. In *GLOBECOM*, pages 555–560, San Francisco, CA.
- Abouzeid, A., Roy, S., and Azizoglu, M. (2000). Stochastic modeling of tcp over lossy links. In *INFOCOM*, Tel Aviv, Israel.
- Bai, H. and Atiquzzaman, M. (2003). Error modeling schemes for fading channels in wireless communications: A survey. *IEEE Communications Surveys and Tutorials*, 5(2):2–9.
- Balakrishnan, H., Padmanabhan, V., Seshan, S., and Katz, R. (1997). A comparison of mechanisms for improving TCP performance over wireless links. *IEEE/ACM Transactions on Networking*, 6(5):756–769.
- Berkeley/LBNL, V. P. U. (2000). ns v2.1b6: Network simulator. <http://www-mash.cs.berkeley.edu/ns/>.
- Biaz, S. and Vaidya, N. (1997). Using end-to-end statistics to distinguish congestion and corruption losses: A negative result. Technical report 97-009, Texas A&M University.
- Biaz, S. and Vaidya, N. (1998a). Discriminating congestion losses from wireless losses using inter-arrival times at the receiver. Technical report 98-014, Texas A&M University.
- Biaz, S. and Vaidya, N. (1998b). Sender-based heuristics for distinguishing congestion losses from wireless transmission losses. Technical report 98-013, Texas A&M University.
- Dawkins, S., Montenegro, G., Kojo, M., Magret, V., and Vaidya, N. (2000). End-to-end performance implications of links with errors. draft-ietf-pile-error-03.txt.
- Floyd, S. (1994). TCP and explicit congestion notification. *ACM Computer Communication Review*, 24(5):10–23.
- Kunniyur, S. and Srikant, R. (2000). End-to-end congestion control schemes: Utility functions, random losses and ECN marks. In *INFOCOM*, pages 1323–1332, Tel Aviv, Israel.
- Liu, C. and Jain, R. (2001a). Improving explicit congestion notification with mark-front strategy. *Computer Networks*, 35(2-3):185–201.
- Liu, C. and Jain, R. (2001b). Improving explicit congestion notification with the mark-front strategy. *Computer Networks*, 35(2-3):185–201.
- Mistra, A., Ott, T., and Baras, J. (1999). The window distribution of multiple TCPs with random loss queues. In *GLOBECOM*, pages 1714–1726, Rio de Janeiro, Brazil.
- Parsa, C. and Garcia-Luna-Aceves, J. (1999). TULIP: A link-level protocol for improving TCP over wireless links. In *WCNC*, pages 1253–1257, New Orleans, Louisiana.
- Ramakrishnan, K. and Floyd, S. (1999). A proposal to add Explicit Congestion Notification (ECN) to IP. RFC 2481.
- Samaraweera, N. K. G. and Fairhurst, G. (1998). Reinforcement of TCP error recovery for wireless communication. *Computer Communication Review*, 28(2).