Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2014, Article ID 134620, 11 pages
http://dx.doi.org/10.1155/2014/134620

Research Article

ECN-Based Congestion Probability Prediction over Hybrid Wired-Wireless Networks

Jin Ye,[1] Jiawei Huang,[2] Jianxin Wang,[2] Shigeng Zhang,[2] and Zhenrong Zhang[1]

[1] *School of Computer and Electronic Information, Guangxi University, Nanning 530004, China*
[2] *School of Information Science and Engineering, Central South University, Changsha 410083, China*

Correspondence should be addressed to Jiawei Huang; jiaweihuang@csu.edu.cn

Received 6 December 2013; Accepted 13 February 2014; Published 16 March 2014

Academic Editor: Zhangbing Zhou

Network congestions and wireless link errors are both potential reasons for packet losses in hybrid wired-wireless networks. Their coexistence necessitates the design of new mechanisms that can differentiate network states more precisely. In this paper, a method is proposed to distinguish between the two reasons of packet losses in hybrid wired-wireless networks (i.e., congestions and wireless link errors) and calculate the probability of network congestion in the former case. This method combines the information contained in CE bits of a sequence of ECN-enabled acknowledge packets to calculate the probability of network congestion; thus it is more accurate than methods using CE bits of a single acknowledge packet. Analysis on the effectiveness of the proposed method in calculating congestion probability is performed via simulations. The ability to differentiate precise network states helps a TCP source response to ECN feedback more adaptively. We discuss how to enhance existing TCP variants with the proposed method. Simulation results show that the enhanced TCP variants can effectively improve network performance.

1. Introduction

Congestion control is one of the most basic issues in networks. Due to the high quality of links, packet losses are almost always caused by network congestions in traditional wired networks. Thus a TCP sender can identify network congestions by observing packet losses and react accordingly to avoid worsening the network state. For example, a TCP sender that identifies network congestion usually lowers its sending rate or halves its congestion window size in order to avoid congestion collapses which may further degrade the performance of the network.

However, the reasons for packet losses are more complicated in wireless networks. Compared with wired networks, link quality in wireless networks may be poor because of higher bit error rate, time-varying capacity and delay, or higher mobility of devices. In wireless networks, besides network congestions, channel errors may also cause packet losses. Though MAC-layer retransmission could recover some packet loss due to channel error, when the wireless channel condition is very bad, the wireless loss becomes unavoidable. For example, when the retry number reaches the retry limit in IEEE 802.11, the packet has to be dropped. Thus schemes that can differentiate network states more precisely are needed in hybrid wired-wireless networks to improve performance of TCP protocols [1, 2]. A host in such hybrid networks should have the ability to distinguish between packet losses that are caused by network congestions and that are caused by other factors, and it should react differently according to different cases.

Explicit Congestion Notification (ECN) [3] provides a mechanism that allows end-to-end notification of imminent network congestions without really dropping packets. When the buffer queue length is higher than a given threshold, the intermediate routers mark the data packets with the Congestion Experienced (CE) code point instead of dropping them in order to signal impending congestion. Upon receiving a date packet with the CE code point, the receiver echoes back this congestion indication using the ECN-Echo flag in the ACK header. When a sender receives an ACK with the ECE bit, it reduces its congestion window as for a packet drop. This mechanism has already been supported by mainstream operating systems such as Microsoft Vista [4] and NetBSD [5]. In recent years, many congestion control mechanisms based on ECN have been proposed, such as XCP [6], VCP [7], D^2TCP [8], and VCP-BE [9]. These methods mainly focus on how to select congestion metrics for routers and how to feed congestion signals back to senders. ECN is already considered as an important approach to marking congestion and controlling TCP behaviors.

Due to its ability in indicating network congestions without packet dropping, ECN has been applied in hybrid wired-wireless networks to help control TCP behaviors [10–13]. One typical work is Wireless ECN (WECN) proposed by Peng [14]. In WECN, a packet loss will be judged as induced by congestion if the ECN-Echo flag of ACK packet is marked with 1; otherwise, it will be judged as induced by other errors. WECN has been applied in several TCP protocols, for example, TCP Jersey [15], TFRC [16, 17], TCP Snoop [18], and cross-layer design TCP [19].

However, some researches have pointed out that the ECN mechanism cannot effectively distinguish different factors that result in packet losses. This is because ECN's ability in indicating different factors depends on how accurate ECN can be aware of congestion [20].

To cope with this problem, Sridharan et al. proposed WMECN [21] based on MECN [22], in which different weights are assigned to two congestion feedback bits. Then the congestion state is categorized into different degrees according to the weighted summation of all the congestion feedback bits. This method outperforms previous congestion control methods that use single ECN feedback. However, the improvement in TCP performance depends largely on the threshold values. Furthermore, this method does not take changes of network states into account.

In this paper, we mainly focus on how TCP sender should respond to congestion signals contained in ECN-enabled ACK packets in hybrid networks in which both link errors and network congestions may cause packets to be dropped. Two aspects should be considered.

Firstly, the delay in hybrid networks should be considered. The feedback delay in hybrid networks is prominently larger and more uncertain than that in traditional wired networks. Simulations by Raghunathan and Kumar show that the throughput in wireless networks is oscillatory and sometimes can be orders of magnitude less than that in wired networks [23]. And for RTT (Round Trip Time) there are similar observations. As a result, even if the congestion state can be notified by ECN correctly, the long and uncertain delay may make the feedback useless for congestion control. Therefore how to alleviate non-real-time effects of ECN is one crucial problem.

Secondly, no matter how routers can be aware of congestions, the information contained in ECN feedback can only indicate imminent congestions. They cannot indicate that congestions already occurred. Because the network state changes dynamically, the congestions may occur or not occur in the next interval; that is, the congestion may either become true if packets are discarded continually or will be cleared up if the link becomes good in a short period. How the randomness of ECN impacts TCP is analysed in [24]. As a result, how TCP senders respond to such inaccurate feedback information should also be considered.

Considering the two aforementioned aspects, we argue that ECN feedback information should not be directly used to adjust TCP behavior; more effective utilization method should be designed. Aiming to mitigate the impact of large delay and reduce the blindness in congestion state judgement, we propose a method to calculate the *probability of congestion* based on an ECN feedback sequence. Our method is more reasonable than previous ones that are based on single loss event or single ECN feedback. Simulation results show that the proposed method can effectively predict congestions in hybrid wired-wireless networks. The proposed method can also be easily extended into existing TCP variants and used to improve TCP performance in hybrid networks.

The rest of the paper is organized as follows. Section 2 presents how to calculate CP (Congestion Probability) and investigates the correlation between CP and network states. Section 3 describes in detail how to extend CP mechanism into existing TCP variants. Then, extensive simulations are conducted to investigate the performance gain by using CP and the results are analyzed in Section 4. Finally, concluding remarks for our work are presented in Section 5.

2. CPECN: Congestion Probability Based on ECN

The key idea of CPECN is to analyze the distribution of CE bits from ECN feedbacks and predict network states according to the correlation between feedback serials.

2.1. Calculation of Congestion Probability. A single ECN feedback only reflects the network state at one earlier interval, while an ECN feedback sequence can reflect the dynamic change of the network state during the past several continuous intervals. So we propose to combine the information contained in CE bits of a sequence of ACK packets together to predict the network state.

Assume that a sender receives an ECN feedback sequence containing m ECN feedbacks. We calculate $\mathrm{CP}(t)$, the congestion probability at the current time t, as

$$\mathrm{CP}(t) = \frac{1}{m} * \sum_{i=0}^{m-1} w_i * \mathrm{ACK}[i].\mathrm{ecnecho}, \tag{1}$$

where $\mathrm{ACK}[i]$.ecnecho is the value of the CE bit in $\mathrm{ACK}[i]$, w_i is the weight of $\mathrm{ACK}[i]$, and the values of all w_i are normalized such that $\sum_{i=0}^{m-1} w_i = 1$.

The value of $\mathrm{CP}(t)$ is calculated using m serial ACKs received till time t and denoted by $\mathrm{ACK}[i]$ and $\mathrm{ACK}[i-1], \ldots, \mathrm{ACK}[i-m+1]$, respectively. For the sake of simplicity in presentation, in the rest of this paper we express $\mathrm{ACK}[i]$.ecnecho as $\mathrm{ACK}[i]$.

The weights of CE bits in different ACK packets are assigned using an *exponentially weighted moving average* method. The ACKs are divided into several segments and all ACKs in the same segment will be assigned with same weight. The reason is that large number of ACKs increases the feedback delay, while single ACK could not reflect the congestion trend. To achieve the tradeoff between sensitivity and stabilization, the m ACKs are divided into n segments and ACKs in segment j are assigned with weight w_j. The value of CP(t) can be calculated as

$$\mathrm{CP}(t) = \sum_{j=1}^{n} w_j * \frac{n}{m} * \sum_{i=(m/n)(j-1)}^{(m/n)j-1} \mathrm{ACK}[i]. \quad (2)$$

We calculate the CP(t) with moving weighted average value of n samples, similar to DCTCP [25]. However, CPCEN tracks the change of CP(t) value to estimate congestion level, while DCTCP only calculates ECN marking fraction similar to CP(t) value. This is the key difference between CPECN and DCTCP. Generally, the newer the ACK packets are, the more accurate it is to predict current network state. So we should assign large weights for new ACKs. The weights of ACKs in each segment are assigned as

$$w_j = \frac{1}{\alpha} * w_{j-1}, \quad \alpha \in (0,1), \quad (3)$$

while all w_j satisfy

$$\sum_{j=1}^{n} w_j = 1. \quad (4)$$

It is clear that, since the value of α is less than 1, the jth segment's weight w_j is higher than the previous segment's weight w_{j-1}.

In high dynamic network, it is very difficult to get accurate traffic state with several packets. However, using large number of packets will limit the feedback speed. Like the way that TCP uses three duplicate ACKs to reflect the packet loss, we choose the four ACKs as one segment to track the congestion state. After receiving three duplicate ACK packets, the sender enters the congestion avoidance mode. Similar to this method, we choose a period for four ACK packets to compose one segment and reflect the congestion state. Thus, we have

$$m = 4 * n. \quad (5)$$

With (3), (4), and (5), we can rewrite (2) as

$$\mathrm{CP}(t) = \sum_{j=1}^{n} w_j * \frac{1}{4} * \sum_{i=4(j-1)}^{4j-1} \mathrm{ACK}[i], \quad i \in [1, m]. \quad (6)$$

Equation (6) can be expanded to

$$\begin{aligned}\mathrm{CP}(t) = {}& w_1 * \frac{1}{4} * \sum_{i=0}^{3} \mathrm{ACK}[i] + \frac{w_1}{\alpha} * \frac{1}{4} * \sum_{i=4}^{7} \mathrm{ACK}[i] \\ & + \cdots + \frac{w_1}{\alpha^{n-1}} * \frac{1}{4} * \sum_{i=4(n-1)}^{4n-1} \mathrm{ACK}[i].\end{aligned} \quad (7)$$

If CP($t+1$) > CP(t), the congestion probability increases; otherwise, the congestion probability decreases. Therefore the change of CP can reflect the change of network state. Defining $\Delta\mathrm{CP} = \mathrm{CP}(t+1) - \mathrm{CP}(t)$, we have

$$\begin{aligned}\Delta\mathrm{CP} = {}& \frac{w_i}{4} * \Big((\mathrm{ACK}[4] - \mathrm{ACK}[0]) \\ & + \frac{1}{\alpha} * (\mathrm{ACK}[8] - \mathrm{ACK}[4]) \\ & + \cdots + \frac{1}{\alpha^{n-1}} * (\mathrm{ACK}[4n] - \mathrm{ACK}[4n-4]) \Big) \\ = {}& \frac{w_i}{4} * \Big(- \mathrm{ACK}[0] + \left(1 - \frac{1}{\alpha}\right) \mathrm{ACK}[4] \\ & + \left(\frac{1}{\alpha} - \frac{1}{\alpha^2}\right) \mathrm{ACK}[8] \\ & + \cdots + \left(\frac{1}{\alpha^{n-1}} - \frac{1}{\alpha^n}\right) * \mathrm{ACK}[4n] \Big).\end{aligned} \quad (8)$$

From (8), it is very clear that ΔCP is only related to ACK[$4i$], that is to say, the last ACK of each segment. We can also see that the value of ΔCP lies in a discrete set that contains 2^{n+1} elements. If the CE bit of the last ACK in the last segment is marked with 0, the value of ΔCP will be in $[-1, 0]$. In this case, it can be concluded that the congestion probability is decreasing, which has the same meaning with one single ECN feedback. On the contrary, if the CE bit of the latest arriving ACK is marked with 1, the value of ΔCP will be in $[-1, 1]$. In this case, it cannot be determined whether the network is going to become congested or not. However, $\Delta\mathrm{CP} \le 0$ always indicates that the network congestion may be relieved, no matter what the CE bit of the new arriving ACK is.

In this paper, the last 8 ACKs are divided into 2 segments; that is, $n = 2$ and $m = 8$. Let $w_1 = 0.25 * w_2$. Using (4) and (3), we can get $w_1 = 0.2$ and $w_2 = 0.8$. It should be noted that the values of m, n, and α are chosen according to simulation results. All these parameters are used in the following simulations. The value of CP can be calculated with Algorithm 1. For example, with these parameter settings, CP is equal to $0.2 * (\mathrm{ACK}[7] + \cdots + \mathrm{ACK}[4]) + 0.8 * (\mathrm{ACK}[3] + \cdots + \mathrm{ACK}[0])$.

```
(1)  $\mathrm{CP}(t+1) = \mathrm{CP}(t)$
(2)  int ecnecho = hdr_flags::access(pkt)
     → ecnecho()
(3)  float $w1, w2$
(4)  for $i = 1$ to $m - 1$ do
(5)      $\mathrm{ACK}[i-1] = \mathrm{ACK}[i]$
(6)  end for
(7)  $\mathrm{ACK}[m-1]$ = ecnecho
(8)  $u_1 = \mathrm{ACK}[m-1] + \cdots + \mathrm{ACK}[m/2]$
(9)  $u_2 = \mathrm{ACK}[m/2-1] + \cdots + \mathrm{ACK}[0]$
(10) $\mathrm{CP}(t) = w1 * u_1 + w2 * u_2$
(11) return CP($t$)
```

Algorithm 1: Procedure of calculating CP.

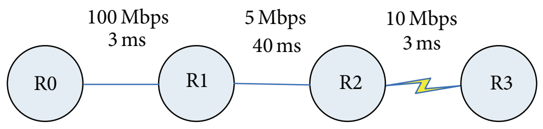


Figure 1: Network topology 1.

2.2. Analysis of Congestion Probability. In this subsection, the correlation between CP value and network state is analyzed by simulations. The network topology used for the simulation is depicted in Figure 1, which is used by TCP Westwood to evaluate TCP performance over hybrid networks [26]. Since CPECN is focused on TCP improvement over hybrid networks, the same topology is used to analyze the congestion probability.

In the simulated scenario, one TCP Reno flow traverses along a bottleneck link and reaches a wireless terminal. As shown in Figure 1, the TCP flow is sent from $R0$ to $R3$. The bandwidth of the access link from $R0$ to $R1$ is 100 Mbps with 3 ms delay; the link from $R1$ to $R2$ is a simulated bottleneck link which will traverse several routers in Internet and its bandwidth is 5 Mbps and its delay is 40 ms; the last hop, $R2$ to $R3$, is a wireless link with bandwidth of 10 Mbps and delay of 3 ms. The buffer of routers in the bottleneck link is set to be 30 packets based on "rule of thumb." Other parameters use NS2 default settings. The simulation lasts for 100s.

When the sender receives the signal of packet dropping or the ACK packet in which the CE bit is 1, it regulates the sending window size according to TCP Reno. Packets on routers will be marked if the average queue length exceeds the given threshold.

The results are shown in Figure 2. Figure 2(a) shows how the size of TCP sending window varies at different times when there is no packet losses due to error in the wireless links. We can find that the TCP sending window size varies very regularly. However, where there exist packet losses caused by link errors, the size of TCP sending window changes irregularly, as shown in Figure 2(b). From Figures 2(a) and 2(b), it is obvious that the average size of the sender window without packet losses due to link errors is larger than that with packet losses due to link errors.

The queue length of the bottleneck link is shown in Figures 2(c) and 2(d). As shown in Figure 2(c), there are 4 periods in which the queue length is larger than the given threshold. In such cases, the router marks the packets and the sender will receive ACK packets with CE bit marked. It is clear that the changes of the sending window are consonant with that of the queue length. In Figure 2(d), there is only one period in which the queue length is larger than the given threshold. But the sending window does not change according to the queue length because there are packet losses due to error in wireless links.

The statistic of CP under different situations are shown in Figures 2(e) and 2(g), respectively. It is shown that the change trend of CP is consistent with that of the queue length. So we can capture network states according to ΔCP. The detailed Figures 2(f) and 2(h) show how CP value ranges in one of the period, which show that CP can characterize congestion state sensitively and accurately. Compared with end-to-end metrics or single ECN feedback, CP can provide more accurate and richer information about network states. For example, single ECN feedback only denotes whether congestion exists or not, while could not reflect the change trend of congestion state.

3. TCP Protocol with CPECN

TCP sender with CPECN responds to loss events based on CP values. After receiving an ACK, a TCP sender will recalculate CP. If the arriving ACK is new, the TCP sender operates the same as TCP NewReno does. If the arriving ACK is duplicate and $\Delta CP > 0$, the TCP sender concludes that a network congestion occurs and adjusts its sending rate. If the arriving ACK is duplicate and $\Delta CP \leq 0$, the TCP sender only retransmits this data packet.

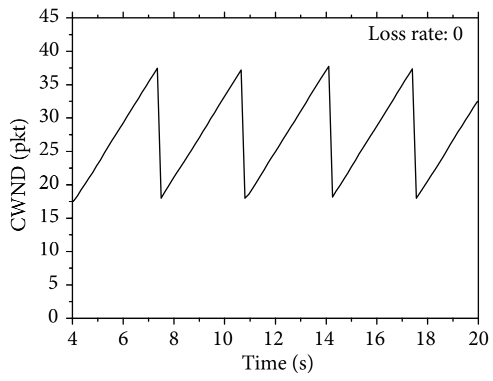
By using CPECN, a TCP sender's behavior is modified according to ΔCP when packet losses occur. This is an important distinguishing feature of the modified TCP compared to other TCP variants. For example, in TCP Reno a sender will trigger congestion control after it identifies loss events without caring about the situation of midrouters; a TCP sender will blindly trigger congestion control if single ECN signal is used to indicate network state. At duplicate ACK arrival, a TCP sender with WECN will decide whether to do Rate Control by single ECN signal, while a TCP sender with CPECN will decide whether to do Rate Control by the change of CP.

Figure 3 is the transition graph of TCP sender modified by CPECN, which indicates how TCP variants with CPECN work.

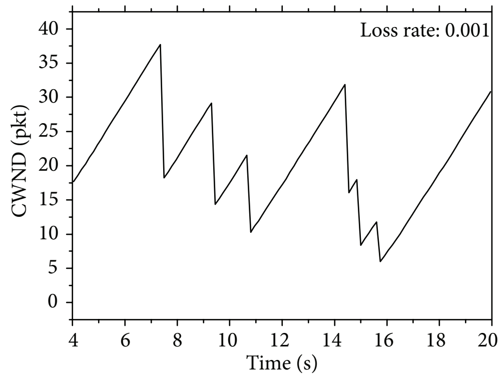
There are two scenarios to show the advantage of TCP with CPECN in hybrid networks. It should be noted that though the two scenarios are specific cases, they could represent some real network scenarios.

As shown in Figure 4, the CE lists at t_1, t_2, and t_3 are given, where t_2 is the time when the sender receives the first ACK packet after t_1, and t_3 is after t_2. At time t_1, the CE list of the latest 8 ACK packets is 00011111. At time t_2, the sender receives a duplicate ACK in which CE = 1. So the CE list of the latest 8 ACK packets is 10001111 at time t_2. In TCP with WECN, the sender will trigger congestion control. However, with CPECN we can calculate that $CP[1] = CP[2] = 0.4$, which means that probability of network congestion occurred at time t_2 is not increased. So TCP sender does not invoke Rate Control procedure at time t_2.

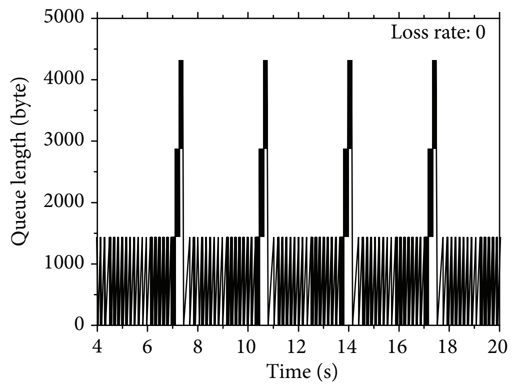
At time t_3, if the sender receives a duplicate ACK in which CE = 1, we have $CP[3] = 0.55$. Because $CP[3] > CP[2]$, the sender predicts that a congestion may occur soon and triggers the congestion control mechanism at the moment. On the contrary, if the sender receives a duplicate ACK in which CE = 0, we have $CP[3] = 0.35$. Because $CP[3] < CP[2]$, the sender does not trigger the congestion control mechanism. Consequently, by using CPECN, the network throughput can be improved by increased link occupancy.



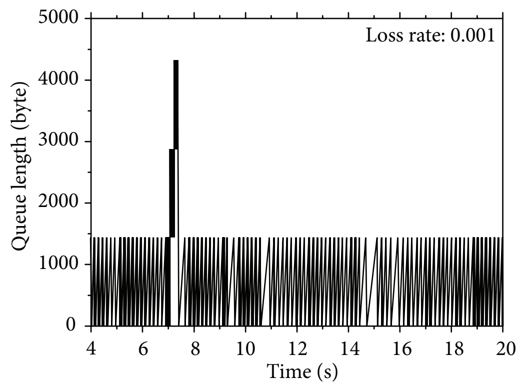
(a) TCP window without link loss



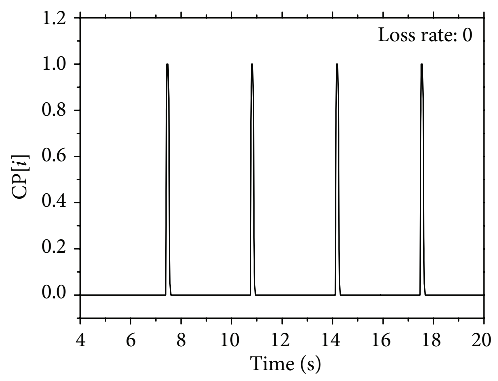
(b) TCP window with link loss



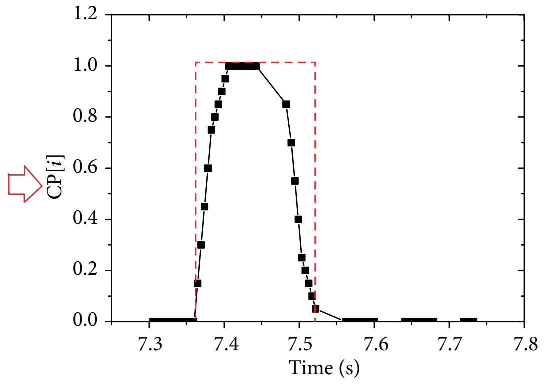
(c) Queue length without link loss



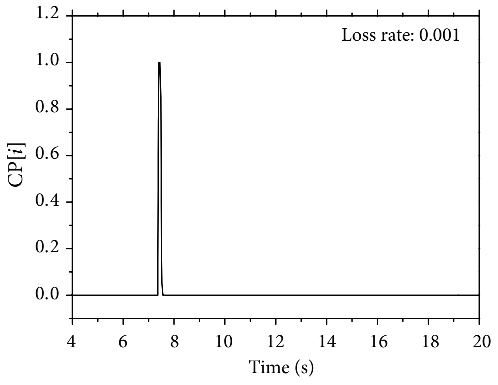
(d) Queue length with link loss



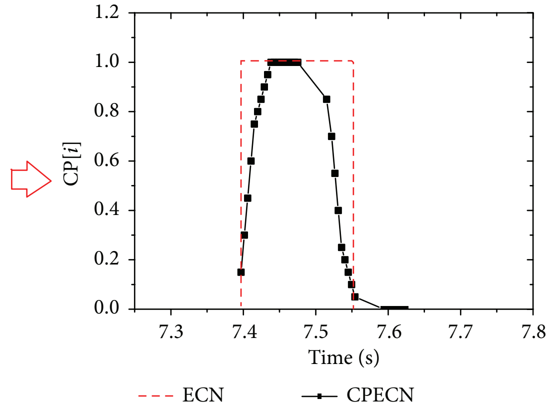
(e) CP without link loss



(f) CP without link loss (large graph)



(g) CP with link loss



(h) CP with link loss (large graph)

Figure 2: Correlation between CP and network state.

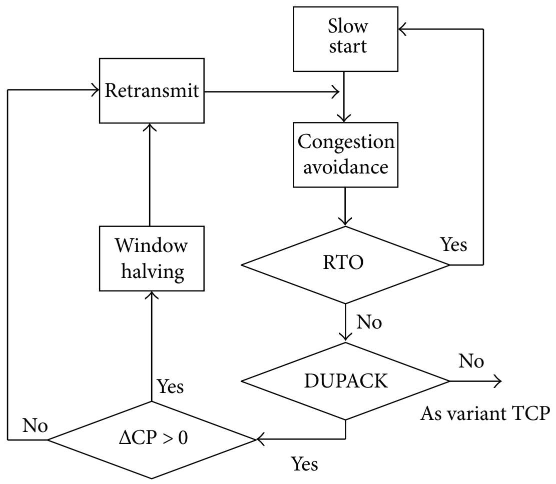


Figure 3: State transition graph of TCP sender with CPECN.

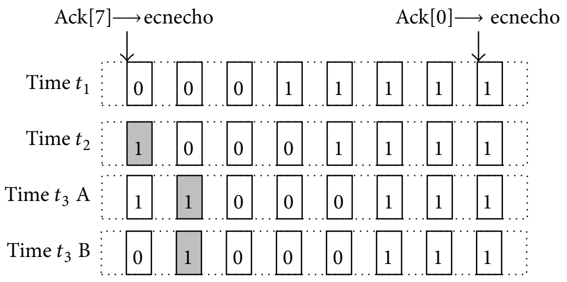


Figure 4: The first scenario.

In the second scenario, we describe the handling process in TCP protocol with CPECN, while noncongestion losses occur successively.

As shown in Figure 5, we assume that the sender receives the duplicate ACK packets with CE = 0 at three successive times, t_1, t_2, and t_3. It implies that three successive loss events may be caused by link error. Because CP[2] < CP[1] and CP[3] < CP[2], the sender does not decrease its sending rate immediately. The sender decreases its sending rate only when it receives an ACK packet with CE = 1 at time t_4, because in this case CP[4] > CP[3]. On the contrary, as long as CP[4] is less than CP[3], the sender will not decrease the sending rate.

It is obvious that by using CPECN, even if the sender receives duplicate ACKs caused by several errors, as long as the queue length does not exceed the threshold or the congestion probability does not increase, the TCP sender will not decrease its sending rate. Contrarily, even if only one duplicate packet is received, the sending rate might be decreased because of CP value being increased.

As a whole, with a precise prediction of the probability of network congestion, CPECN can protect TCP from short-term, recoverable, noncongestive losses and enhance its performance in hybrid network eventually.

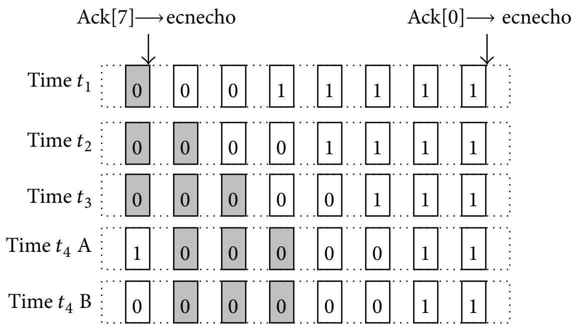


Figure 5: The second scenario.

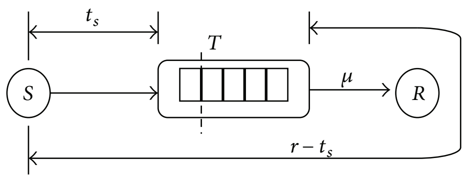


Figure 6: Queue model of single TCP flow.

4. Performance Evaluation

There are two methods to mark CE bits of a packet in routers, that is, probability-based method and threshold-based method. In order to support CPECN, a TCP sender needs to track the change of queue lengths in routers. Thus, routers must provide such determinate information to end hosts with threshold-based marking method. In this type of methods, CE bits of a packet will be marked as 1 if EWMA value of the average queue length exceeds a given threshold, which is less than the buffer size.

Here we describe a simple and reasonable model to analyze dynamic changes of the queue length. We have the following hypotheses in the model: (1) senders always have data to send and will send as many as their windows allow; (2) receiver windows are large enough; (3) there are no delayed ACK; and (4) all packets have the same length. The used analytical model is shown in Figure 6. The sender's window size at time t is denoted as $w(t)$. Round Trip Time (RTT) is r and the bottleneck link bandwidth is u. The value of t_s is the time a packet traverses from the sender S to the RED router with the threshold of T.

For the case there is only one TCP connection along the path, if the bottleneck link bandwidth is u packets per second, the downstream packet interarrival time and the acknowledgement interarrival time on the reserve link must be greater than or equal to $1/u$ [27]. The queue length at the congestion router is $Q(t) = w(t - t_s) - ru$. The maximum queue length in the slow start phase is $2T + ru + 1$ and the maximum queue length in the congestion avoidance phase is $T + 1$. The minimum queue length is $(T - ru + 1)/2$. The optimal threshold $\widehat{T}$ for the RED queue should make the bottleneck link be fully utilized and queue delay be zero; that is, $(T - ru + 1)/2 \leq 0$, which means $\widehat{T} \leq ru - 1$.

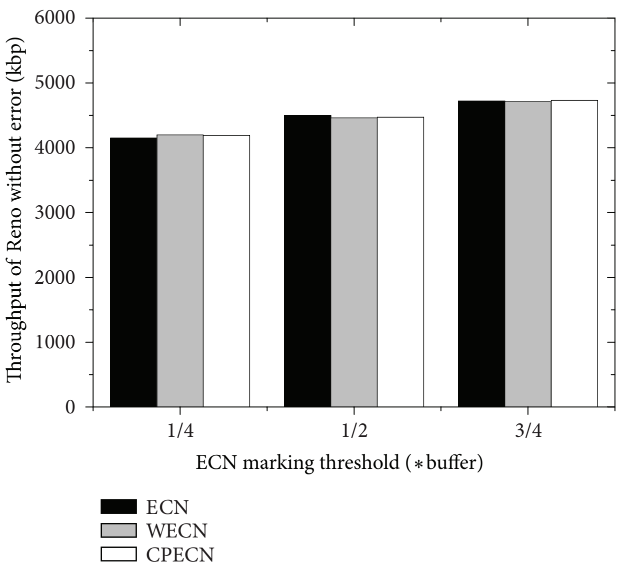


Figure 7: Throughput comparison of Reno without link loss.

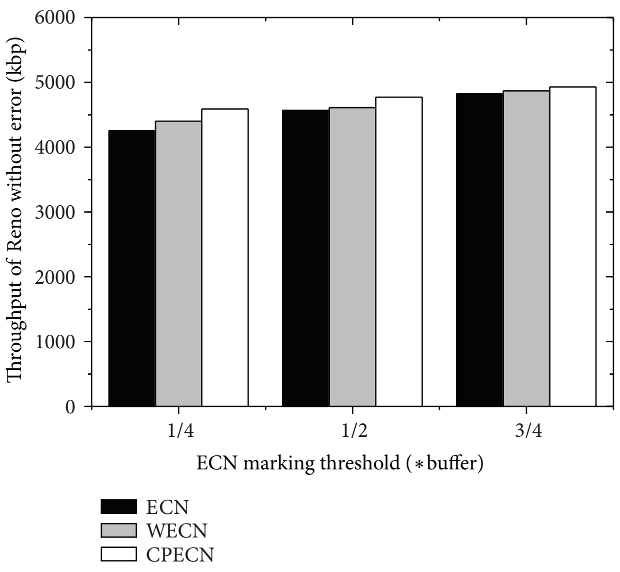


Figure 8: Throughput comparison of SACK without link loss.

For the case there are multiple TCP flows along the path, a similar analysis shows that the optimal threshold $\widehat{T}$ should satisfy $u - m \leq \widehat{T} \leq ru - 1$. It is obvious that the optimal threshold of ECN marking is different in different cases. Some detailed method to find optimal threshold is shown in existing works such as [28]. In order to decouple the effect of ECN mechanism, we evaluate the performance improvement of TCP with CPECN using different ECN marking thresholds. We will show that whatever ECN marking threshold is used, TCP performance with CPECN can be improved. Actually, the different ECN schemes can be applied in different TCP protocols. In the following, we use the same NS2 [29] simulation settings and network topology of Section 2 to analyze the throughput performance of TCP Reno, SACK, and Westwood. The reason we choose Westwood is that it is the typical transport protocol to copy with wireless link error in hybrid networks.

4.1. Throughput Analysis without Link Losses. When the link error rate is zero, the throughput of Reno with ECN, WECN, and CPECN are shown in Figure 7. We use three different marking thresholds for router buffer, that is to say, 1/4, 1/2, and 3/4 (× buffer size of bottleneck link), respectively. In Figure 7, TCP throughput becomes larger with the increasing of marking threshold. It is obvious as shown in Figure 7 that throughput of TCP Reno with three ECN schemes are similar. When marking threshold is equal to 3/4 buffer size of bottleneck link, CPECN throughput is slightly higher than other ECN schemes.

Figure 8 shows the throughput performances of TCP SACK. Compared with TCP Reno, TCP SACK only retransmits the lost packet and thus achieves higher throughput. TCP throughput of Westwood is shown in Figure 9, whose changing trend is similar to TCP Reno. However, TCP throughput with WECN is decreased slightly. This is because WECN only depends on a single ECN feedback to predict

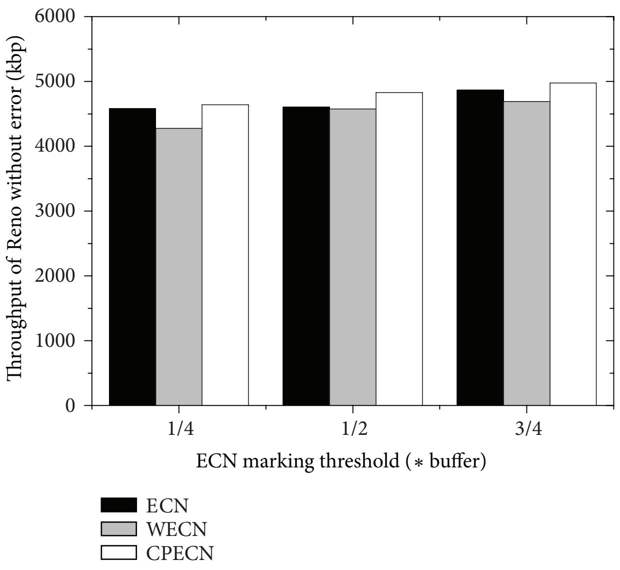
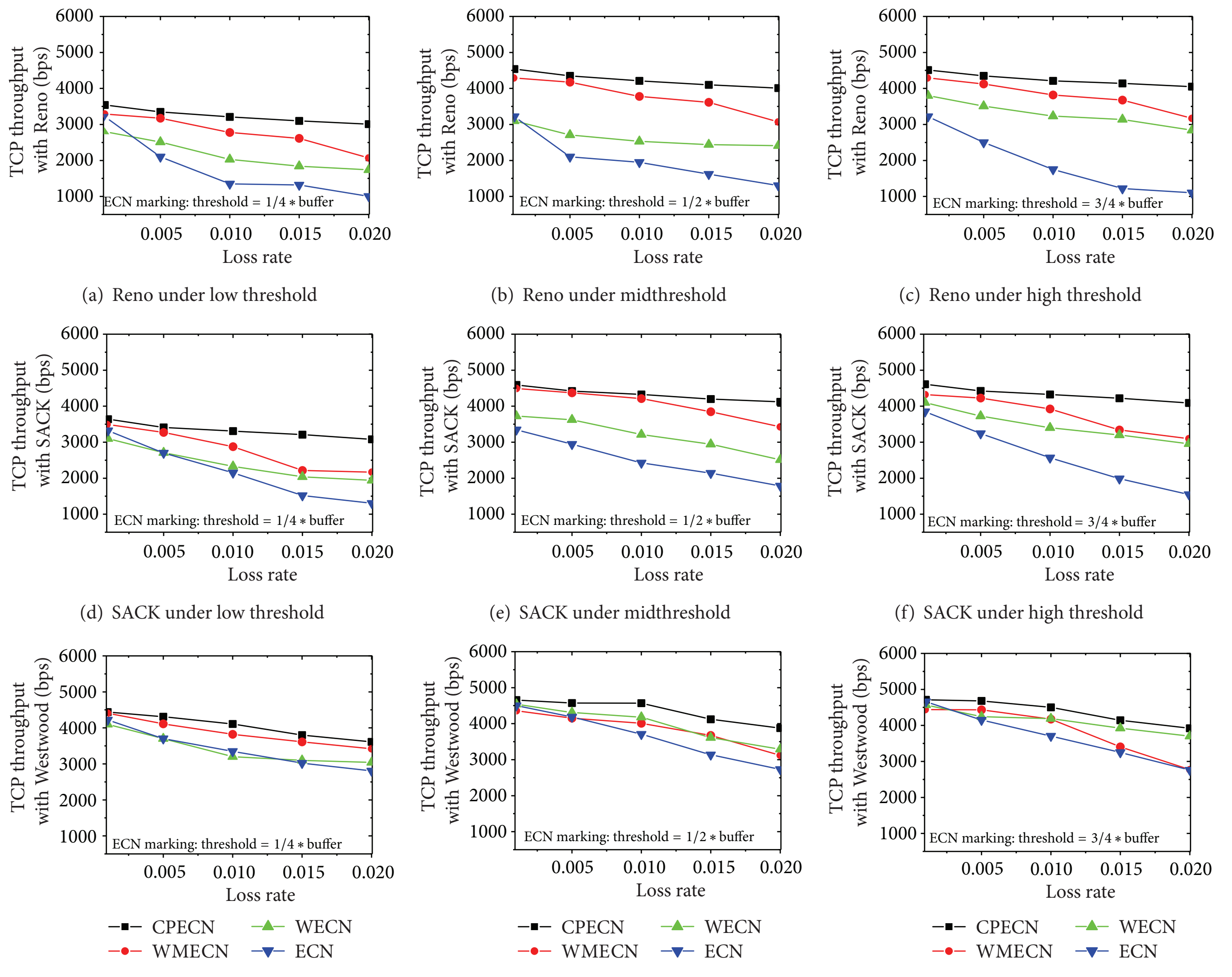


Figure 9: Throughput comparison of Westwood without link loss.

congestion probability, which is sensitive to packet losses. This problem is avoided by CPECN.

It should be noted that we only focus on the performance comparison of three ECN schemes. It is shown that CPECN's throughput is highest in the three ECN schemes. The three typical marking thresholds are set between 0 and 1. If the threshold is 1, the queue management will be the same as traditional Drop Tail method without ECN.

4.2. Throughput Analysis with Link Losses. In this section, we use the two-state Markov-modulated channel model with correlated errors to evaluate the throughput performances with link losses. The wireless link is assumed to be in one of two states: Good or Bad. Let the mean duration of Good and



(a) Reno under low threshold (b) Reno under midthreshold (c) Reno under high threshold

(d) SACK under low threshold (e) SACK under midthreshold (f) SACK under high threshold

(g) Westwood under low threshold (h) Westwood under midthreshold (i) Westwood under high threshold

Figure 10: Throughput comparison with link loss.

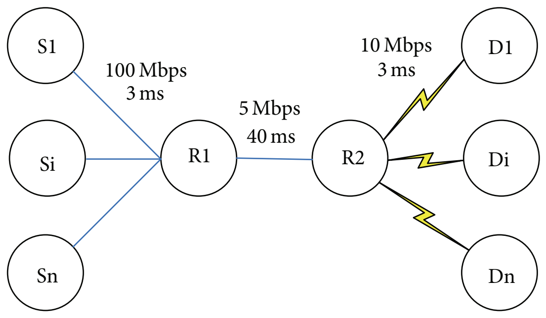


Figure 11: Network topology 2.

Table 1: Fairness comparison of 10 same Reno flows.

Loss rate	ECN	WECN	CPECN
0	0.98	0.991	0.999
0.001	0.993	0.992	0.981
0.005	0.991	0.986	0.988
0.01	0.983	0.981	0.993

Bad states be equal. The packet loss rate in the Good state is assumed to be 0.1%, and the packet loss rate in the Bad state is varied from 0.1% to 2%.

With link loss rate ranging from 0.001 to 0.02, TCP throughput of Reno, SACK, and Westwood are shown in Figure 10, in which the effect of CPECN becomes more distinct as link loss rate increases. It is shown in Figures 10(a)–10(c) that the improvement of TCP throughput also becomes salience as marking threshold increases. For example, when link loss rate is 0.005 and the marking threshold is 1/4 of buffer size, TCP throughput of Reno with CPECN outperforms that of Reno with WECN by 1.18 times. When the marking threshold is 3/4 of buffer size, the improving factor is 1.72 times. TCP SACK performances are presented in Figures 10(d)–10(f). With the increasing of marking threshold, the throughput improvement also becomes higher.

Table 2: Friendliness performance of CPECN with TCP Reno.

ECN : CPECN	Loss rate: 0		Loss rate: 0.001		Loss rate: 0.005		Loss rate: 0.01	
	ECN (kbps)	CPECN (kbps)	ECN (kbps)	CPECN (kbps)	ECN (kbps)	CPECN (kbps)	ECN (kbps)	CPECN (kbps)
3 : 7	1440	**1427**	1351	**1411**	1022	**1231**	824	**913**
5 : 5	1283	**1291**	1253	**1412**	982	**1110**	831	**923**
7 : 3	1057	**1237**	1103	**1252**	953	**1014**	825	**917**

Table 3: Friendliness performance of CPECN with TCP Westwood.

ECN : CPECN	Loss rate: 0		Loss rate: 0.001		Loss rate: 0.005		Loss rate: 0.01	
	ECN (kbps)	CPECN (kbps)	ECN (kbps)	CPECN (kbps)	ECN (kbps)	CPECN (kbps)	ECN (kbps)	CPECN (kbps)
3 : 7	1162	**1174**	1123	**1171**	1295	**1327**	1121	**1199**
5 : 5	1316	**1331**	1315	**1321**	1298	**1322**	1288	**1342**
7 : 3	1519	**1536**	1522	**1529**	1494	**1518**	1378	**1512**

Figures 10(g)–10(i) show the improvement of CPECN over the other three schemes. When the loss ratio is 0.01, TCP throughput of Westwood with ECN is increased by 17% with marking threshold of 1/4 buffer size. When marking threshold is 3/4 of buffer size, corresponding value is 24%. The results show that (1) improvement of TCP Westwood with CPECN is limited and (2) marking threshold on router is of less affection on TCP Westwood with CPECN. This is because that rate adjustment of TCP Westwood can cope with lossy links in some degree.

4.3. Fairness and Friendliness Analysis. Fairness is an important metric for evaluating TCP protocol. Multiple connections of the same TCP scheme must cooperate fairly. In this part, we choose the dump-bell topology in Figure 11, which is the classic topology used for TCP fairness evaluation. There are 10 TCP flows that share a 10 Mbps bottleneck wireless link. In this case, we believe that the Jain's fairness index could evaluate the TCP fairness [30].

4.3.1. Fairness and Friendliness of TCP Reno with CPECN. As shown in Table 1, fairness of TCP Reno decreases with the increasing of loss rate. This is because flows suffering error loss will decelerate and compete for channel again, so short term unfairness exists in some extent. However simulation results of Table 1 show that this impact is not serious. At the same time, fairness of TCP Reno with CPECN approximates to that of TCP Reno with other two ECN schemes. Jain's fairness indexes of ECN, WECN, and CPECN are all as high as about 0.99.

The experiment of friendliness partitions ten flows into two parts, one uses TCP Reno with CPECN and the other uses TCP Reno with traditional ECN.

Table 2 gives the change of mean throughput when there are 10 pairs of connections, of which m are Reno with ECN connections and n are Reno with CPECN connections. The mean throughput of Reno with ECN and CPECN is calculated by summing up the throughput of the same ECN scheme and dividing by the number of connections. It is observed that the mean throughputs of two kinds of connection are approximate in case of no link losses, which indicates that bandwidth allocation of TCP Reno connection with different ECN schemes is close to its fair at bottleneck bandwidth link. When loss rate is 0.01, mean throughput of Reno with CPECN is 10% higher than that of ECN, which is within a tolerate range.

Table 4: Fairness comparison of 10 same Westwood flows.

Loss rate	ECN	WECN	CPECN
0	0.935	0.962	0.997
0.001	0.991	0.995	0.993
0.005	0.992	0.987	0.995
0.01	0.986	0.991	0.992

4.3.2. Fairness and Friendliness of TCP Westwood with CPECN. The experiment of friendliness partitions ten flows into two parts, one uses TCP Westwood with CPECN and the other uses TCP Westwood with traditional ECN. Table 3 gives the change of mean throughput when there are 10 pairs of connections, of which m are Westwood with ECN connections and n are Westwood with CPECN connections. The simulation results show that CPECN scheme can coexist with Westwood when a lossy link occurs.

Fairness comparison of 10 Same TCP Westwood flows is shown in Table 4. The simulation results show that fairness of TCP Westwood with CPECN is favorable.

It can be concluded from the simulation results that TCP Westwood with CPECN performs satisfactorily in the fairness and friendliness aspects. This is because Westwood already overcomes some limitation of traditional TCP protocol. Firstly, Westwood can control sending rate more elaborately than Reno by estimating available bandwidth. Secondly, WECN schemes decouple the error control and congestion control with the help of midway routers. CPECN

Table 5: Comparison of queue management algorithm.

Algorithm	ECN (kbps)	WECN (kbps)	CPECN (kbps)
Drop tail	1733	3478	4125
RED	1745	3542	4375
PI	1738	3518	4410

scheme utilizes congestion notification in an efficient way based on the above two components, so its fairness and friendliness are satisfactory.

4.4. Comparison of Queue Management Algorithm. In this section, we compare the ECN, WECN, and CPECN performances with different queue management algorithms, which are Drop Tail, RED [31] and PI [32]. Drop Tail is default algorithm in current routers. RED, also known as random early drop is an active queue management algorithm that monitors the average queue size and drops or marks packets based on statistical probabilities. PI is a generic control loop feedback controller, which attempts to regulate the queue level to a desired reference value. RED and PI are typical active queue management algorithms, which adopt the default settings in NS2. Other simulation settings are the same as that of Section 2. Table 5 presents the throughput under 1% loss rate. By controlling the queue length, RED and PI could achieve smaller queue delay and lower packet drop rate. Thus, the throughput of RED and PI are higher than that of Drop Tail. The throughput performances of CPECN are similar for RED and PI.

5. Conclusion

In this paper we propose a novel congestion control technology, CPECN. It includes two key points: (1) adopting CP to predict network state from ECN series arrived in latest period and (2) modifying TCP behaviors when loss events occur. By using CPECN, TCP sender will trigger the rate adjustment only when network state is judged as congestion. Simulation results show that CPECN can be extended to TCP variants easily and can improve TCP throughput with lossy links. Similarly, CP can be applied into other ECN-oriented methods such as XCP to further improve their performances over hybrid networks.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors greatly appreciate the anonymous reviewers for their insightful and helpful comments. This work has been supported by NSF of China (61163060, 61103204), Open fund of Guangxi Key Laboratory of Information and Communication (PF12097x), and Guangxi Science Foundation (2013GXNSFCA019019).

References

[1] A. Gurtov and S. Floyd, "Modeling wireless links for transport protocols," *ACM SIGCOMM Computer Communications Review*, vol. 34, no. 2, pp. 85–96, 2004.

[2] Y.-J. Feng, L.-M. Sun, H.-L. Qian, and C. Song, "Improving TCP performance over MANET: a survey," *Journal of Software*, vol. 16, no. 3, pp. 434–444, 2005.

[3] S. Floyd, "TCP and explicit congestion notification," *Computer Communication Review*, vol. 24, no. 5, pp. 10–23, 1994.

[4] RFC 3168, 2007, http:// www3.ietf.org/proceedings/07mar/slides/tsvarea-3/sld6.htm.

[5] R. Paulo, 2012, http://netbsd-soc.sourceforge.net/ecn/.

[6] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *Proceedings of the ACM conference of Special Interest Group on Data Communication Conference (SIGCOMM '02)*, pp. 89–102, August 2002.

[7] Y. Xia, L. Subramania, I. Stoica, and S. Kalyanaraman, "One more bit is enough," in *Proceedings of the ACM Conference of Special Interest Group on Data Communication (SIGCOMM '05)*, 2005.

[8] B. Vamanan, J. Hasan, and T. N. Vijaykumar, "Deadline-aware datacenter TCP, (D2TCP)," in *Proceedings of the ACM Conference of Special Interest Group on Data Communication (SIGCOMM '12)*, 2012.

[9] J. Wang, P. Dong, J. Chen, J. Huang, S. Zhang, and W. Wang, "Adaptive explicit congestion control based on bandwidth estimation for high bandwidthdelay product networks," *Computer Communication*, vol. 36, no. 10, pp. 1235–1244, 2013.

[10] W. Jung, S. Son, and B. Rhee, "A study of enhanced TCP for vertical Handover using explicit congestion notification (ECN)," in *Proceedings of the 3rd International Conference on Ubiquitous and Future Networks (ICUFN '11)*, pp. 305–308, June 2011.

[11] J. Huang and J. Wang, "An ECN-Based congestion control algorithm for TCP enhancement in WLAN," in *Proceedings of the 11th IEEE International Conference on High Performance Computing and Communications (HPCC '09)*, pp. 464–469, June 2009.

[12] M. Welzl, "Using the ECN nonce to detect spurious loss events in TCP," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '08)*, pp. 2525–2530, December 2008.

[13] J. Ye, J. Wang, J. Huang, and X. Zhang, "A cross-layer ECN to achieve fairness among TCP flows in wireless mesh networks," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '09)*, December 2009.

[14] F. Peng, S. Cheng, and J. Ma, "An effective way to improve TCP performance in wireless/mobile networks," in *Proceedings of the Information Systems for Enhanced Public Safety and Security EUROCOMM*, pp. 250–255, 2000.

[15] K. Xu, Y. Tian, and N. Ansari, "TCP-Jersey for wireless IP communications," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 4, pp. 747–756, 2004.

[16] S.-J. Bae and S. Chong, "TCP-friendly wireless multimedia flow control using ECN marking," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '02)*, pp. 1794–1799, November 2002.

[17] F. Peng, H. Alnuweiri, and V. C. M. Leung, "A novel flow control scheme for improving TCP fairness and throughput over heterogeneous networks with wired and wireless links," in *Proceedings of the IEEE International Conference on Communications (ICC '05)*, pp. 3565–3569, May 2005.

[18] M. D. Chawhan and A. R. Kapur, "TCP performance enhancement using ECN and snoop protocol for Wi-Fi network," in *Proceedings of the 2nd International Conference on Computer and Network Technology (ICCNT '10)*, pp. 186–190, April 2010.

[19] J. Ye, J. Wang, Q. Liu, and Y. Luo, "An improved TCP with cross-layer Congestion notification over wired/wireless hybrid networks," in *Proceedings of the 9th International Conference for Young Computer Scientists (ICYCS '08)*, pp. 368–373, November 2008.

[20] H. Bai, D. Lilja, and M. Atiquzzaman, "Applying speculative technique to improve TCP throughput over lossy links," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '05)*, pp. 3676–3681, December 2005.

[21] M. Sridharan, A. Durresi, C. Liu, and R. Jain, "Wireless TCP enhancements using multi-level ECN," in *Internet Quality of Service*, Proceedings of SPIE, pp. 180–187, September 2003.

[22] A. Durresi, M. Sridharan, C. Liu, M. Goyal, and R. Jain, "Congestion control using multilevel explicit congestion notificationin satellite networks," in *Proceedings of the IEEE International Conference on Computer Communications and Networks (ICCCN '01)*, 2001.

[23] V. Raghunathan and P. R. Kumar, "A counterexample in congestion control of wireless networks," *Performance Evaluation*, vol. 64, no. 5, pp. 399–418, 2007.

[24] X. Chen, S.-C. Wong, and C. K. Tse, "Adding randomness to modeling internet TCP-RED systems with interactive gateways," *IEEE Transactions on Circuits and Systems II*, vol. 57, no. 4, pp. 300–304, 2010.

[25] M. Alizadeh, A. Greenberg, D. A. Maltz et al., "Data Center TCP (DCTCP)," in *Proceedings of the 7th International Conference on Autonomic Computing (SIGCOMM '10)*, pp. 63–74, September 2010.

[26] TCP Westwood, http://www.cs.ucla.edu/NRL/hpi/.

[27] C. Liu and R. Jain, "Improving explicit congestion notification with the mark-front strategy," *Computer Networks*, vol. 35, no. 2-3, pp. 185–201, 2001.

[28] N. Xiong, A. V. Vasilakos, L. T. Yang et al., "A novel self-tuning feedback controller for active queue management supporting TCP flows," *Information Sciences*, vol. 180, no. 11, pp. 2249–2263, 2010.

[29] NS2: network simulator, 2004, http://www.isi.edu/nsnam/ns.

[30] R. Jain, D. chiu, and W. Hawe, "A quantitative measure of fairness and discrimination in shared computer systems," DEC Res.Rep.TR-301, 1984.

[31] D. Lin and R. Morris, "Dynamics of random early detection," in *Proceedings of the ACM Conference of Special Interest Group on Data Communication (SIGCOMM '97)*, 1997.

[32] C. V. Hollot, V. Misra, D. Towsley, and W.-B. Gong, "On designing improved controllers for AQM routers supporting TCP flows," in *Proceedings of the 20th IEEE Conference International on Computer and Communications Societies*, pp. 1726–1734, April 2001.