IEICE TRANS. COMMUN., VOL.E101–B, NO.2 FEBRUARY 2018

PAPER

TCP Network Coding with Adapting Parameters for Bursty and Time-Varying Loss

Nguyen VIET HA[†a)], ***Nonmember***, **Kazumi KUMAZOE**[††b)], ***and*** **Masato TSURU**[†††c)], ***Members***

SUMMARY The Transmission Control Protocol (TCP) with Network Coding (TCP/NC) was proposed to introduce packet loss recovery ability at the sink without TCP retransmission, which is realized by proactively sending redundant combination packets encoded at the source. Although TCP/NC is expected to mitigate the goodput degradation of TCP over lossy networks, the original TCP/NC does not work well in burst loss and time-varying channels. No apparent scheme was provided to decide and change the network coding-related parameters (NC parameters) to suit the diverse and changeable loss conditions. In this paper, a solution to support TCP/NC in adapting to mentioned conditions is proposed, called TCP/NC with Loss Rate and Loss Burstiness Estimation (TCP/NCwLRLBE). Both the packet loss rate and burstiness are estimated by observing transmitted packets to adapt to burst loss channels. Appropriate NC parameters are calculated from the estimated probability of successful recoverable transmission based on a mathematical model of packet losses. Moreover, a new mechanism for coding window handling is developed to update NC parameters in the coding system promptly. The proposed scheme is implemented and validated in Network Simulator 3 with two different types of burst loss model. The results suggest the potential of TCP/NCwLRLBE to mitigate the TCP goodput degradation in both the random loss and burst loss channels with the time-varying conditions.

key words: *TCP, network coding, link loss rate, loss burstiness, Gilbert loss model*

1. Introduction

In response to the request of reliability and efficiency of wired and wireless integrated information networking, packet-loss tolerant end-to-end data transmission across lossy networks becomes more importance. The transmission control protocol (TCP) remains a dominant transport protocol for reliable end-to-end data transmission over various types of wired and wireless networks. However, in conventional TCP, even though a packet loss is caused by a lossy network, it is recognized as a sign of network congestion; thus, the sending rate is cut down. Reducing the sending rate mistakenly in the cases of non-congestion packet loss makes the goodput performance of TCP considerably degrade. The lossy networks referred in this paper are networks in which packet losses likely happen without congestion (i.e., buffer overflow at a relay node). Some examples of lossy networks include wireless links under severe conditions (e.g., the satellite network, long-distance fixed wireless backbone), wired links under severe conditions (e.g., power line communications in noisy environments) and links between mobile nodes (e.g., vehicular ad-hoc networks). In such networks, packet losses are heavy and sometimes bursty. As burst loss, consecutive packet losses or a very bad link condition lasting within a certain duration is considered. Such burst losses are not very rare events but sometimes happen in certain environments. Although it is difficult to model those phenomena of packet losses, many studies have challenged stochastic modeling on such situations.

TCP with network coding (TCP/NC) was presented [1] to address the problem of TCP in lossy networks. The term Network Coding (NC) is being used in a broader sense as coding techniques that improve the throughput, delay, and resilience. In this paper, NC is referred as a technique in which multiple original packets are combined into multiple coded packets to traverse a network. And they are decoded to obtain the original packets at the destination. In TCP/NC, the source sends the data as the random linear NC combination packets (referred to as combination packets or combinations) to the sink. Therefore, the sink is expected to recover all data using the received combinations without retransmission even though some of the combinations are lost on the way.

A new NC layer is added to the protocol stack between TCP and IP layers as shown in Fig. 1 to provide recovery ability. This layer operates transparently with TCP layer; the transport layer functionalities of TCP/NC such as congestion control and retransmission mechanism rely on those of the original TCP protocol. Thus, TCP/NC can recover the lost packets by combining Forward Erasure Correction (FEC) in NC and Automatic Repeat-request (ARQ) in TCP. By masking lost packets in NC layer, an unnecessary reduction of the sending rate (i.e., the congestion window (CWND) size) in TCP layer's congestion control is avoidable. While TCP/NC can keep a stable sending rate, it will work negatively in network congestion. Therefore, TCP/NC is considered not necessarily universal but essentially useful in certain environments with particular settings.

The original TCP/NC does not always work well with heavy, burst, and time-varying losses for two main reasons. First, there is no apparent scheme to decide and change the network coding-related parameters (NC parameters) appro-

Manuscript received January 10, 2017.
Manuscript revised June 7, 2017.
Manuscript publicized July 27, 2017.

[†]The author is with the University of Science, Ho Chi Minh City, Vietnam.
[††]The author is with the Network Design Research Center, Kyushu Institute of Technology, Iizuka-shi, 820-8502 Japan.
[†††]The author is with the Graduate School of Computer Science and Systems Engineering, Kyushu Institute of Technology, Iizuka-shi, 820-8502 Japan.
a) E-mail: nvha@hcmus.edu.vn
b) E-mail: kuma@ndrc.kyutech.ac.jp
c) E-mail: tsuru@cse.kyutech.ac.jp
DOI: 10.1587/transcom.2017EBP3010

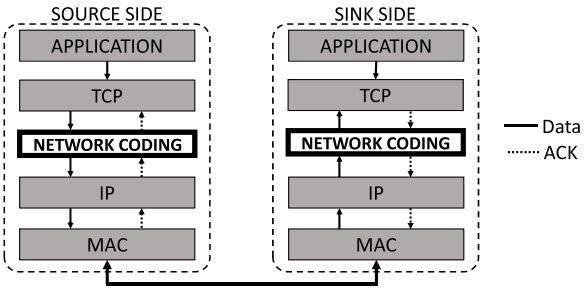


Fig. 1 NC layer in TCP/IP model.

priately to suit the diverse and changeable loss conditions such as the link loss rate and burstiness. Some studies such as SANC-TCP [2], ANC-TCP [3], TCP/NC with Dynamic Coding (DynCod) [4] and TCP/NC with Self-Adaptive Scheme (TCP/NC with SAS) [5] have an ability of automatically adjusting the NC parameters to adapt to a change of channel. However, they retained a simple baseline method or its variants to calculate the NC parameters based on the expectation (average) number of lost packets [1], which cannot always achieve the best performance. This calculation method is called the basic formula in the present paper. The basic formula only provides a way to calculate the minimum values of the NC parameters to recover the lost packets assuming random loss, i.e., the packet losses happen independently and separately. On the other hand, increasing the NC parameters from the minimum values without a proper guideline will potentially waste the link bandwidth causing the decrease of goodput. Moreover, the burst loss affects the goodput performance more severely than the random loss even at the same average link loss rate [6].

Second, the number of packet losses sometimes increases suddenly and exceeds the recovery capacity of the NC system in heavy, bursty or time-varying loss conditions. TCP layer must retransmit those lost packets triggered by triple-duplicate ACK or TCP timeout mechanism. The original TCP/NC is inefficient regarding retransmission because it can retransmit only one packet at one time. Therefore, an efficient retransmission mechanism of unrecoverable packets in TCP/NC was developed in the previous work (TCP/NC with Enhanced Retransmission (TCP/NCwER) [7]) to cope with this problem. TCP/NCwER can retransmit multiple lost packets sequentially; thus, the retransmission time can be shortened to improve the goodput performance.

In this paper, to address the first mentioned problem, a new TCP/NC variant, called TCP/NC with Loss Rate and Loss Burstiness Estimation (TCP/NCwLRLBE) is proposed. In TCP/NCwLRLBE, not only the packet loss rate but also the loss burstiness, i.e., the "average length of consecutive losses" are estimated by observing transmitted packets on the loss channel. Appropriate NC parameters are calculated from the estimated probability of successful recoverable transmission based on a mathematical model of packet losses, which is more appropriate especially in burst loss condition than the basic formula in the original TCP/NC. Moreover, TCP/NCwLRLBE introduces a new mechanism of coding window handling to update the NC parameters in coding system promptly in time-varying loss condition. Note that TCP/NCwLRLBE also adopt previously developed TCP/NCwER for efficient retransmission. Simulation evaluation will verify the significant incremental improvement in goodput by TCP/NCwLRLBE from TCP/NCwER. TCP/NCwLRLBE is partly based on the previous work ([8], [9]), but it is the essentially enhanced version of them. More precisely, they improved the goodput performance in Random loss channel and Burst loss channel based on NS3 Burst loss model, respectively. Meanwhile, TCP/NCwLRLBE uses a new algorithm to compute NC parameters based on the Gilbert loss model and is shown to perform well in Random loss channel and Burst loss channels based on different models, i.e., NS3 Burst loss model and Gilbert loss models with different burstiness.

The remainder of this paper is organized as follows. Section 2 describes the fundamental concept of TCP/NC. The proposed protocol is presented in Sect. 3. Simulation settings and results are discussed in Sect. 4, and the conclusion is outlined in Sect. 5.

2. TCP with NC Overview

2.1 TCP/NC Scheme

TCP/NC protocol was presented in 2009 [1]. It successfully implemented NC into protocol stack with a minor change by adding the NC layer between TCP and IP layer, as shown in Fig. 1. TCP/NC allows the source to send m combination packets (C) created from n original packets (p) with $m \geq n$ using Eq. (1) where α is the coefficient. If the number of lost combinations is less than $k=m-n$, the sink can recover all the original packets using the received combinations without retransmission except for the case of the linearly dependent combinations. Therefore, TCP layer is unaware of the light loss events occurring and maintains the CWND appropriately to improve the goodput performance. The processes of creating m combinations and regenerating n original packets are called encoding and decoding, respectively. In theory, the encoding and decoding processes are performed in each coding window (CW) which is a group of n original packets; thus, n is the coding window size (CWS) in packet.

The encoding process uses a simple Random Linear Network Coding (RLNC) algorithm, and the decoding process uses Gauss-Jordan elimination. All the computations are performed on a certain Galois Field, e.g., GF(2^8) to decrease the computation complexity. The adder is designed by XOR operation. The multiplier can be done by Lookup Table, which can reduce computational complexity but increases memory consumption. The decoding process requires a computational cost proportional to n^2. However, according to some results [10] in 2006, the encoding and decoding throughput are not the problems in the 1 Mbps network bottleneck when n is small. In the case that the tiny devices cannot integrate NC, e.g., with less memory and power, the TCP/NC tunneling has been proposed to convey

end-to-end TCP sessions on a single TCP/NC flow traversing a lossy network between two special gateways [11].

$$C[i] = \sum_{j=1}^{n} \alpha_{ij} p_j \; ; \quad i = 1, 2, 3, ..., m \tag{1}$$

There are two NC parameters. Redundancy factor R is the degree of redundancy that equals the ratio of m to n. The above-mentioned parameter k is called the recovery capacity in one CW (the CW recovery capacity). It is also understood as the maximum number of packet losses in each CW that can be recovered without retransmission. If the link loss rate is high, an appropriately large R can improve the goodput performance. However, a too large R incurs an unnecessary redundancy and reduces the goodput. k will be chosen based on the types of channels (e.g., random or burst loss channel), the time taken by the sink to wait for decoding (decoding delay) and the hardware limitations (e.g., processor, memory).

Besides executing the encoding and decoding processes, NC layer also allows a new interpretation of ACKs by using the degree of freedom concept and the seen/unseen definition shown below. NC layer can generate an ACK packet with the ACK number set to a sequence number of the oldest "unseen" packet that will be decoded after the new combinations being received.

Definition 1 (seeing a packet). *A node is said to have seen a packet p if it has enough information to compute a linear combination of the form $(p+q)$, where q is a linear combination itself involving only packets that arrived after p at the sender* [12].

Figure 2 is an example of the encoding, decoding and ACK packet returning processes. The packets p_1, p_2, p_3 and p_4 are encoded to the combination $C[1]$, $C[2]$, $C[3]$, $C[4]$, $C[5]$ and $C[6]$. When a new packet comes from TCP layer to NC layer, the combinations will be created and transported immediately. Due to the two lost combinations, the NC layer cannot decode any combinations until receiving the combination $C[6]$. For each received combinations, NC layer returns an ACK packet whose ACK number corresponds to the smallest sequence number of the unseen packet (e.g., sequence number of p_2, p_3, p_4 and p_5). During the process, the TCP layer is unaware of any loss events; thus, the CWND keeps increasing, and the performance is stable. In the above example, R equals $\frac{6}{4}$, k equals 2. The packets p_1 to p_4 are in the same CW. If a new packet comes e.g., p_5, it will be in the next CW.

The original TCP/NC only uses the retransmission and congestion control mechanisms of TCP layer. If the number of lost combinations exceeds the recovery capability, one or some packets are "unseen" in all received combinations in NC layer at the sink. Thus, TCP layer at the source receives the duplicate ACK numbers from NC layer and retransmits the "unseen" packets, which are simply forwarded to the lower layer.

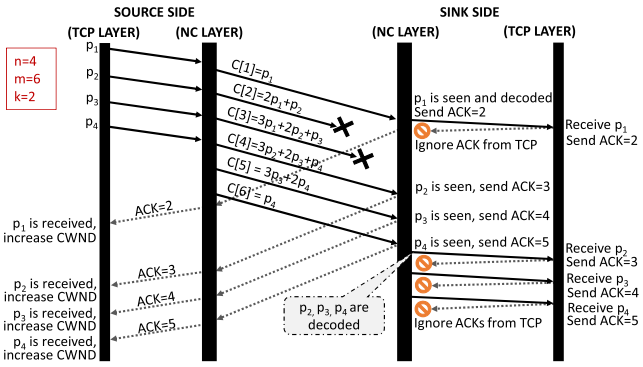


Fig. 2 NC process.

2.2 TCP/NC with Enhanced Retransmission

TCP/NC relies on TCP layer for retransmitting the packets that are not recovered by redundant combinations in NC layer. However, the original TCP/NC cannot accommodate an efficient retransmission mechanism of TCP layer such as Selective-ACK (SACK), and the lost packets are retransmitted one by one in each RTT. In response to this problem, TCP/NCwER was developed that was shown to outperform TCP NewReno, TCP TCP Westwood+, and the original TCP/NC in heavy and burst loss conditions [7].

In the bursty loss condition, repeated losses (i.e., a loss of retransmission packets) sometimes happen, which are likely to cause the TCP timeout (TO) with degrading the goodput performance [7]. All the retransmitted packets are encoded with the current NC parameters in TCP/NCwER to limit the repeated losses. Moreover, TCP/NCwER introduced a detection and retransmission mechanism for the linearly dependent combinations which cause the decoding failure at the sink. It is not explicitly handled in the original TCP/NC but essential in any real system that uses a kind of the random linear coding. In this paper, therefore, TCP/NCwER is integrated into TCP/NCwLRLBE for efficient retransmission of unrecoverable lost packets and linearly dependent combinations.

3. TCP/NC with Loss Rate and Loss Burstiness Estimation

This section describes the design of TCP/NCwLRLBE to automatically adjust appropriate NC parameters to suit the random and burst loss channels with time-varying conditions.

3.1 Estimating Link Loss Rate and Burstiness Loss Rate

First, NC layer must count the number of lost packets to estimate the link loss rate r. The additional information in NC header and NC-ACK header are proposed to use in [7]. A new Packet-id (Pid) field is added to NC header and a new Packet-id echo-reply field ($Pid\text{-}reply$), Redundancy flag ($R\text{-}flag$) is added to NC-ACK header, as shown in Fig. 3.

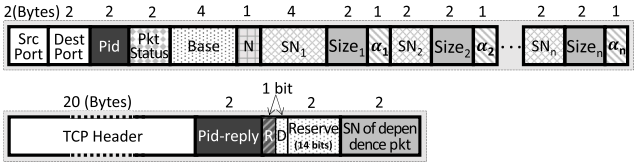


Fig. 3 NC header (above) and NC-ACK header.

Table 1 NC header fields.

Field name	Description
$SrcPort$	The source port number
$DestPort$	The destination port number
Pid	The packet identity
Pkt status	The packet status. Using for the returning ACK process
$Base$	The sequence number (SN) of the oldest packet in the NC buffer of the source. Using for buffer management at the sink
N	The number of original packet in the combination
SN_1	The SN of the first original packet
SN_n	Equal to the SN of the *n-th* packet subtract to SN_1
$Size_n$	The payload size of n-th packet
α_n	The n-th NC coefficient
$Pid\text{-}reply$	The packet identity echo reply
R	The redundancy flag
D	The dependence flag
$Reserve$	Reserved for the future use
SN of the dependence pkt	The SN of the dependence packet at the sink. Using to notify the source to retransmit this packet

Each combination has a unique Pid number interpreted as an identification number. After receiving the combination, the sink gets Pid in NC header and puts it to $Pid\text{-}reply$ in NC-ACK header of the returned ACK packet. Based on the $Pid\text{-}reply$ in the ACK packet, the source estimates r by determining the number of lost packets and dividing it by the number of sending combinations. Note that, the total bytes of NC-ACK header is only 8 bytes (including 1 byte and 6 bits for reservation); thus, a new ACK packet is 28 bytes. It is not large compared to 20 bytes of the original ACK packet or data payload size (e.g., 1000 bytes).

In contrast to the original TCP/NC, the sink of TCP/NCwLRLBE has to return an ACK packet for each received combination including the redundant combinations as well as the normal combinations. A normal combination is the combination used for decoding at the sink. A redundant combination is the combination unnecessary for decoding and ignored at the sink. The sink uses *R-flag* in ACK packet to distinguish two types of ACK, that are for normal and redundancy combinations. At the source, if the received ACK packet is for a normal combination, it will be forwarded to TCP layer. Otherwise, it will be dropped after its control information is extracted. All fields in the NC header and NC-ACK header of TCP/NCwLRLBE are briefly shown in Table 1. The fields used by the efficient retransmission mechanism of TCP/NCwER ([7]) are not explained in detail in this paper.

Second, in addition to loss rate r, the average length of the consecutive losses (the average burst loss size denoted by L) is estimated as a degree of loss burstiness. The *Pid* and the *Pid-reply* fields are used again for this purpose. Based on the received *Pid-reply*, the source can know not only the number of lost packets but also which packets were lost; thus, the length of the consecutive losses in each loss event (referred to as the burst loss size) can be inferred. L is calculated periodically by using Eq. (2), where l is the maximum of the burst loss size, N_j is the number of loss events with j consecutive packet losses, and N is the total number of loss events (the sum of N_j).

$$L = \sum_{j=1}^{l} \frac{N_j \times j}{N} \tag{2}$$

The estimated r and L are smoothed with lowering an influence of unreliable measurements by using a dynamic exponential moving average scheme as explained later in Sect. 3.3.

Note that this paper assumes the sink returns an ACK for each received combination. The source relies on those returned ACKs to monitor a loss pattern in data transmission to estimate the link loss rate and loss burstiness. Thus, TCP/NCwLRLBE requires enough ACK for stable estimation and is not assumed to adopt the delayed ACK. This drawback of the ACK-based loss estimation mechanism should be improved in the future work.

3.2 Estimating NC Parameters

Based on estimated r and L, the appropriate NC parameters (n and k) are calculated from the estimated probability of successful recoverable transmission in one CW based on a mathematical model of packet losses. Let $S(n, k, r, L)$ be this probability and C be the predefined target threshold value. n and k are computed so that $S(n, k, r, L)$ is closest greater value to threshold C within some restriction on n and k. In the simulation evaluation, the target threshold C=0.9 can achieve a high goodput performance. n is greater than 3 and limited up to 40, k is limited up to 10 to limit the encoding/decoding time. When the link loss rate is high, the estimated n (the CW size) is small. Therefore, to consider the impact on the goodput, the value of $S(n, k, r, L)$ is simply normalized to the maximum of $n + k$, i.e., m=50 (where m=n+k); thus, S'=$S^{50/m}$≥C is used instead S≥C.

In this paper, the proposed scheme will be evaluated in two burst loss channel models. The first is "NS3 loss model" which is available in the simulation software (Network Simulator 3 – ns-3 [13]) and the second is a well-known "Gilbert loss model" ([14]–[16]) shown in Fig. 4. The characteristics of NS3 loss model and the estimation of the NC parameters based on NS3 loss model were described in [9]. However, since NS3 loss model is too simple, a new method to calculate NC parameters by leveraging Gilbert loss model that is expected to approximate a wider range of conditions is developed. The NC parameters derived by the Gilbert loss model-based method are evaluated two distinct loss models (Gilbert loss model-based channels and NS3

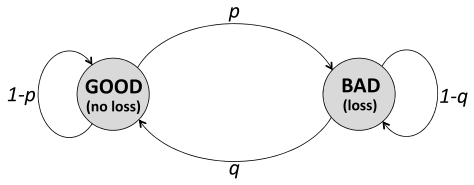


Fig. 4 Gilbert loss model.

loss model-based channels) regarding the goodput of TCP. This is a good starting point to demonstrate the potential of TCP/NCwLRLBE scheme. Note that in the loss model adopted here, the consecutive loss (i.e., bustiness) is defined not by the time duration of loss condition but by the number of packets to be lost. It means that a large consecutive loss always causes a loss of retransmission packets, which can be more severe situation compared with reality.

Gilbert loss model is determined by p and q which are the transition probabilities from "Good" state to "Bad" state and vice versa. These probabilities can be calculated based on r and L as Eq. (3) where r is the loss rate, and L is the average burst loss size.

$$p = \frac{q \times r}{1 - r} \qquad \text{where } q = \frac{1}{L} \tag{3}$$

To investigate the statistical property of the number of lost packets in one CW, the states (F, S, C) and their state transition probabilities are considered, where F, S are the number of failure sending (i.e., lost packets), success sending in one CW and C is the current state (Good or Bad). Therefore, in one CW, the number of all states is equal to $(m{+}1)(m{+}2)$; where $m{=}n{+}k$. The probabilities of state transition from state (f, s, c) to state (f, s, c), $(f, s{+}1, c)$, $(f{+}1, s, c)$, $(f, s{+}1, {\sim}c)$ and $(f{+}1, s, {\sim}c)$ are defined in Eqs. (4–8), as shown below. Note that ${\sim}c$ is the inverse state of c where $c{=}1$ is a Good state and $c{=}0$ is a Bad state. The simple example of the evolution of state transition is shown in Fig. 5.

$$\Pr[(f,s,c) \to (f,s,c)] = \begin{cases} 1 & \text{if } f{+}s = m \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$\Pr[(f,s,c) \to (f,s{+}1,c)] = \begin{cases} 1-p & \text{if } f{+}s{+}1 \le m; c = 1 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$\Pr[(f,s,c) \to (f{+}1,s,c)] = \begin{cases} 1-q & \text{if } f{+}s{+}1 \le m; c = 0 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$\Pr[(f,s,c) \to (f,s{+}1,{\sim}c)] = \begin{cases} q & \text{if } f{+}s{+}1 \le m; c = 0 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

$$\Pr[(f,s,c) \to (f{+}1,s,{\sim}c)] = \begin{cases} p & \text{if } f{+}s{+}1 \le m; c = 1 \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

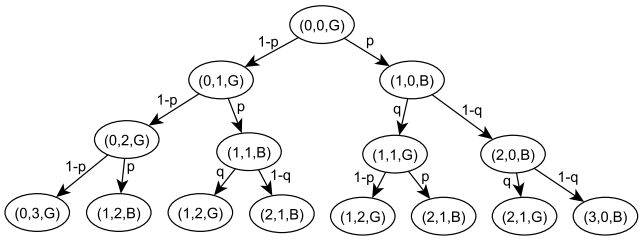


Fig. 5 Evolution of the state transition with m=3 and l=2.

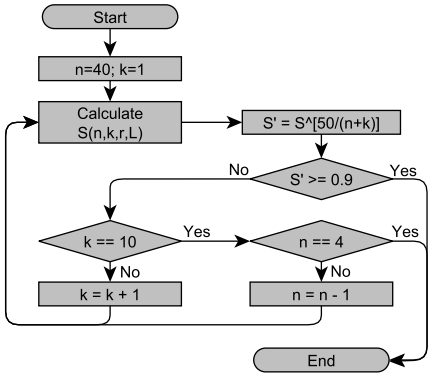


Fig. 6 Estimating n and k based on $S(n, k, r, L)$.

Let Q be the transition probability matrix with the dimension of $(m{+}1)(m{+}2)$, which is built using Eqs. (4)–(8). And let $V_{ini}[(f, s, c)]$ be the existence probability of state (f, s, c) at the beginning of CW. Since each CW starts with $(f, s, c){=}(0, 0, 1)$ definitely, $V_{ini}[(0, 0, 1)]{=}1$ and $V_{ini}[(f, s, c)]{=}0$ for any $(f, s, c){\neq}(0, 0, 1)$, that is, vector $V_{ini}{=}(1, 0, 0, \ldots, 0)$ with the dimension of $(m{+}1)(m{+}2)$. At the end of CW, all $m{=}n{+}k$ packets have been sent and some of them have been lost; all possible final states are $\{(f, s, c)|f{+}s{=}m\}$. Let $V_{fin}[(f, s, c)]$ be the existence probability of state (f, s, c) at the end of CW. The final state existence probability vector V_{fin} can be computed by:

$$V_{fin} = V_{ini} \times Q^m \tag{9}$$

The probability of the number of lost packets in $(n{+}k)$ sending packets no more than k is:

$$S(n, k, r, L) = \sum_{i=0}^{k} V_{fin}[(n + k - i, i)]. \tag{10}$$

The determination process of n and k based on $S(n, k, r, L)$ is shown in Fig. 6.

3.3 Changing NC Parameter

In existing studies [2], [3], NC parameters can be changed but only at the starting of a new encoding process or after sending m combinations. In other words, the old NC parameters are still used until the current NC process finishes even if the current CW size is large. This delay of adapting NC parameters can cause a significant performance degradation

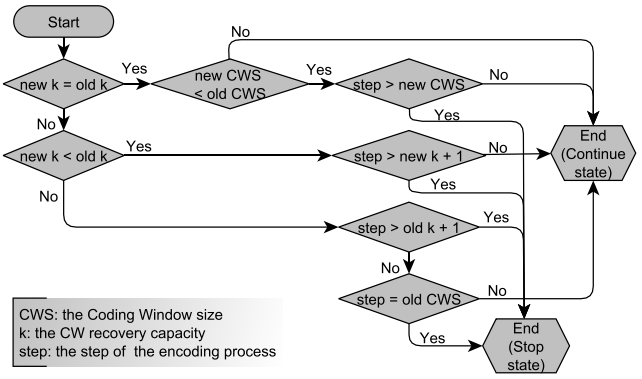


Fig. 7 Adjusting NC.

Table 2 Terminologies in Fig. 7.

Terminology	Description
old *CWS* (or k)	*CW* size (or k) used in the current encoding process.
new *CWS* (or k)	*CW* size (or k) used in the next encoding process.
step	The row number of the encoding matrix to be used as the next encoding coefficient (from 1 to m).

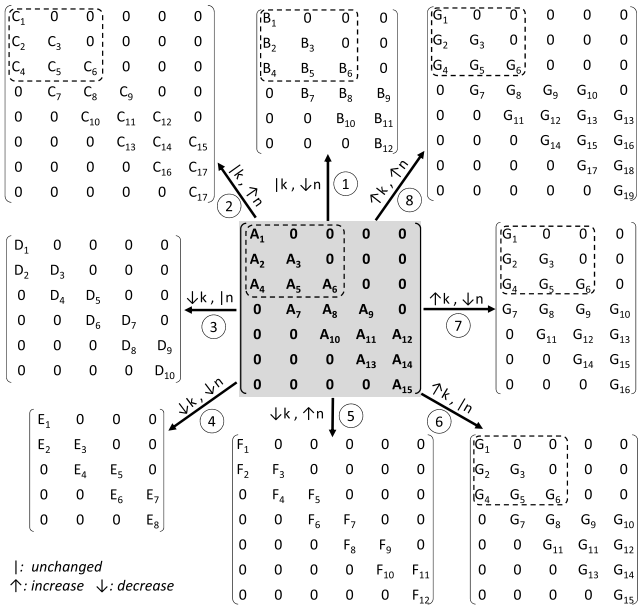


Fig. 8 Example of eight possible change patterns of NC parameters.

in some situations. For example, when the link condition becomes worse, and a TCP timeout happens due to burst loss, if the NC parameters cannot be changed in the remaining CW, packet losses in retransmissions will likely happen due to inappropriate NC parameters, resulting in TCP timeout again. Based on the previous study [8], a mechanism was proposed to adjust the NC parameters at any time without affecting the recovery ability of the current *CW*. It can determine the state to stop the current encoding process safely or continue with the current process with a new coefficient matrix. "Safely stop the current encoding process" means that the encoding process will send k combinations sequentially to retain the recovery capacity of the current process. After that, the encoding process can change to a new process with a new coefficient matrix.

Figure 7 shows the adjusting process of the NC parameters and Table 2 describes the terminologies used in this figure. Assume that at the time when the NC parameters should be updated, the encoding process already sent the $(i-1)$-th combination according to the $(i-1)$-th row of the current coefficient matrix (called $step_i$). If the $(i-1)$-th row of the current coefficient matrix and the newly updated coefficient matrix have the same structure, the encoding process can continue and apply the new coefficient matrix intermediately. Otherwise, the system has to stop the current encoding process by sending $k_{current}+1$ combinations to provide $k_{current}$ redundancy combinations. When a new original packet comes in the next time, it will belong to a new CW (a new encoding process) with the updated NC parameters. The example is shown in Fig. 8 with the current coefficient matrix in the center of the figure. There are 8 cases of changing NC parameters corresponding to 8 new coefficient matrices shown around the current matrix. Assume that the NC parameters should be changed when the fourth packet of the current encoding process comes. The system will compare the structure of three first rows of the current coefficient matrix with a new matrix. The structure is unchanged in cases 1, 2, 6, 7 and 8; in such cases, the current encoding process can continue with a new coefficient matrix and with the next step number (step 4). Otherwise, the current encoding should stop by sending two more combinations. A new encoding process will start with a new coefficient matrix. The step number will be reset to 1.

The estimated values will be updated periodically in each predetermined period (T). The number of sending packets which is used to estimate the loss rate and burstiness in each T second should be large to maintain accurate estimation. In the simulation in Sect. 4, the bandwidth of the link is 1 Mbps, and the average throughput of TCP/NCwLRLBE varies but is expected to be more than 0.3 Mbps. With the packet size of 1 KB (8 Kb), T=5 [sec] implies that M is greater than around 190 packets, which is sufficient to get the stable estimation. Therefore, T=5 [sec] is used in the simulation evaluation. In general, T should be appropriately set depending on the average throughput and the time-scale of changing channel condition. If the throughput is very low, the system is unaware of a rapid change of the link condition by this kind of passive self-monitoring.

A dynamic moving average scheme based on the Exponential Moving Average (EMA) is adopted to make the estimation of r and L robust and stable. Equation (11) calculates the averaged value of r and L where e and $\hat{e}$ are the instant and the averaged value, respectively. Weight α should be chosen for a good balance in fast adaptation to time-varying conditions (by a large α) and robustness to unreliable and inadequate estimations (by a small α). The number of packets used for estimation changes over time because the sending rate often changes due to the congestion

control of TCP; the estimated r and L become unreliable when the number of packets is small. Therefore, α is proposed to be changed dynamically based on a comparison between the averaged value ($\hat{M}$) and the instant value (M) of the number of packets which is currently used to estimate r and L. Equation (12) determines α starting with the initial α of 0.3. $\hat{M}$ is again calculated based on Eq. (11) but α is fixed to 0.2. In Eq. (12), α=0.3 represents the normal case, which is often seen in practical use of EMA. Since the expected estimation error is roughly proportional to $\frac{1}{\sqrt{M}}$, when $\frac{M}{\hat{M}} \geq \frac{4}{9}$ happens, the estimation error of the instant value is likely less than $\frac{3}{2}$ times of the averaged estimation error that can represent the normal case error. We consider it within the normal case and set α to 0.3. When $\frac{M}{\hat{M}} \leq \frac{1}{9}$ happens, the estimation error of the instant value is likely more than 3 times of the averaged estimation error. We consider it in the extreme case; the contribution of an unreliable instant value should be at most $\frac{1}{3}$ of the normal case, and thus α is set to 0.1. When $\frac{1}{9} \leq \frac{M}{\hat{M}} < \frac{4}{9}$ happens, it is considered as the intermediate case and set α to 0.2.

$$\hat{e}_i = \begin{cases} \dfrac{(i-1).\hat{e}_{i-1} + e_i}{i} & \text{if } i < \left\lceil \dfrac{1}{\alpha} \right\rceil \\ (1-\alpha)\hat{e}_{i-1} + \alpha e_i & \text{if } i \geq \left\lceil \dfrac{1}{\alpha} \right\rceil \end{cases} \tag{11}$$

$$\alpha = \begin{cases} 0.3 & \text{if } \dfrac{M}{\hat{M}} \geq \dfrac{4}{9} \\ 0.2 & \text{if } \dfrac{1}{9} \leq \dfrac{M}{\hat{M}} < \dfrac{4}{9} \\ 0.1 & \text{otherwise} \end{cases} \tag{12}$$

Furthermore, an additional consideration is taken in estimating $\hat{L}$ because of the denominator in Eq. (2), i.e., the number N of loss events is sometimes tiny, resulting in an inaccurate estimation computed by a sample mean with a small set of samples. In this paper, to lower the inadequate influence of instant L obtained by Eq. (2) with a small N, a simple approach is taken. First, N should be greater than or equal to 3 to avoid a meaningless sample mean. Thus, if N<3, α is set to small (0.1) in Eq. (11) instead of using Eq. (12). Second, a single but large sample (i.e., an event of accidentally long consecutive losses) should not impact on $\hat{L}$ too much. Thus, if l (the largest number of consecutive losses in a loss event) is greater than N (i.e., N<l), α is set to small (0.1). The reason behind the second condition is as follows. Let L_2 be a new sample mean computed by excluding the single largest sample with a value of l. The relation between L in Eq. (2) and L_2 is $NL=(N-1)\times L_2+l$. Assume that the largest sample should not increase the sample mean double, i.e., $L<2L_2$, being rewritten to $(N+1)L_2>l$. To lower the impact of the most extreme case, we consider the set of loss events consisting of $(N-1)$ loss events of a single packet and one loss event of l packets. In this case, L_2=1 which leads to the simple condition $N+1>l$ to be required for normal estimation of L.

The simulation is performed to compare the estimated $\hat{L}$ by the proposed dynamic EMA and the standard EMA in the simulation settings in Sect. 4 to examine the effect of the dynamic EMA. The Root Mean Square Error (RMSE) of the estimated value $\hat{L}$ with the preset value (e.g., L_{preset}=5) is calculated over 100 simulation times to show the efficient of the proposed method. RMSE is 0.74 in the proposed method while it is 0.84 in the standard EMA method (α is constant), which is expected to result from the elimination of extreme and unreliable observed values in dynamic EMA.

Note that these value settings are only shown to work well in the simulation conditions and could not be shown to be optimal. However, they expected to be not-so-sensitive values and can work in a wide range of conditions.

4. Simulation Results

4.1 Simulation Setup

The implementation of TCP/NCwLRLBE is accomplished by using ns-3. The topologies of the simulation consist of a backbone with four tandemly arranged routers and the sources and the sinks on either side of the backbone. One source and one sink are used to evaluate the performance in lossy networks shown in Fig. 9. Five sources and five sinks are used to evaluate the performance of network congestion shown in Fig. 10. All links have a bandwidth of 1 Mbps and a propagation delay of 5 ms. The buffer size of the links is set to 100 packets. The TCP protocol type is NewReno, and the payload size is 1000 bytes.

Two types of the loss channel are used. First is the random loss channel that causes the independent and separate losses. Second is the burst loss channel that causes the bursty losses. To evaluate the proposed method in diverse conditions of burst loss channels, NS3 loss model, and Gilbert loss model are used. In the NS3 loss model, under a given link loss rate, the number of consecutive packet losses in one loss event is chosen randomly from 1 to the max burst loss size l. In the Gilbert loss model, the max burst loss size cannot be used to control the burst loss size; thus, the average burst loss size (L) is used under a given link loss rate. To compare two types of loss model with a "similar" setting, L of Gilbert burst loss model is calculated based on l of NS3 loss model under the same link loss rate. L is chosen so that the average burst loss size of Gilbert loss model is equal to that of NS3 loss model with l, i.e., $L=\frac{l+1}{2}$.

The simulations are run at least 20 times to obtain the average value. In the case of the variance among the simulation times is small, i.e., in random loss channel or burst loss channel with a small burst loss size (l=3 or L=2), 20 iterations is enough. In the case of the variance among the simulation times is large, i.e., in burst loss channel with a large burst loss size (l=9 or L=5) or network congestion condition, the simulation is run in 100 times to increase the accuracy of averaged results. The NC parameters are calculated and adjusted every 5 seconds according to the methods described in Sect. 3.

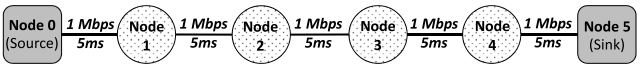


Fig. 9 Simulation topology in lossy network.

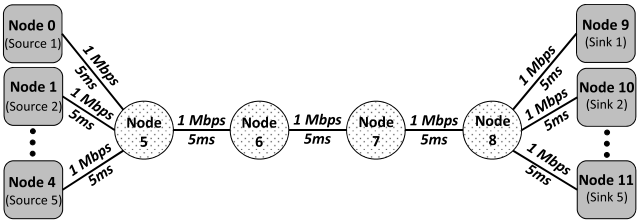


Fig. 10 Simulation topology in network congestion.

Table 3 NC parameters of TCP/NCwER.

Terminology	For channel	*R*	*n*	*k*
TCP/NCwER-1	Random loss	1.11	27	3
TCP/NCwER-2	Random loss	1.25	20	5
TCP/NCwER-3	Burst loss	1.12	40	5
TCP/NCwER-4	Burst loss	1.25	40	10

4.2 Goodput Performance in Constant Channel

In this simulation, the topology in Fig. 9 is used. The link loss rate r and the loss burstiness L are constant in all simulation time. All packet losses occur on the link connected to the source and the router. The simulation will finish when 100 Mbytes data is transferred successfully. Three protocols including TCP NewReno, TCP/NCwER, and TCP/NCwLRLBE are used for evaluation. Since TCP/NCwER uses the static NC parameters, four cases of the parameters are used. Note that the redundancy factor R of TCP/NCwER is calculated using the basic formula [1], i.e., $R=\frac{1}{1-r}$ where r is chosen to equal 0.1 and 0.2. k is chosen to equal 3 and 5 in random loss channel, 5 and 10 in burst loss channel as shown in Table 3.

4.2.1 Random Loss Channel

Figure 11 shows the goodput comparison in random loss channel. The goodput performance of TCP NewReno is reduced quickly when the link loss rate is increased; thus, TCP NewReno is almost useless in heavy loss conditions. TCP/NCwER sends too much redundancy combinations at a small link loss rate but too few redundancy combinations at a high link loss rate because it uses the static NC parameters. Thus, it is only useful in a few cases of link loss rate around at 0.05 for TCP/NCwER-1 and 0.1 for TCP/NCwER-2. The results also show a backward of the basic formula of calculating the NC parameters as follows. Although TCP/NCwER-1 and TCP/NCwER-2 are tuned with NC parameters expecting to work well at the target link loss rates of 0.1 and 0.2, respectively, the actual performance begins to decrease at the link loss rates (0.05 and 0.11, respectively) less than the target ones in Fig. 11. Meanwhile, the proposed TCP/NCwLRLBE

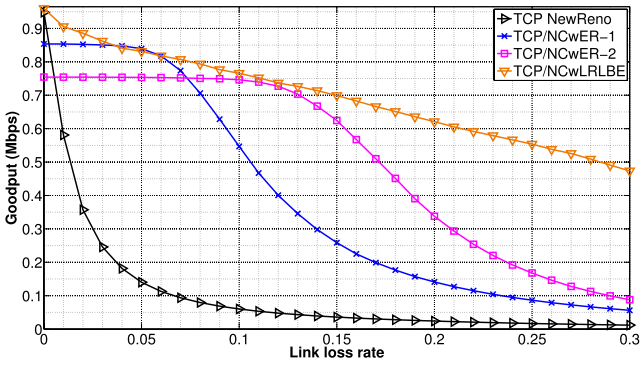


Fig. 11 Goodput comparison in random loss channel.

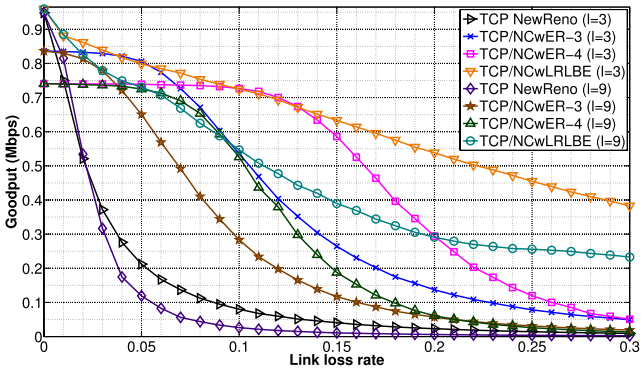


Fig. 12 Goodput comparison in NS3 burst loss channel.

can work well in all cases of link loss rate by choosing appropriate NC parameters; its goodput gradually decreases as the link loss rate increases.

4.2.2 NS3 Burst Loss Channel

In the simulation with burst loss channel, TCP/NCwER-3 and TCP/NCwER-4 are used with a large k (5 and 10). Fig. 12 shows that the goodput comparison in the NS3 burst loss channel with the max burst loss size l equals 3 and 9.

Increasing the loss rate r leads to increasing the occurrence probability that two or more burst losses happen continuously; thus, it increases the probability that the burst loss size exceeds k; the goodput performance is decreased in all protocols. Increasing the max burst loss size l also causes the same problem. In general, similar to the case of random loss channel, while TCP/NCwLRLBE can keep the goodput performance stably in all conditions of link loss rate, the goodput performance of TCP NewReno and TCP/NCwER is increased quickly at a significant link loss rate. TCP/NCwER is effective only at a few r, e.g., around 0.04 (TCP/NCwER-3), 0.11 (TCP/NCwER-4) for channel with l=3 and around 0.02 (TCP/NCwER-3), 0.07 (TCP/NCwER-4) for the channel with l=9.

4.2.3 Gilbert Channel

In Gilbert channel, besides the link loss rate r, the aver-

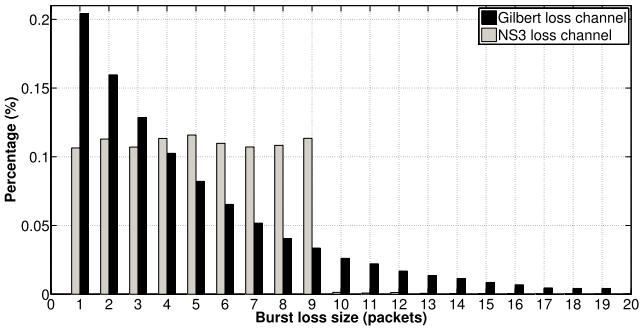


Fig. 13 Burst loss distribution at r=0.05, l=9 (for NS3 model) or L=5 (for Gilbert model).

Table 4 Occurrence probability that the burst loss size exceeds 10 at r=0.05.

l / L	In NS3 channel	In Gilbert channel
3 / 2	0.00 %	0.08 %
5 / 3	0.01 %	1.74 %
7 / 4	0.28 %	5.42 %
9 / 5	0.44 %	10.56 %

age burst loss size L is used to set the burst loss behavior. TCP/NCwER-3 and TCP/NCwER-4 continue to be used in this simulation.

The occurrence probability that the burst loss size exceeds k in Gilbert loss model is higher than that in NS3 loss model especially when L increases. The example of the burst loss distribution over 10^6 sending packets at r=0.05, l=9 (for NS3 loss model) and L=5 (for Gilbert loss model) is shown in Fig. 13. The probability that the burst loss size exceeds the maximum of k (equal to 10) in NS3 loss model is small (less than 0.5%) while this probability in Gilbert loss model is large (more than 10.5%) as shown in Table 4. The probabilities in other cases of l or L are also shown. In some exceptional circumstances, the burst loss size of Gilbert loss model can reach to 40 packets. Therefore, the Gilbert channel causes TCP timeout more often and degrades the goodput more severely than NS3 burst loss channel even with the same average burst loss size.

Figure 14 shows the goodput comparison in the Gilbert channel with the average burst loss size L of 2 and 5. The relationship among all protocols in goodput performance is similar to that in the NS3 burst loss channel or random loss channel. When L=2, TCP/NCwER is effective only at a few r around 0.04 (TCP/NCwER-3) and 0.1 (TCP/NCwER-4). The proposed TCP/NCwLRLBE can achieve the most stable performance. When L is increased to 5, the performance of all protocols is significantly decreased, but TCP/NCwLRLBE is still more stable compared to TCP/NCwER especially at r around and higher than 0.1.

Finally, Fig. 14 also shows that in some conditions with a significant burstiness and a specific loss rate, the goodput performance TCP/NCwER can be slightly better than TCP/NCwLRLBE. For example, at around loss rate r=0.05 with L=5 in Gilbert loss model, TCP/NCwER-4 slightly outperforms TCP/NCwLRLBE. In this condition,

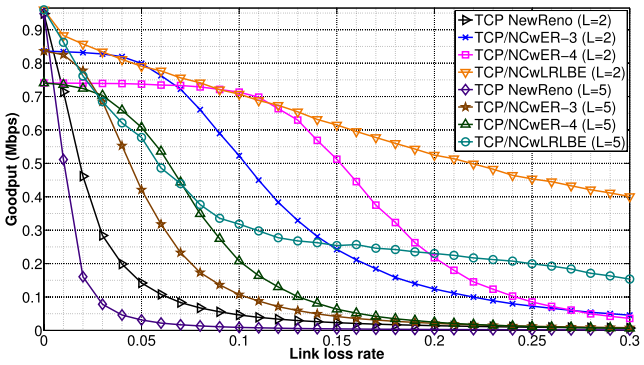


Fig. 14 Goodput comparison in Gilbert channel.

TCP/NCwLRLBE selects the pair (n, k) of $(40, 8)$ based on the threshold C=0.9 of the probability of successful recoverable transmission, while TCP/NCwER-4 uses the pair (n, k) of $(40, 10)$. Using the pair (n, k) of $(40, 10)$, the probability of successful recoverable transmission at r=0.05 is 95%, while using the pair (n, k) of $(40, 8)$, the probability of successful recoverable transmission at r=0.05 is 91%. This fact suggests that threshold C=0.9 is not necessarily optimal for goodput performance. However, using a larger threshold of the probability of successful recoverable transmission yields a more redundant transmission which wastes more link bandwidth at a high loss rate r. In this paper, choosing an optimal value of the threshold C based on multiple factors is not considered; it remains as future work.

4.3 Time-Varying Channel

According to the setting of T in TCP/NCwLRLBE, it takes at least 5 seconds to observe the channel to update the appropriate NC parameters. Besides, more time may be necessary if the channel changes significantly due to the conservative setting of EMA. Thus, the performance of TCP/NCwLRLBE will be decreased in case that the channel condition quickly changes to the worse. TCP/NCwER-3 is used in all simulations in the following section; thus, the name of TCP/NCwER is used in short. Since TCP/NCwER only achieves the best performance at a few link loss rates, its performance will be low in diverse channel conditions. Moreover, to investigate the ideal possible performance, TCP/NCwTrueLRLB (True Loss Rate and Loss Burstiness) is also evaluated. TCP/NCwTrueLRLB calculates and changes NC parameters in the same way as TCP/NCwLRLBE but knows the current true r and L being set in the simulation, which allows applying appropriate NC parameters immediately when the channel condition changes.

Two parameters are considered to simulate the time-varying channel. The first parameter refers the loss behavior called “Sequence”, which is corresponded to r and L. The second parameter indicates the interval of each loss condition (Interval). Two cases of Sequence and two cases of Interval are considered as shown in Table 5; thus, there are

Table 5 List of parameters in time-varying channel.

Seq. 1		Seq. 2		Int. 1	Int. 2
r	*L*	*r*	*L*	seconds	seconds
0.00	0	0.00	0	00-59	00-09
0.05	1	0.05	1	60-119	10-69
0.05	3	0.15	1	120-179	70-79
0.05	1	0.05	1	180-239	80-139
0.15	1	0.05	3	240-299	140-149
0.15	3	0.15	3	300-359	150-209
0.15	1	0.05	3	360-419	210-219
0.10	1	0.05	2	420-479	220-279
0.10	3	0.15	2	480-539	280-289
0.10	1	0.05	2	540-599	290-349

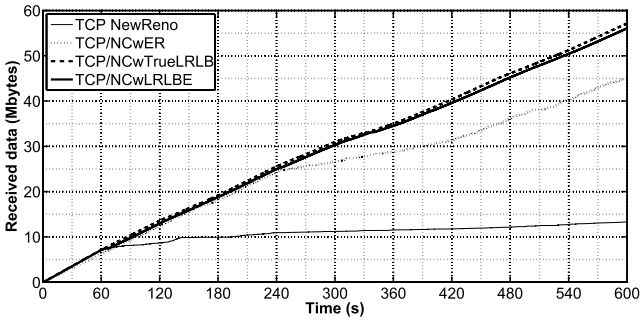
Table 6 Setting cases of time-varying channel.

Case number	1	2	3	4
Sequence	1	2	1	2
Interval	1	1	2	2

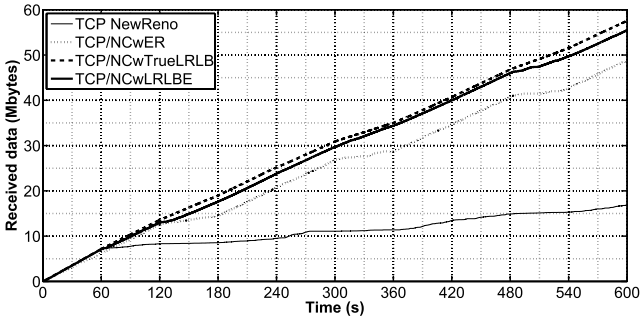
four combination cases of the time-varying channel in total presented in Table 6. The simulation results are shown in Fig. 15 that indicates the time evolution of the size of successfully received data in the application layer at the sink.

In general, in the number of received bytes, TCP/NCwLRLBE is slightly smaller than TCP/NCwTrueLRLB while TCP/NCwER and TCP NewReno are much smaller. In Case 1 and Case 2, the time interval is set to 60 seconds. This interval is much larger than the estimation time of TCP/NCwLRLBE; thus, the performance reduction in entering an 60 second period with a worse condition can be recovered at an early stage of the period. For example, in Case 2 (Fig. 15(b)), the performance is decreased when the link loss rate is increased from 0.05 to 0.15 at 120-th, 300-th, and 480-th seconds, which is represented by a gentle gradient in the graph (i.e., a degraded goodput). After that, although this gentle gradient of TCP/NCwER continues for 60 seconds, the gradient of TCP/NCwLRLBE is quickly resolved as expected and can follow that of TCP/NCwTrueLRLB.

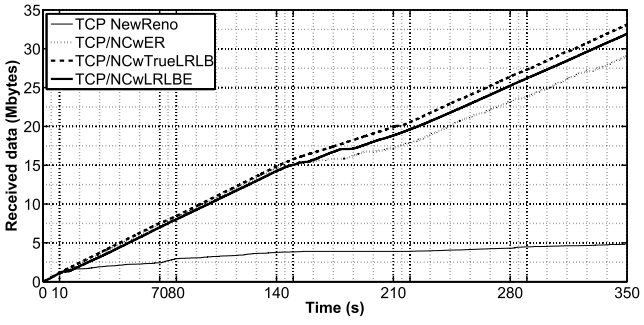
The periodical changes in a static interval might not be realistic; thus, we try to interleave the short time interval (10 seconds) and the long time interval (60 seconds) in Case 3 and Case 4. The results are shown in Fig. 15(c) and (d). Considering the channel from 140-th second in Case 3, the loss rate is increased from 0.05 to 0.15, and it continues for 80 seconds. Also, the burstiness is also increased at the 150-th second, and it continues for 60 seconds. Since TCP/NCwLRLBE requires at least 5 seconds to be aware of the change of condition and able to change the NC parameters, the goodput of TCP/NCwLRLBE is reduced for about 15 seconds from the 140-th second, but it is recovered after that. Similar behaviors of TCP/NCwLRLBE can be seen at 70-th, 150-th, and 280-th seconds in Case 4. In all cases, the results suggest that TCP/NCwLRLBE can follow the ideal performance in time-varying channels achieved by TCP/NCwTrueLRLB.



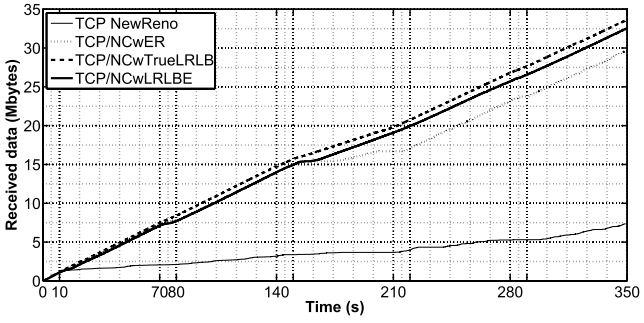
(a) Case 1



(b) Case 2



(c) Case 3



(d) Case 4

Fig. 15 Received bytes in time-varying channel.

4.4 Network Congestion

The topology shown in Fig. 10 is used to evaluate the proposed scheme with multiple sessions. Five sessions start in 10 seconds apart each. The average goodput of all sessions is counted from the 100-th second until finishing the simulation at 1100-th second. Gilbert loss model is used with

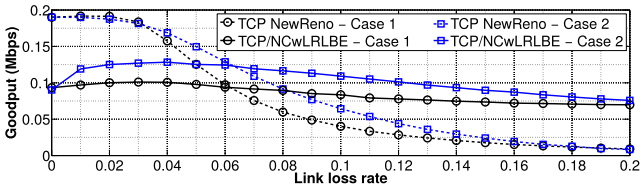
the moderate link loss rate ranging from 0 to 0.2 and L=2. There are three scenarios used for performance evaluation:

– First scenario (Case 1): all the five edge links connecting to the sources are lossy. All the five sessions run with the same protocol which is either TCP NewReno or TCP/NCwLRLBE.
– Second scenario (Case 2): the backbone link connecting the first two intermediate routers (node 5 and node 6) is lossy. All the five sessions run with the same protocol again.
– Last scenario (Case 3): two edge links connecting to the sources are lossy while the remaining three edge links are loss-free. Two sessions traversing the two lossy links run with TCP/NCwLRLBE while the three sessions traversing the three loss-free links run with TCP NewReno.

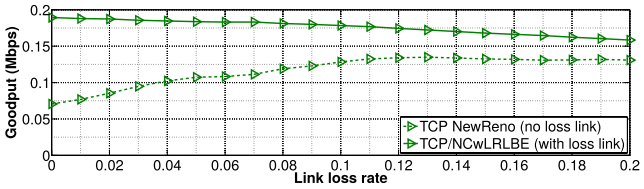
The simulation results are shown in Fig. 16. The goodput performance is averaged in all sessions which run the same protocol in the same loss condition.

First, the cases of network congestion by a competition of homogenous sessions (Case 1 and Case 2) are investigated. The completely different characteristics of TCP NewReno and TCP/NC are seen in Fig. 16(a). At a small link loss rate (less than 0.03), by a fast adaptation of the packet sending rate based on the TCP congestion control, TCP NewReno can achieve an efficient share of 1 Mbps of the backbone bandwidth by five sessions, that is 0.2 Mbps each. However, as the link loss rate increases, the average goodput of TCP NewReno decreases rapidly due to too much reduction of the packet sending rate based on the TCP congestion control. In such conditions, no more congestion happens, and the backbone link is under-utilization. Note that the average goodput in Case 2 is better than that in Case 1. This difference comes from the place where the link loss happens (the dedicated link or the shared link). In Case 1, packet losses by the link loss occur with a high probability and some burstiness in each session separately. Therefore, the performance degradation will periodically occur, resulting from a limited CWND, on each session equally. In Case 2, packet losses by the link loss happen with a high probability and some burstiness over five sessions averagely. It also lowers the loss burstiness on each session. Therefore, the performance degradation will occur on some sessions severely but some other sessions lightly at the same time. It will make it possible for some sessions have a relatively high goodput at each period, which leads to a larger total goodput over five sessions than Case 1.

TCP/NCwLRLBE is designed to recover the packet losses as much as possible without retransmission by sending the redundant combinations adapting to the link loss rate; thus, it can keep the goodput performance stably around 0.1 Mbps even if the link loss rate is high. However, the ability of masking packet losses is a double-edged sword; it makes TCP/NCwLRLBE insensitive to congestion and damages a fast adaptation of the packet sending rate. Therefore, TCP/NCwLRLBE is not very efficient with congestion, and the average goodput of TCP/NCwLRLBE is lower than



(a) Case 1 and Case 2



(b) Case 3

Fig. 16 Goodput comparison in network congestion and lossy network (Gilbert loss channel with L=2).

that of TCP NewReno when the loss rate is moderate (less than 0.06). As in the case with TCP NewReno, the average goodput of TCP/NCwLRLBE in Case 2 is better than that in Case 1. In both cases, the results suggest that the use of TCP/NCwLRLBE instead of TCP NewReno is a better choice when the loss rate is higher than moderate. Note that the goodput can be increased if TCP/NCwLRLBE can distinguish the type of packet losses caused by lossy networks or congestion [17] to decide the best NC parameters. This work will be one of the future works.

Second, we investigate the case of network congestion by a competition of heterogeneous sessions (Case 3) where two sessions of TCP/NCwLRLBE traversing lossy links and three sessions of TCP NewReno traversing loss-free links share the backbone bandwidth. This scenario is motivated by the fact that TCP/NCwLRLBE is better in heavily 'lossy links and TCP NewReno is better in slightly lossy links. In Fig. 16(c), the average goodput of the TCP/NCwLRLBE can be kept stably with a higher performance (larger than 0.16 Mbps) compared to Case 1 and Case 2 in Fig. 16(a). The total goodput performance of five sessions in Case 3 at link loss rate of 0.1 and 0.2 are 0.84 Mbps and 0.77 Mbps, respectively, which indicates a relative good link utilization. Meanwhile, in Case 1, with the same link loss rates, if all sessions run TCP NewReno, the total goodput are 0.20 Mbps and 0.05 Mbps, respectively. If all sessions run TCP/NCwLRLBE, the total goodput is 0.38 Mbps and 0.43 Mbps, respectively. In Case 2, the total goodput by TCP NewReno are 0.42 Mbps and 0.35 Mbps, respectively; the total goodput by TCP/NCwLRLBE are 0.55 Mbps Mbps and 0.38 Mbps, respectively. The results suggest that the combinational use of TCP/NCwLRLBE and TCP NewReno is a possible choice when those sessions traversing either lossy links or loss-free links compete on a loss-free but bottleneck link.

4.5 ACK Loss Evaluation

In TCP/NCwLRLBE, the source relies on returned ACKs to

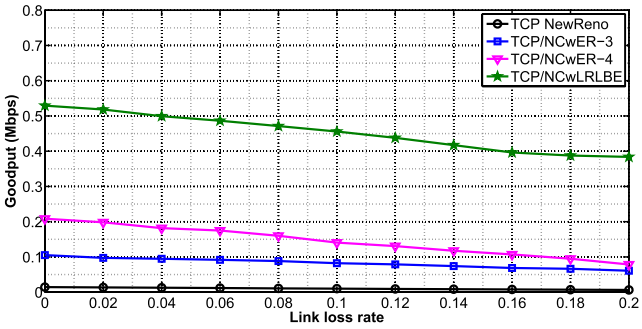


Fig. 17 Goodput comparison with link loss rate of data direction equals 0.2.

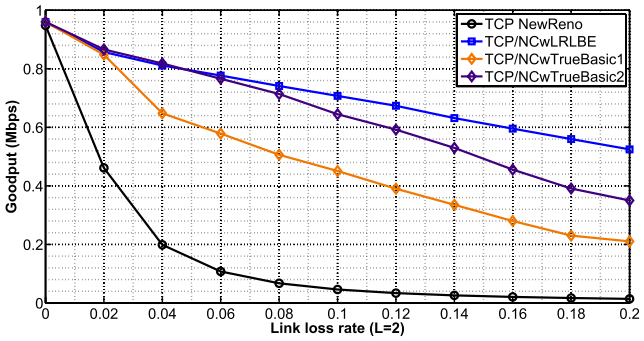


Fig. 18 Goodput comparison in Gilbert loss channel with L=2.

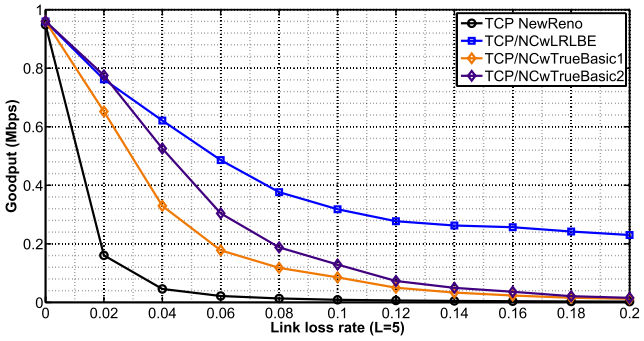


Fig. 19 Goodput comparison in Gilbert loss channel with L=5.

observe a loss pattern in data transmission to estimate the link loss rate and loss burstiness. If an ACK is lost, the source just counts it as a combination (DATA) packet loss, resulting in an overestimation on the link loss rate and loss burstiness. Thus, the number of redundancy packets increases, resulting in goodput degradation.

Figure 17 shows the impact of ACK loss on goodput degradation of NewReno, TCP/NCwER-3, TCP/NCwER-4, and TCP/NCwLRLBE in Gilbert channel with r=0.2 and L=5. The goodput performance of all protocols decreases when the link loss rate of ACK direction (r_{ACK}) increases. The lost ACK packet can slow down the retransmission process, e.g., the source must wait a long time for Triple duplicate ACK, or it can cause TCP timeout. Thus, the goodput performance is decreased in all protocols. On another hand, TCP/NCwLRLBE suffers from an incorrect estimation of link loss rate as well as loss burtiness. Thus, R is increased mistakenly. In this simulation, R increases from 1.7 in loss-free case (r_{ACK}=0) up to 2.4 in heavy-loss case (r_{ACK}=2.0 that is equal to r). However, TCP/NCwLRLBE can still outperform NewReno and TCP/NCwER even in this situation.

4.6 NC Parameter Calculation

As mentioned in Sect. 1, existing studies essentially rely on the basic formula and its variants. Although each paper proposes a distinct complicated algorithm, they all produce the redundancy factor $R=\frac{(n+k)}{n}$ that converges to $\frac{R_{ini}}{(1-r)}$ when r does not change, where R_{ini} is the preset initial value of R. In addition, they assume a fixed n and change k based on calculated R either explicitly or implicitly. Meanwhile, the proposed TCP/NCwLRLBE determines n and k based on the probability of successful recoverable transmission. Figure 18 and Fig. 19 show the goodput comparison among TCP/NCwLRLBE, TCP NewReno, TCP/NCwTrueBasic1, and TCP/NCwTrueBasic2, in Gilbert loss channel with the average consecutive loss (L) of 2 and 5, respectively. TCP/NCwTrueBasic1 calculates R based on the basic formula with R_{ini}=1 where preset loss rate r is known by the protocol, and TCP/NCwTrueBasic2 calculates R based on the basic formula with R_{ini}=1.05. The results show that the proposed scheme in calculating NC parameters is superior to the basic formula.

5. Conclusion

In this paper, a new variant of TCP/NC, TCP/NCwLRLBE, has been proposed to automatically adjust the NC layer behaviors for adapting to time-varying channel in both random and burst loss conditions. The link loss rate and burstiness are estimated from the continuous observation of the packet transmission between the source and the sink. A new algorithm is proposed to calculate the NC parameters based on the probability distribution of the number of lost packets and the loss burstiness in one CW, which is more appropriate than the basic formula used by existing work. Moreover, a new mechanism of coding window handling is developed to update NC parameters in the coding system promptly. The simulation results on ns-3 have shown that the proposed TCP/NCwLRLBE has the potential to outperform other protocols such as TCP NewReno, the original TCP/NC, and its variants over a certain range of lossy networks.

This work is partly supported by JSPS KAKENHI (16K00130) and KDDI Foundation.

References

[1] J.K. Sundararajan, D. Shah, M. Medard, M. Mitzenmacher and J. Barros, “Network coding meets TCP,” Proc. 28th Conf. on Comput. Commun., no.2, pp.280–288, Rio de Janeiro, Brazil, April 2009. DOI:10.1109/INFCOM.2009.5061931

[2] S. Song, H. Li, K. Pan, J. Liu and S Y R Li, “Self-adaptive TCP protocol combined with network coding scheme,” Proc. 6th Conf. on Sys. and Net. Commun., pp.20–25, Barcelona, Spain, Oct. 2011.

[3] C.Y. Cheng and H.Y. Yi, "Adaptive network coding scheme for TCP over wireless sensor networks," J. Comput. Commun. and Control, vol.8, no.6, pp.800–811, Dec. 2013. DOI:10.15837/ijccc.2013.6.26
[4] T.V. Vu, N. Boukhatem, T.M. T. Nguyen, and G. Pujolle, "Dynamic coding for TCP transmission reliability in multi-hop wireless networks," Proc. IEEE Int. Sym. on a World of Wireless, Mobi. and Multi. Net., pp.1–6, Sydney, Australia, June 2014. DOI: 10.1109/WoWMoM.2014.6918949
[5] H. Zhang, W. Yu, C. Wu, X. Hu, L. Zhao, and X. Cui, "Self-adaptive scheme to adjust redundancy for network coding with TCP," Comp. Eng. and Tech., pp.81–91, July 2013. DOI:10.1007/978-3-642-41635-4_9
[6] N.V. Ha, K. Kumazoe, M. Tsuru, "TCP Network Coding with Forward Retransmission," Proc. IEEE of Asia Pacific Conf. on Wireless and Mobile, pp.136–141, Bandung, Indonesia, Aug. 2015. DOI:10.1109/APWiMob.2015.7374970
[7] N.V. Ha, K. Kumazoe, and M. Tsuru, "TCP network coding with enhanced retransmission for heavy and bursty loss," IEICE Trans. Commun., vol.E100-B, no.2, pp.293–303, Feb. 2017. DOI:10.1587/transcom.2016EBP3101
[8] N.V. Ha, K. Kumazoe, and M. Tsuru, "Making TCP/NC adjustable to time varying loss rates," Proc. 8th Int. Conf. on Intelligent Networking and Collaborative Systems, pp.457–462, Ostrava, Czech Republic, Sept. 2016. DOI:10.1109/INCoS.2016.77
[9] N.V. Ha, K. Kumazoe, and M. Tsuru, "TCP with network coding meets loss burstiness estimation for lossy networks," Proc. 11th Int. Conf. On Broad-Band Wireless Comp., Commun. And Applications, pp.303–314, Asan, Korea, Nov. 2016. DOI:10.1007/978-3-319-49106-6_28
[10] M. Wang and B. Li, "How practical is network coding?," Proc. 14th IEEE Int. Work. on Qua. of Ser., pp.274–278, New Haven, USA, Nov. 2006. DOI:10.1109/IWQOS.2006.250480
[11] N.V. Ha, K. Kumazoe, K. Tsukamoto, and M. Tsuru, "Masking lossy networks by TCP tunnel with network coding," Proc. 22nd IEEE Sym. on Comp. and Commun. (ISCC), 6 pages, Crete, Greece, July 2017.
[12] J.K. Sundararajan, D. Shah and M. Medard, "ARQ for network coding," Proc. IEEE Int. Sym. on Info. Theory, pp.1651–1655, Toronto, Canada, July 2008. DOI:10.1109/ISIT.2008.4595268
[13] "Network simulator (ns-3)," https://www.nsnam.org/, accessed March 1, 2016.
[14] E.N. Gilbert, "Capacity of a burst-noise channel," Bell Labs. Tech. J., vol.39, no.5, pp.1253–1265, Sept. 1960. DOI:10.1002/j.1538-7305.1960.tb03959.x
[15] C. Jiao, L. Schwiebert, and B. Xu, "On modeling the packet error statistics in bursty channels," Proc. 27th Local Comp. Net., pp.534–541, Florida, USA, Nov. 2002. DOI:10.1109/LCN.2002.1181827
[16] G. Hasslinger and O. Hohlfeld, "The Gilbert-Elliott model for packet loss in real time services on the internet," Proc. 14th Conf. Measu., Model. and Eval. of Comp. and Commun. Sys., pp.1–15, Dortmund, Germany, March 2008.
[17] S. Cen, P.C. Cosman, and G.M. Voelker, "End-to-end differentiation of congestion and wireless losses," J. IEEE/ACM Trans. Networking, vol.11, no.5, pp.703–717, Oct. 2003. DOI:10.1109/TNET.2003.818187

Nguyen Viet Ha received the B.S. degree (2009), M.S. degree (2012) in Electronics and Telecommunications, from University of Science, Ho Chi Minh City, Vietnam and Ph.D. degree (2017) in Computer Science and System Engineering from Kyushu Institute of Technology, Japan. His research interests are transport layer protocols and network coding.

Kazumi Kumazoe received the B.E. degree (1993), M.E. degree (1995) and D.E. degree (2007) in computer science from Kyushu Institute of Technology, Japan. Since April 2013, she has been a visiting researcher at Network Design Research Center, Kyushu Institute of Technology. Her research interests are in various areas of computer networks, including transport layer protocols, network coding and network management. She is a member of IEICE.

Masato Tsuru received the B.E. and M.E. degrees from Kyoto University, Japan in 1983 and 1985, and then received his D.E. degree from Kyushu Institute of Technology, Japan in 2002. He has been a professor in the Department of Computer Science and Electronics, Kyushu Institute of Technology since 2006. His research interests include performance measurement, modeling, and management of computer communication networks. He is a member of the ACM, IEEE, IEICE, IPSJ, and JSSST.