Congestion Exposure Enabled TCP with Network Coding for Hybrid Wired-Wireless Network

Hong Zhang[a], Kaiping Xue[a]*, Peilin Hong[a], Sean Shen[b]*
[a] The Department of EEIS, University of Science and Technology of China, Hefei, Anhui 230027 China
[b] DNSLAB, China Internet Network Information Center, Beijing 100190 China
*kpxue@ustc.edu.cn, sean.s.shen@gmail.com

***Abstract*—TCP with network coding (TCP/NC) makes the packet loss, which is caused by wireless transmission error, have no effect on congestion control. Current proposals prefer to use delay-based congestion control aspect(congestion avoidance phase) of TCP Vegas to deal with the congestion problem of TCP/NC. However, it is oversimplified and may lead to unfairness when both TCP flow and TCP/NC flow coexist in the congested wired bottleneck link in hybrid wired-wireless network. In this paper, congestion exposure enabled TCP/NC, named CEE-TCP/NC, is proposed to make TCP/NC be friendlier to TCP protocols in the case of congestion. CEE-TCP/NC replaces TCP's loss-based congestion indicator with a method based on analyzing gaps in the ACK stream that arrive at the TCP sender. Further, different levels of congestion can be detected and actions against congestion are taken accordingly. By theoretic analysis and simulation, we show that the scheme not only inherits the advantage of network coding to eliminate the effect of wireless transmission error, but also avoids damaging the performance of other competing flows.**

I. Introduction

TCP is a widespread protocol for reliable data delivery. However, it performs poorly in lossy wireless links because of its inability to distinguish the packet loss caused by wireless link errors("random loss" for short) from the loss caused by congestion("congestion loss" for short). Several existing schemes, such as NCPLD[1] and TCP Westwood[2], suggest changing congestion control scheme or cooperating with intermediate nodes to improve TCP performance in wireless environment.

Different from former researches, Sundararajan et al. [3] propose a new concept, TCP with on line network coding (TCP/NC), to mask loss due to bit errors from the congestion control algorithm. Numerous research works have focused on improving the performance of TCP/NC. The optimization of redundancy parameter in multihop wireless network is concerned in[4] , the universality of TCP/NC is improved in[5], and attention is paid to the models and properties of TCP/NC in[6, 7].

TCP/NC proposes an alternative approach to handle packet loss, however, as a side effect, it significantly alters TCP's congestion control. Some proposals [3, 8, 9]suggest using congestion avoidance phase of TCP Vegas [10] to deal with the congestion problem in TCP/NC. It is delay-based congestion control and the congestion window is additively increased or additively decreased(AIAD). The schemes are effective if there only exist TCP/NC flows in pure wireless network.

However, in hybrid wired-wireless network, current congestion control schemes used in TCP/NC may introduce unfairness in the congested state. A simple example is shown in Fig.1. Mobile terminal and PC download files with TCP/NC and TCP respectively. Two flows run through the same bottleneck. When packet drop due to congestion happens, TCP/NC always linearly decreases its congestion window or does not reduce it at all. However, TCP multiplicatively decreases its window or reduces it to initial size if timeout exists, which is unfair compared to TCP/NC.

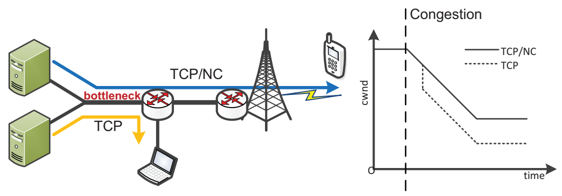


Fig. 1. Unfair scenario

The detailed reasons of the unfairness are described below.

Current congestion control schemes for TCP/NC may not detect the correct congestion level and react to it appropriately in some specific cases. Vegas differs significantly from the Reno with its adjustments of the congestion window size both in response to packet loss and to variations of the delay[11]. Adopting what kind of adjustment is depended on the congestion level. The scheme for TCP/NC is not exactly the same as TCP Vegas since it only responses to the variation of RTT and adjusts its window linearly.

In[3]and[8], it is assumed that packet loss due to congestion appears as longer RTT[1]. Whereas, the packet loss caused by congestion and the increase of delay is not closely related. The characteristic of TCP Vegas is sensitive to the number of connections. If $N\alpha < B$ is satisfied,[2] all of the flows behave exactly like TCP Reno [12]. Then the packet loss may happen before RTT exceeds upper bound. Therefore, it is unreasonable to just depend on longer RTT to expose congestion. Even though the loss indeed occurs after RTT becomes much longer, punishment to the increase of RTT in TCP/NC congestion

[1]RTT here is the duration between the time when coded packets are sent and the time when related packets are acknowledged at NC layer.

[2]N is the number of connections, α is permitted low bound of buffered packets, and B is the total buffer size in the network.

control scheme is too light to the congestion level which packet loss occurs.

Furthermore, in TCP, timeout event occurs because of insufficient duplicate ACKs or loss of retransmitted packets. However, using TCP/NC, timeout probability is extremely low or tends to zero, even though congestion is serious and loss rate is high. The reasons are as follows. On one hand, retransmission is triggered without duplicate ACK. On the other hand, even if retransmitted packets are lost, newer retransmissions are triggered timely[8, 13].

Hence, TCP/NC can neither detect the correct congestion level nor react to it rationally in the above circumstances. When TCP and TCP/NC coexist in the congested link, TCP multiplicatively decreases its window size or times out frequently while TCP/NC only decreases its congestion window linearly or keeps it unchanged. Unfairness between TCP flows and TCP/NC flows will result in low throughput of TCP flows.

Retransmission in TCP/NC may also have negative effect on congestion control. Redundant(or retransmitted) packets are transmitted immediately after receiver side detects loss[8, 13], which is aimed at reducing decoding delay. Whereas, when congestion loss happens, too much retransmissions without adjusting congestion window rationally may aggravate congestion and induce more packet loss.

These induce unfairness between the original TCP and TCP/NC in wired-wireless network when in congested state. After the network state becomes stable again, TCP/NC flows will occupy more bandwidth resources than other TCP flows because of different reactions against the same congestion level. The unfairness here means TCP/NC flows occupy more resources which should belong to other TCP flows. The objective of TCP/NC should be to make better use of available resource rather than reducing the performance of other competing flows in the same bottleneck.

Considering in hybrid wired-wireless network, we provide a new congestion control mechanism for TCP/NC, named Congestion Exposure Enabled TCP/NC (CEE-TCP/NC), to ensure TCP/NC making full and fair use of bandwidth resource in congested state. CEE-TCP/NC can detect the correct congestion level and react to it more appropriately regardless of the relationship between the congestion loss and variation of RTT.

The main contributions of the work presented in this paper are as follows:

- We propose Congestion Exposure Enabled TCP/NC. With the information in ACK, loss reason and loss rate is deduced. If congestion based, sender will decrease its congestion window size according to congestion loss rate. Otherwise, redundant packets are scheduled dynamically to cover random loss.
- We derive the theoretic congestion loss detection delay and timeout possibility, and further use simulation to prove the effectiveness and friendliness of CEE-TCP/NC.
- Despite masking random loss and retransmitting timely, CEE-TCP/NC distinguishes congestion loss from random loss without depending on the variation of RTT and reacts to the congestion level appropriately. No cooperation with intermediate nodes is required. The scheme is especially suitable for the hybrid wired-wireless network and makes TCP/NC available for a wider scope .

The rest of the paper is organized as follows. Section II provides a brief introduction to basic concept of TCP/NC. The details of CEE-TCP/NC is given in Section III. Furthermore, we use theoretic analysis to prove the efficiency in Section IV. NS2 simulation results and performance comparisons are given in Section V. Section VI concludes the paper.

II. Background

To make it easy to understand, we briefly introduce the rationale of TCP/NC in this section.

The aim of TCP/NC is to mask loss from wireless environment and keep congestion window large when no congestion occurs. In TCP/NC [3], an intermediate layer, network coding layer (NC layer), is inserted between transport layer (TCP layer for short) and IP layer as shown in Fig.2. When new traffic arrives at TCP layer, it is segmented and passed to NC layer, and then NC layer combines several native segments with randomly choosing coefficients. This is called the coding procedure. For example, D_1, D_2, D_3 are three native segments, and $C_1 = D_1 + D_2 + D_3$ is a coded packet. With different coefficients, different coded packets are generated. The coding procedure is transparent to all the layers below NC layer. When receiver gets enough coded packets, packets can be decoded at NC layer, and native segments are transferred to TCP layer. Sender transmits redundant packets to cover random losses in wireless channel. The number of redundant packets is decided by redundancy parameter R. Usually, R is larger than $\frac{1}{1-p}$, where p is the probability of wireless random loss. Hence, TCP/NC can still keep congestion window large when random loss happens, thus improving throughput significantly.

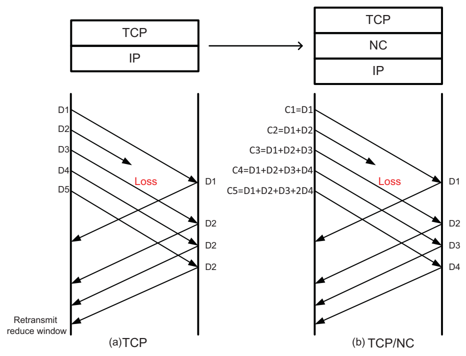


Fig. 2. Difference between TCP and TCP/NC

Acknowledgement mechanism of TCP/NC is redesigned and accomplished at NC layer in the receiver side. Considering the example in Fig.2(a), for TCP, when random loss happens, receiver sends back duplicate ACKs if the arriving packet is not the expected one. In Fig.2(b), a coded packet C_2 is lost while C_3 arrives at NC layer. As specified in TCP/NC, receiver has "seen" D_2. D_2 is said to be "seen" because D_2's information which is expected by receiver is contained in C_3 and C_3 is uncorrelated to C_1 (i.e. it reveals one unit of new information) [3]. Receiver regards it as receiving the expected packet, and sends back non-duplicate ACK. Later, redundant packet C_5 is transmitted to cover random loss. As a result, congestion window need not be reduced when there only exists random loss. Furthermore, according to [8], the number of packets, which is randomly lost, is also feedback, and can be used to control retransmission dynamically.

III. Congestion Exposure Enabled TCP with network coding

In this section, we detail our proposed congestion exposure enabled TCP/NC scheme, including retransmission on demand, congestion exposure and related control actions. We redesign recovery scheme for TCP/NC based on different packet loss scenarios. Focusing on the congestion control scheme, the goal is to make TCP/NC flows avoid damaging the performance of other TCP flows in some specific cases.

To detect loss, ACKs convey two types of sequence information, including the maximum sequence number of native TCP segment involved in coded packets, denoted by Seq_m, and maximum sequence number of "seen" packets, denoted by $Seen_m$. Let δ be the difference between Seq_m and $Seen_m$, which represents the number of lost packets. If loss happens, congestion exposure scheme is used to infer the reason. If congestion based, then congestion control scheme, mentioned in Section III.B, is triggered to decrease congestion window size according to the congestion level(i.e. loss rate). Otherwise, redundant packets are scheduled dynamically to cover wireless random loss only, see Section III.A.

A. On-demanded retransmission scheme

Packet loss due to wireless link error is not exactly periodical and burst packet loss might happen [14, 15]. Taking advantage of retransmission schemes in [8, 13] to reduce decode delay, we further modify them to fit our scheme. Retransmitted(or redundant) packets are scheduled dynamically within the scope of redundancy parameter. Note that, retransmitted packets and redundant packets are the same type of packets here because they are both used to cover losses. Below are the key points of the scheme, which is different from the aforementioned ones:

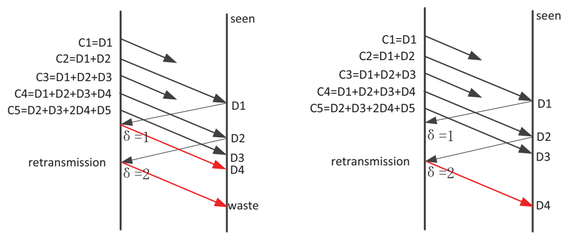
- Retransmission on demand is triggered when there exists burst losses.
- The number of retransmitted packets is restricted to the redundancy parameter R, which is only used to cover random loss.

The reason to limit the number of redundant packets is that wireless loss probability is low and relatively stable in long term, even though burst noise may cause loss rate changing temporarily, especially in the scenario illuminated in Fig. 1. Wireless random loss can be covered by a reasonable redundancy parameter R, which gives a guidance towards average loss gap and restricts the account of retransmitted packets for wireless random loss. The loss gap size(lossless duration) is the amount of successfully transmitted packets between two loss events. The details of loss gap size is in section III.B.

The retransmission timing is decided by *Redundancy quota* and δ. Redundancy quota is initialized as 0. Transmitting new coded packet increases redundancy quota by $(R-1)$; on the contrary, retransmission due to wireless loss consumes redundancy quota by 1.

If the δ is more than one and lager than the last recorded one(denoted by δ_{old}), which means more packet losses(or burst loss) happen, sender need to retransmit one oldest undecodable segment per-round apart from sending new coded packets. This is called retransmission on demand. If the quota equals or exceeds 1(i.e. quota is full), a redundant packet is sent to cover potential loss.

Burst loss will consume much redundancy quota, and then plenty of native coded packets should be sent before quota becomes full again. It matches the loss characteristic in wireless environment.



(a) Unreasonable retransmission (b) Reasonable retransmission

Fig. 3. Example of retransmission

What is also to be noted is that, if δ just equals to one, no packet is retransmitted until redundancy quota is full or more losses happen. It avoids unnecessary waste of redundancy quota. As shown in Fig. 3(a), ACK that announces one packet lost arrives at sender side after redundant packet C_5 has already been sent(because the quota gets full). If sender retransmits immediately, then the second retransmitted packet is waste because C_5 has covered the loss already. On the contrary, if sender waits for more information of packet loss, shown in Fig. 3(b), then retransmissions exactly cover losses.

The on-demand retransmission scheme in this section is only aimed at dealing with wireless random loss. If the loss is caused by congestion, retransmission does't consume redundancy quota and congestion control action is taken.

B. Congestion Exposure and Congestion Control

The new congestion control scheme focuses on the responses to packet loss. It contains two steps. Firstly, when packet loss happens, sender uses congestion exposure scheme to infer the reason of loss. Secondly, if the loss is congestion based, sender will decrease its congestion window according to the congestion level.

1) Congestion Exposure: Different from the schemes that infer the reason of loss from RTT[8, 9], we propose that congestion may happen if sender detects continuous small loss gaps (usually 3 times as backward-compatible with TCP which use 3 duplicate ACKs as loss indicator). Pseudo-code 1 shows how sender abstract loss gap size from ACKs.

```
Pseudo-code 1.Congestion Exposure Scheme
Note: $Seq_{old}$ and $Seen_{old}$ is the last recorded one. $rtx$ is the number of retransmitted packet in lossless round. $small\_gap$ is the number of continuous small loss gaps.
$lost = (Seq_m - Seq_{old}) - (Seen_m - Seen_{old})$
$loss\_gap = \frac{Seq_m - Seq_{old} + rtx}{lost}$
if $loss\_gap < ave\_gap$
    $small\_gap = small\_gap + lost$
else
    $small\_gap = 0$
end
if $small\_gap \geq 3$
    congestion is true
end
```

Packet loss gap is an effective approach in estimating current transmission state[16]. If the loss is caused by wireless transmission error, the gap seems large. On the contrary, if the reason is congestion, the gap is much smaller. The reason is that, when a packet is lost due to congestion, all packets sent shortly after this packet will be lost with a great probability. This high loss correlation among packets could be motivated by a droptail queue scheme[17]. To avoid the effect of burst loss in wireless links, we adopt 3 continuous tiny loss gaps as an indicator of congestion loss empirically.

```
Pseudo-code 2. Sender side of CEE-TCP/NC
Note:$quota$ is redundancy quota and $cwnd$ is the congestion window size
1) Receive ACK
   Update $Seq_m$ and $Seen_m$
   Use congestion avoidance(AIAD) to tune cwnd
   if $\delta > 0$ and $\delta_{old} < \delta$
      Use Congestion Exposure Scheme
      if congestion loss happen
         Adjust $cwnd$
         Retransmit packet
      end
      if no congestion
         if $\delta == 1$
            Wait for redundant packet to cover loss
         else if $\delta > 1$
            Retransmit packet
            $quota = quota - 1$;
         end
      end
   end
2)TCP Layer generate new segment
   $quota = quota + (R - 1)$
   Generated coded packet
   if $quota \geq 1$
      $quota = quota - 1$;
      Generated redundant packet
   end
```

Monitoring the network state, if new loss happens, sender records it, abstracts the loss gap size and compares the size with the average loss gap size which can be estimated by $(\frac{R}{R-1} - 1)$. R is the redundancy parameter decided by sender. When NC layer captures 3 successive small loss gaps, it triggers congestion alert at TCP layer.

2) Congestion Window Adjustment: In most TCP protocols, when congestion losses happen, congestion windows will be either multiplicatively decreased or reduced to initial size. To keep consistence with TCP, it is necessary to expose the congestion level and reduce the congestion window accordingly. The congestion level is deduced by delay and congestion loss rate, and especially depended on the later one when congestion loss occurs. Congestion avoidance(AIAD) will be adopted if only RTT varies. When congestion loss happens, considering that temporal packet loss rate is inversely proportional to the loss gap size, congestion window is adjusted according to equation(1), where $loss\_gap(t)$ is the current loss gap size, K and b are constants, and $K \cdot loss\_gap(t) \leq 1$. It is assumed that the initial window size is 2.

$$cwnd(t + \Delta t) = \min(2, K \cdot loss\_gap(t) \cdot cwnd(t) + b) \quad (1)$$

The congestion window is adjusted in this manner because we need to consider both duplicate ACK event and timeout event of TCP flows when TCP and CEE-TCP/NC coexist.

The complete scheme of sender side is illustrated in Pseudo-code 2. At receiver side, it only need to be modified to encapsulate two types of sequence information, including the maximum sequence number of native TCP segment involved in coded packets and the maximum sequence number of "seen" packets.

IV. Theoretical Analysis

It is not always effective to differentiate congestion loss from random loss by the variation of RTT as mentioned in section I. In this section, we analyse the congestion loss detection delay for TCP/NC and CEE-TCP/NC in the case that packet loss is unrelated with the variation of RTT. We make a comparation between them to prove that CEE-TCP/NC can archive better performance. We also analyze the relationship between packet loss rate and timeout probability to prove that it is necessary to adjust congestion window according to loss rate, which makes CEE-TCP/NC react to the congestion level appropriately. Some necessary notations are given here.

Seq_m	Max sequence number involved in the coded packet
$Seen_E$	The sequence number of the expected packet
w	Congestion window size
W	Coding window size
R	Redundancy parameter
p_r	Random loss rate
p_c	Congestion loss rate
$Delay$	Congestion loss detection delay (i.e. the number of packets sent since congestion loss occurred)
P_{TO1}	The probability of timeout caused by insufficient duplicate ACKs
P_{TO2}	The probability of timeout caused by retransmission error

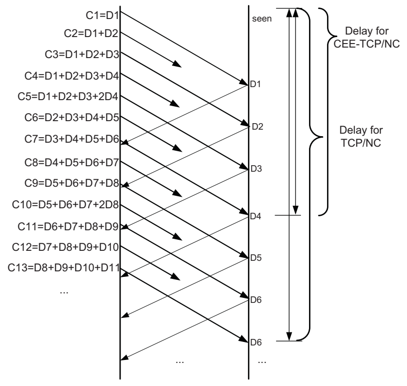


Fig. 4. Congestion Detection Delay

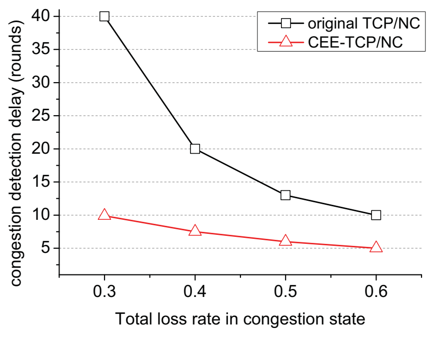


Fig. 5. Comparison of congestion delay detection

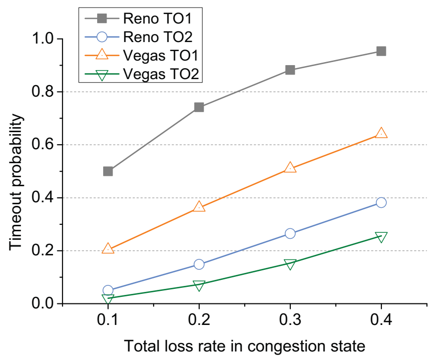


Fig. 6. Timeout probability (w=4)

A. Theoretical Analysis of Congestion Loss Detection Delay and Timeout Occurrence Possibility

1) Congestion Loss Detection Delay Analyses: We assume a scenario illustrated in Fig. 4 in which total packet loss probability is 50%. It is also assumed that the congestion loss is unrelated with the variations of RTT. Relative sequence number is adopted for convenient, so $Seen_E$ and Seq_m is set to 0 before congestion loss first occurs.

For the original TCP/NC, e.g.[3, 8, 9], receiver sends back duplicate ACKs when $Seq_m - Seen_E = W$ is satisfied because the expected information is not included in coded packets any more. Then congestion is detected. The expected sequence number of seen packets can be decided by $Seen_E = \lceil Seq_m \cdot R \cdot (1 - p_c - p_r) \rceil$. The delay is denoted by (2). The Seq_m can be decided by (3).

$$Delay = f(p_c, p_r, R, W) = \lfloor Seq_m \cdot R \rfloor \tag{2}$$

$$\lceil Seq_m \cdot (1 - R \cdot (1 - p_r - p_c)) \rceil = W \tag{3}$$

In our scheme, the congestion exposure is depended on neither duplicate ACKs nor the variations of RTT. If sender detects 3 successive ultra small loss gaps, the loss reason can be inferred as congestion. The loss gap is $\frac{1}{p_c+p_r}$, so the theoretic congestion loss detection delay is (4).

$$Delay = \frac{3}{p_c + p_r} \tag{4}$$

On Fig. 5, the theoretic congestion loss detection delay changes with the variation of total loss rate. It includes random loss and congestion loss. Considering high correlation between drops in congestion state, the temporal packet drop rate is high within a certain period of time. This is true especially in drop tail queue scheme. The fixed coding window size is W=4 and fixed redundancy parameter is 1.25. The comparison is between the traditional TCP/NC and CEE-TCP/NC. Obviously, the theoretic delay of the traditional TCP/NC is much higher than CEE-TCP/NC. TCP/NC may never detect congestion loss if loss rate is low. However, CEE-TCP/NC can detect congestion loss before congestion becomes severer. Hence, it is proved that, CEE-TCP/NC performs better when congestion loss is unrelated with RTT.

2) Timeout Occurrence Probability Analyses : Timeout probability is related to the loss rate. We analyse timeout probability of TCP Reno and TCP Vegas under burst loss model to show that timeout probability is positively related to loss rate. For TCP Reno, referring to [18], P_{TO1} and P_{TO2} are shown in equation (5) and (6), where $p = p_r + p_c$.

$$P_{TO1} = \min(1, \frac{(1-(1-p)^3)(1+(1-p)^3(1-(1-p)^{w-3}))}{1-(1-p)^w} \tag{5}$$

$$P_{TO2} = (1 - P_{TO1}) \cdot p \tag{6}$$

Vegas is different from Reno due to the fine-grained retransmission scheme which does not have to wait for 3 duplicate ACKs. P_{TO2} for TCP Vegas is represented in equation (6) in similar way, where the P_{TO1} is changed to equation (7).

$$\begin{aligned} P_{TO1} &= \min(1, A(w,0) + \sum_{k=1}^{w} A(w,k) \cdot C(k,0)) \\ &= \min(1, \frac{p + p \cdot (1-p) \cdot (1-(1-p)^{w-1})}{1-(1-p)^w}) \end{aligned} \tag{7}$$

Here, $A(w,k)$ denotes the probability that the first k packets in w packets sent are successfully transmitted, given that one or more packets are lost. $C(n,m)$ denotes the probability that the first m packets are received and the rest $(n-m)$ packets are lost.

$$A(w,k) = \frac{(1-p)^k \cdot p}{1-(1-p)^w} \tag{8}$$

$$C(n,m) = \begin{cases} (1-p)^m \cdot p & m \le n-1 \\ (1-p)^n & m = n \end{cases} \tag{9}$$

The probability of timeout event is positively related to loss rate(the sum of congestion loss rate and random loss rate) as shown in Fig. 6 ($w = 20$). Congestion window of TCP is reset to initial size when timing out, so TCP/NC also needs to reduce the congestion window in a greater degree when congestion loss rate is high.

V. Performance Simulations

The CEE-TCP/NC protocol is simulated using ns-2. We compare the performance of CEE-TCP/NC with TCP VON[8]. TCP VON is chosen for the reason that the congestion control scheme in TCP VON is the typical method used by most versions of TCP/NC schemes. The difference between TCP VON and the traditional TCP/NC[3] is that TCP VON can retransmit timely which is not the key point in this paper. The main improvement of CEE-TCP/NC compared to TCP VON is that it can detect congestion loss timely, react to it in accordance with the congestion level and keep fair with other TCP protocol.

The topology for simulation is shown in Fig. 7, which is a hybrid wired-wireless network. The bottleneck link capacity is 4Mb/sec. The capacity of other links is 5Mb/sec, with much fewer flows running on them. The coding window size is 4 for both TCP VON and CEE-TCP/NC. Only N_3, N_4, N_5 are mobile nodes. The version of TCP protocol is TCP Vegas. We mainly concern the performance of the scheme in congested state.

A. Congestion Window Behavior of CEE-TCP/NC

The window size is recorded every 0.1 second. High-speed background flow starts at 10.1s, which induces congestion and congestion loss. Two groups of experiments are performed, represented by scenario 1 and scenario 2. Related configuration is show in Table I. Wireless random packet loss probability is 0.02, and the redundancy parameters of these two CEE-TCP/NC flows are 1.05 and 1.09 respectively.

TABLE I

Scenario	Flow ID	Protocols	Source	Destination
1	1	CEE-TCP/NC	N1	N3
	2	CEE-TCP/NC	N2	N4
2	1	TCP VON	N1	N3
	2	TCP VON	N2	N4

On Fig. 8, congestion windows do not reduce immediately after congestion starts because buffers in the network are not full at that time. CEE-TCP/NC flow(marked as“congestion aware”) perceives congestion loss and the congestion level timely and adapts its congestion window accordingly. TCP VON only linearly decreases its window with AIAD scheme, which is not fit for the current network state. After congestion ends, CEE-TCP/NC flow can increase its window continuously. However, since too many retransmitted packets are sent during congestion and the network status is worse, congestion window does not return to growth immediately after congestion ends when using TCP VON.

B. Fairness property of CEE-TCP/NC

Simulation time is set to 120s here. Congestion randomly occurs in the bottleneck link and its duration is uncertain too.

1) Throughput fairness: To prove that CEE-TCP/NC improves the congestion sensitivity of TCP/NC, we treat the average throughput in wired bottleneck as fairness index. When high-speed background flow enters, congestion, together with congestion loss, happens. Two groups of experiments are conducted. The throughput is averaged over the whole simulation time, including congestion state and non-congestion state. In each sample point, the output is also averaged. The difference among each sample point is the rate of background flow.

TABLE II

Scenario	Flow ID.	Type	Source	Destination
1	1	CEE-TCP/NC	N1	N3
	2	TCP	N2	N6
	3	Background	GW	BS
2	1	TCP VON	N1	N3
	2	TCP	N2	N6
	3	Background	GW	BS

Related flow configuration is listed in Table II, all the flows must run through the wired bottleneck link. The random packet loss probability is 0.03 in wireless links. The redundancy parameter of CEE-TCP/NC is set to 1.05. The background flow rate in two groups varies from 3290Kbps to 3440 Kbps with step of 30Kbps, which makes the network state change from non-congestion to heavy-congestion.

On Fig. 10, we compare the difference of the average throughput between TCP and TCP/NC(TCP VON or CEE-TCP/NC). When TCP and TCP VON coexist, marked as “TCP VON”, TCP VON occupies much more bandwidth than the original TCP. In contrast, the result, marked as “CEE-TCP/NC”, is much lower.

The results of these two groups nearly equal to each other if background flow rate is 3320Kbps(low-level congestion). At this point, the buffer is nearly full and congestion level is low. As the congestion level increases, the result marked as “TCP VON” increases but the result marked as “CEE-TCP/NC” keeps low. Obviously, fairness is improved dramatically, despite the slight difference of throughput between two flows which is unavoidable. It is proved that CEE-TCP/NC is friendlier to TCP flow in severe congestion environment.

Similarly, Fig. 11 shows the absolute value of the throughput in two groups(TCP-Sce “x” means TCP flow in Scenario “x”). Two benefits can be gained. Firstly, fairness is improved. Secondly, the total throughput in scenario2(i.e. TCP and CEE-TCP/NC coexist) is slightly higher than that in scenario1(i.e. TCP and TCP VON coexist). Hence, the bandwidth utilization is improved because of the fairness character of CEE-TCP/NC. What to be noticed is that, as the rate of background flow increases, the total throughput of TCP flow and TCP/NC(TCP VON or CEE-TCP/NC) flow does not change much, because packet loss probability of background flow increases with the deterioration of congestion.

2) Packet loss fairness: In this section, we study the packet loss rate of background flow. Related flow setting is shown in Table III. To make the bottleneck link congested, the adopted

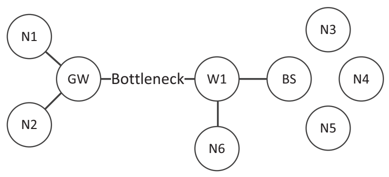


Fig. 7. Simulation Topology with Bottleneck

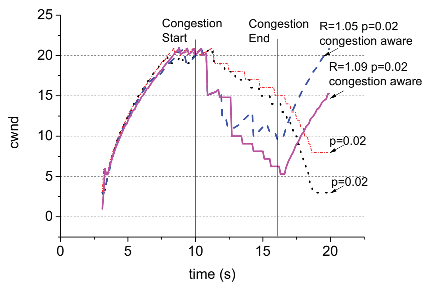


Fig. 8. Comparison of Congestion Window Size

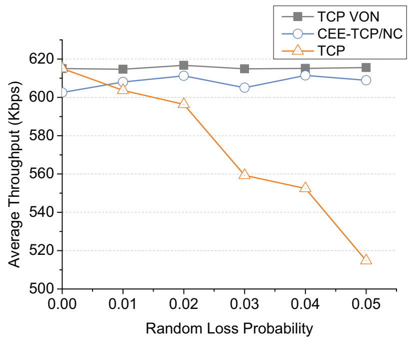


Fig. 9. Comparison among TCP, TCP/NC and CEE-TCP/NC

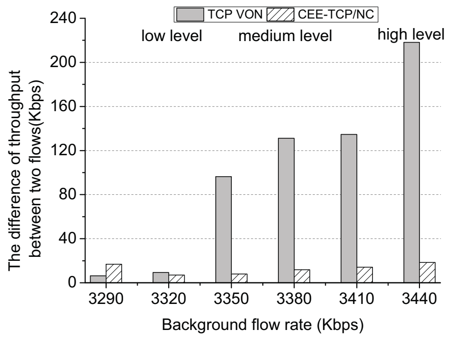


Fig. 10. The difference of throughput

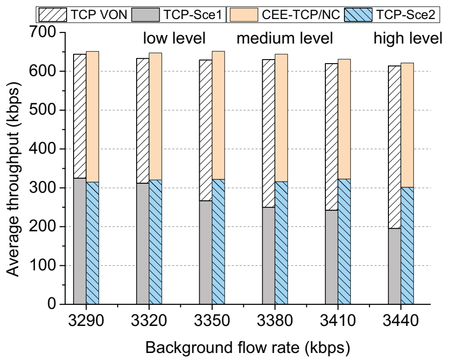


Fig. 11. Absolute throughput

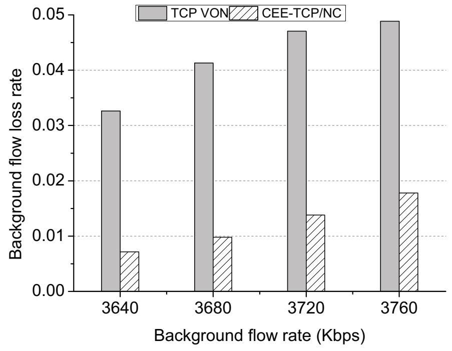


Fig. 12. Packet loss rate of background flow

rate of background flow is increased compared to the above section, because the original TCP flow is removed.

TABLE III

Scenario	Flow ID.	Type	Source	Destination
1	1	CEE-TCP/NC	N1	N3
	2	Background	GW	BS
2	1	TCP VON	N1	N3
	2	Background	GW	BS

On Fig. 12, loss probability of background flow ascends with the deterioration of congestion . Obviously, the loss rate is lower when using CEE-TCP/NC(marked as“CEE-TCP/NC”), because it can detect congestion loss and related congestion level timely.

C. Comparison among TCP, TCP/NC and CEE-TCP/NC

In this section, we conduct 3 groups of experiments. In each group, there is only one connection from N1 to N3. The redundancy parameter for CEE-TCP/NC under different loss rate is listed in Table IV.

TABLE IV

Random packet loss rate	0.01	0.02	0.03	0.04	0.05
Redundancy parameter	1.03	1.03	1.05	1.05	1.07

Fig. 9 illustrates the throughput as a function of the random packet loss probability in non-congestion environment. When in lossless case, TCP VON and TCP perform better than CEE-TCP/NC because CEE-TCP/NC sends extra redundant packets, which induces extra burden to network. As loss rate increases, both TCP VON and CEE-TCP/NC outperforms TCP.

By simulation, we show that CEE-TCP/NC can react to congestion more sensitively, decrease its window according to the congestion level in a more appropriate way, and perhaps most importantly, conform with TCP’s required congestion control behavior. If there is no congestion, both CEE-TCP/NC and TCP VON can perform better than TCP in lossy environment. However, when congestion occurs, CEE-TCP/NC performs more friendlier.

VI. Conclusion

In this paper, we propose congestion exposure enabled TCP with network coding. It detects packet loss due to congestion without depending on the variation of RTT or duplicate ACK. Further, it estimates the correct congestion level and adjusts the congestion window accordingly. By theoretic analysis and simulation, CEE-TCP/NC is proved to be more sensitive to congestion and friendlier to other TCP flows when running through congested wired bottleneck compared to former TCP/NC. Meanwhile, CEE-TCP/NC can achieve high throughput as other TCP/NC schemes in lossy environment. Consequently, it prevents TCP/NC from damaging the performance of other TCP flows. It can be applied to hybrid wired-wireless network, especially for mobile applications.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under Grant No. 61379129, No. 61170231, DSLAB Open Foundation and Intel ICRI MNC.

REFERENCES

[1] N. Samaraweera, “Non-congestion packet loss detection for tcp error recovery using wireless links,” in *IEE Proceedings on Communications*, vol. 146, pp. 222–230, IET, 1999.

[2] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, “Tcp westwood: Bandwidth estimation for enhanced transport over wireless links,” in *Proceedings of the 7th annual international conference on Mobile computing and networking*, pp. 287–297, ACM, 2001.

[3] J. K. Sundararajan, D. Shah, M. Médard, S. Jakubczak, M. Mitzenmacher, and J. Barros, “Network coding meets tcp: Theory and implementation,” *Proceedings of the IEEE*, vol. 99, no. 3, pp. 490–512, 2011.

[4] T. Van Vu, N. Boukhatem, T. M. T. Nguyen, and G. Pujolle, “Adaptive redundancy control with network coding in multi-hop wireless networks,” in *Proceedings of the Wireless Communications and Networking Conference (WCNC)*, pp. 1510–1515, IEEE, 2013.

[5] C. Chen, C. Dong, W. Hai, and Y. Weibo, “Anc: Adaptive unsegmented network coding for applicability,” in *Proceedings of the International Conference on Communications (ICC)*, pp. 3552–3556, IEEE, 2013.

[6] M. Kim, M. Médard, and J. Barros, “Modeling network coded tcp throughput: A simple model and its validation,” in *Proceedings of the 5th International ICST Conference on Performance Evaluation Methodologies and Tools*, pp. 131–140, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2011.

[7] H. J. Medina-Ruiz, M. Kieffer, B. Pesquet-Popescu, *et al.*, “Tcp and network coding: Equilibrium and dynamic properties,” in *Proceedings of the IEEE International Symposium on Network Coding*, pp. 1–6, 2013.

[8] W. Bao, V. Shah-Mansouri, V. W. Wong, and V. Leung, “Tcp von: Joint congestion control and online network coding for wireless networks,” in *Proceedings of the Global Communications Conference (GLOBECOM), 2012 IEEE*, pp. 125–130, IEEE, 2012.

[9] H. M. Ruiz, M. Kieffer, and B. Pesquet-Popescu, “Redundancy adaptation scheme for network coding with tcp,” in *Preceedings of the International Symposium on Network Coding (NetCod)*, pp. 49–54, IEEE, 2012.

[10] L. S. Brakmo and L. L. Peterson, “Tcp vegas: End to end congestion avoidance on a global internet,” *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1465–1480, 1995.

[11] J. Olsén, *Stochastic modeling and simulation of the TCP protocol*. PhD thesis, Uppsala University, 2003.

[12] T. Bonald, “Comparison of tcp reno and tcp vegas via fluid approximation,” 1998.

[13] J. Chen, W. Tan, L. Liu, X. Hu, and F. Xu, “Towards zero loss for tcp in wireless networks,” in *Proceedings of 28th International Conference on Performance Computing and Communications Conference (IPCCC)*, pp. 65–70, IEEE, 2009.

[14] A. A. Abouzeid, S. Roy, and M. Azizoglu, “Stochastic modeling of tcp over lossy links,” in *Proceedings of the 9th Annual Joint Conference on the IEEE Computer and Communications Societies*, vol. 3, pp. 1724–1733, IEEE, 2000.

[15] F. Liu, T. H. Luan, X. S. Shen, and C. Lin, “Dimensioning the packet loss burstiness over wireless channels: a novel metric, its analysis and application,” *Wireless Communications and Mobile Computing*, 2012.

[16] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, “Measurement and modelling of the temporal dependence in packet loss,” in *Proceedings of the 8th Annual Joint Conference on the IEEE Computer and Communications Societies*, vol. 1, pp. 345–352, IEEE, 1999.

[17] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, “The macroscopic behavior of the tcp congestion avoidance algorithm,” *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 3, pp. 67–82, 1997.

[18] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, “Modeling tcp reno performance: a simple model and its empirical validation,” *IEEE/ACM Transactions on Networking (ToN)*, vol. 8, no. 2, pp. 133–145, 2000.