Tolerance of TCP with network coding to reverse-direction packet loss and packet reordering

社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

Tolerance of TCP with network coding to reverse-direction packet loss and packet reordering

Nguyen VIET HA† and Masato TSURU†

† Faculty of Computer Science and System Engineering, Kyushu Institute of Technology
680–4 Kawazu, Iizuka-shi, Fukuoka, 820–0067 Japan
E-mail: † nguyen.viet-ha503@mail.kyutech.jp, † tsuru@cse.kyutech.ac.jp

Abstract Transmission Control Protocol with Network Coding (TCP/NC) is one of the potential proposals for improving the goodput performance of the current TCP protocol in lossy networks (e.g., wireless networks). TCP/NC uses additional sub-layer called Network Coding layer below TCP layer to handle packet losses without sensed by TCP layer. The authors introduced some variants of TCP/NC such as TCP/NCwLRLBE (TCP/NC with Loss Rate and Loss Burstiness Estimation) which can improve the retransmission and adapt to the changing of the channel. However, most versions of TCP/NC do not consider two problems, that are the bi-directional packet loss and the unordering packet receiving, which affect the goodput performance seriously. Therefore, in this paper, we investigate the goodput performance degradation by the bi-directional packet loss and the unordering packet receiving, and propose solutions. The result of our simulation on ns-3 (Network Simulation 3) shows that the proposed scheme can work well when loss happens in both directions as well as in unordering packet receiving environment compared to the TCP NewReno and our previously proposed protocol, TCP/NCwLRLBE.

Key words TCP/NC, Network Coding, Bi-directional loss, unordering packet receiving, lossy networks, ns-3.

1. Introduction

On the wide of applications, Transmission Control Protocol (TCP) is a current choice for reliable transmission due to its advantages on connection-oriented and congestion control features. However, its goodput performance of TCP severely degrades in the packet loss environments (e.g., in wireless networks). In TCP' s congestion control mechanism, a packet loss considered as a congestion signal; hence, TCP will decrease the sending rate by reducing the congestion window (CWND) for every packet loss detecting. This act is necessary for fair bandwidth sharing to all concurrent sessions running in a network. However, the packet loss is caused not only by network congestion but also a channel. Instead of reducing, CWND should be kept stable in the lossy case to overcome the temporary lousy condition of the channel. But, TCP has no method to separate the type of losses. TCP decreases the CWND mistakenly, resulting in affecting to transmission performance seriously. Some TCP variants have been proposed to overcome this issue, e.g., TCP Westwood+ [1]. Another approach is combining Network Coding with TCP (called TCP with Network Coding - TCP/NC) [2] which has more benefits than using only the TCP.

The basic idea of TCP/NC is the adding a new NC sub-layer between TCP and Internet layer shown as Fig. 1. The primary role of NC sub-layer is encoding and decoding. When sending the packet at the source, this sub-layer receives n TCP segments from the upper layer, combines them to m combination packets with $m>n$ (referred to encoding process), and forward to the lower layer. The sink is expected to recover all n original segments if the number of lost packets is no more than $m-n$ (referred to as decoding). While NC can apply on wide range area such as bandwidth optimizing, network scheduling. In our study, we focus only on its advantage on a high degree of packet loss robustness. And we focus on applying NC only at end-devices, not at the intermediate nodes due to the limitation of this paper scope.

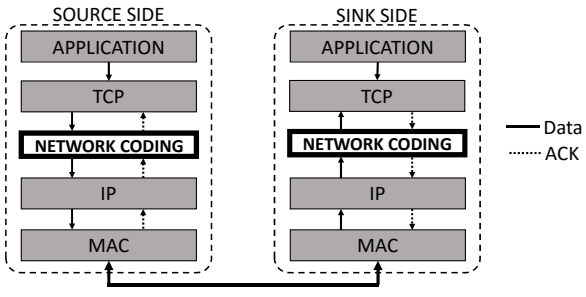


Fig. 1 NC layer in TCP/IP model

A few recent years, some variants of TCP/NC have been

introduced to improve the goodput performance. TCP/NC with Enhanced Retransmission (TCP/NCwER [3]) can improve the retransmission process by sending multiple retransmission in one Round Trip Time (RTT); besides, all the retransmission are encoding to prevent the lost again which leat to TCP Timeout (TO). Self-Adaptive NC-TCP (SANC-TCP [4]), Adaptive NC (ANC [5]), and Dynamic Coding (DynCod [6]) focus on the channel condition estimation (link loss rate) and NC parameters adaptation (n and m) to work well in the practical channels frequently changed over time. Especially, TCP/NC with Loss Rate and Loss Burstiness Estimation (TCP/NCwLRLBE [7]) can estimate the channel condition of burst loss environments (both link loss rate and loss burstiness) and be flexible adjusting the NC parameters without effect to the current settings. However, the most mentioned protocols do not consider two channel conditions which affect the goodput performance severely. First is a bi-directional loss. Most protocols use the information of the ACK packet to determine the necessary retransmission and/or estimate the correct channel conditions to determine the proper NC parameters. Therefore, they need to receive all acknowledgment (ACK) packets. Lack of ACK causes the estimation information to be wrong. Lossing ACK often occurs in the practical channel with the bi-directional loss. Second is an unordering problem which mention to the not in-order receiving the packets. The unordering problem causes goodput performance degradation in not only the regular TCP [8] but also the current TCP/NC variants. The unordering problem causes returning many duplicated ACK at the sink resulting in reducing the CWND mistakenly and retransmitting the unnecessary packets at the source. Moreover, this problem also produces the same issue with the bi-directional loss problem when the source receives not in order ACK packets.

In this paper, we propose a new scheme which can help the sink notify to the source the status (received or lost) not only of the currently received packet but also of the previously received or the lost packets. Consequently, the source can collect all the information correctly to estimate all the necessary values even though of some ACK packets are missing. Additional, the proposed scheme can estimate the unordering conditions, such as unordering length, unordering rate, and the number of duplicated ACK to react the unordering affectation. This scheme can change the NC parameters based on not only loss/burstiness but also the unordering information. Besides adjusting the NC parameters, we introduce the Pause-ACK mechanism. Pause-ACK helps the source delay some the duplicated ACK in the determined period to wait for the proper ACK packet. This mechanism can limit the number of time of incorrectly decreasing CWND to stable the goodput performance.

The remainder of this paper is organized as follows. Sect. 2 introduces the overview of TCP/NC. Sect. 3 explains the detail of the proposed scheme. Simulation evaluation is presented in Sect. 4 and conclusion is given in Sect. 5.

2. TCP/NC Overview

2.1 Network coding in protocol stack

TCP/NC introduces a new NC sub-layer putting between TCP and network layer shown in Fig. 1. This sub-layer handles the incoming and outgoing packets from TCP and network layer, respectively. The key is this sub-layer works transparently with other layers; thus, TCP/NC can simply apply to any current devices. If NC sub-layer can recover all packet losses, TCP layer is unaware of the loss events occurring. Besides, NC sub-layer will return ACK packet with ACK number determining based on the degree of freedom and the seen/unseen definition [2] not based on the decoded or received packet. The CWND is maintained even though the combination packets have not decoded yet (will be decoded later). Thus, the transmission performance is stable through lossy channels.

2.2 Coding process

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. TCP/NC using a sliding method to combine the original packets into a combination packet with the number of combined packets in one combination packet (referred to as the sliding window) is $k{+}1$. Besides, α is selected randomly; thus, the coding algorithm is also called Random Linear Network Coding (RLNC [9]). The computation implemented in a Galois field. All operators expressed to "exclusive or" (XOR) and lookup table; hence, the complexity of computation is small to apply to the real system.

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

2.3 TCP functionality

As mentioned, TCP/NC is proposed to work transparently to other layers. And the TCP functionalities have been studied and worked stably in a long history. TCP/NC should take all these advantages. Two most important mechanisms are retransmission and congestion control. If the number of packet losses is larger than the recovery capacity of NC sub-layer, the source needs to retransmit the necessary packets. In this case, the NC sub-layer returns some duplicate

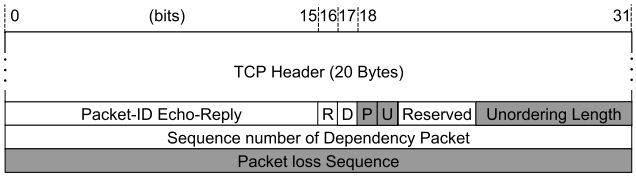


Fig. 2 NC-ACK header

ACK number equaling the oldest unseen packet. The retransmission will be started normally based on the original Triple-Duplicate-ACKs or TCP timeout. Increasing or decreasing the CWND is also controlled by TCP layers, not NC sub-layer.

3. Proposed scheme

The proposed scheme is the development of the previous study in [7] (TCP/NCwLRLBE) which can automatically estimate the channel conditions (e.g., link loss rate, loss burstiness) and adjust the NC parameters (n and m). The proposed scheme can be called the TCP/NCwLRLBE add with three mechanisms that are the ACK accumulation, unordering adaptation, and Pause-ACK. We will describe the detail of them in the following.

3.1 ACK accumulation

Calculating the estimated value immediately whenever the source receives each new ACK packet can fail if some ACK packets are lost or received not in order. Therefore we proposed to add more information in the ACK packet containing thirty-three packet status to indicate which packets received at the sink. Consequently, some ACKs loss will not affect the estimation process of the source. The source can update the estimated values over time based on the previous information stored in each ACK packets.

The NC header and NC-ACK header retained from TCP/NCwLRLBE. And the content of the header fields discussed in [3], [7]. The modification is only on NC-ACK header shown in Fig. 2 and Table. 1; hence, in this part, we focus on *Pid Echo-Reply* fields, and a new field called "*Packet loss Sequence*." *Pid* in NC header of sending combination and *Pid Echo-Reply* in NC-ACK header of the returning ACK packet is the same number which is unique. The source uses *Pid Echo-Reply* to detect whether a combination was lost or received at the sink. The Packet loss Sequence field includes thirty-two binary number presenting to the status (received or lost) of recently thirty-two consecutive packets. With this field, the sink informs to the source that not only the successfully receiving of the combination packet having *Pid* but also the status of the thirty-one previous packets. Therefore, the sink can collect all the information to estimate the channel condition even though some ACK packets are lost.

Table 1 NC-ACK header fields

Field name	**Description**
Pid-reply	The packet identity echo reply
R	The redundancy flag
D	The dependence flag
P	The Pause-ACK flag
U	The update indication 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
Packet loss Sequence	Store the status of the 32 previous packets start from the newest received packet having the *Pid* equal the *Pid Echo-Reply*.

Besides, NC header is used instead of the normal TCP header. The size of the NC header is 20 bytes for the combination containing one original packet. And five bytes is added for each additional original packet. The maximum size of NC header is 70 bytes for the k of 10. In NC-ACK header, twelve bytes is added in the normal ACK packet. The total size of NC-ACK header is 32 bytes including 20 bytes of the normal ACK header. The additional overhead is negligible compared to data payload (e.g., 536 bytes in the simulation); thus, it does not affect the goodput performance.

TCP/NCwLRLBE determines immediately the number of packet losses and the loss burstiness size (number of continuous packet losses) whenever receiving the ACK packets. These values are accumulated over time until the estimation process starts in every periodic time (configurable parameter). Therefore, losing some ACK packets affects the estimation process of TCP/NCwLRLBE. TCP/NCwLRLBE increases the number of redundancy packets mistakenly; hence the transmission performance is degraded. It is clear to see that calculating immediately channel condition is unnecessary because the system must wait until the estimation process starts. In our proposed scheme, the status of all packets in one period time is stored and updated whenever receiving the new ACK packet. Therefore, the source has enough time to update the correct status of the packet to estimate the correct channel conditions. The losing ACK packet does not affect the transmission performance if the total continuous loss in both directions (sending data packet and receiving ACK packet) does not exceed thirty-two packets.

3.2 Unordering estimation

Receiving not in order packets causes the goodput degradation as shown in the study [8]. When the sink receives leapfrog packets, it thinks some packets are lost on the channel; hence, it returns the duplicated ACKs. When the source receives too many duplicated ACKs, the TCP will enter the retransmission process and reduce the CWND by the half

(e.g., in TCP NewReno). To solve this problem, we propose to estimate the unordering length (L_u). Then, the source will the encoded combination packet containing L_u original packets. Therefore, when the sink receives the leapfrog packets, the sink can return an ascending ACK based on the "seen" original packet inside the combination packet, even though the sink cannot decode the combination packet to the original packet.

The Sink does the unordering estimation. The sink focuses on two actions that are recognizing the unordering packet and calculating the unordering length. The unordering length L_u is the number of consecutive packets arriving not in order. If the Pid of the received combination packet ($Pid_{current}$) is less than the Pid of the previous combination packet ($Pid_{previous}$), the packet comes not in order. And, if the $Pid_{current}$ is higher than the $Pid_{previous}$ at least two packets, the unordering length equal $Pid_{current} - Pid_{previous} - 1$. However, packet loss also causes this sign. Therefore, the sink must update which packet in this unordering length received in the future based on the method of recognizing the unordering packet mentioned before. L_u is updated periodically in every pre-configuration period (e.g., 5 seconds in this paper). In this paper, we also get the average value (l_u) using the Simple Moving Average (SMA) method for preventing the suddenly large change. After having l_u, the sink will send ceiling of it to the source via ACK packet by using the "unordering length" field. The flag U (update indication) is set to 1 also to let the source know the change to update the NC parameters.

When the source receives the updated l_u with the D flag, it will find the new NC parameters (n and m) which having the redundancy factor R approximate $\frac{n_{old}+k_{old}+l_u}{n_{old}}$. Noted that the maximum of k is 10.

3.3 Pause-ACK

Although the source update k based on the unordering length average l_u, the duplicated ACK still happens when the current unordering length is greater than k. It will affect the goodput performance least to the sending rate degradation. When the sending rate is low, the estimated l_u will be not accurate because this length depends on the sending rate proportionally. Thus, maintaining the stable sending rate is the primary key to this process. We propose the Pause-ACK mechanism to let the source delay some duplicated ACK to limit the number of mistakenly reducing the CWND.

Not the source, the sink will decide which ACK packet will be delayed to send to the upper layer (at the source). That is because the source need receive as much as ACK packets to estimate the channel condition. Thus, skipping some ACK packets at the sink is not a good solution. The sink will indicate the delayed ACK packet by using P flag in the NC-ACK header. The number of the delayed packets calculated from the average length of the duplicated ACK. We also get the average value (l_d) using the SMA method for preventing the suddenly large change. In our simulation, we see that the number of the delayed packets (pause length) equal to l_d+2 having the good goodput performance. At the source, when it receives the Pause-ACK packet, the source will store this packet to the paused buffer and set the sending schedule of 200 ms (following the setting of Delay-ACK of the regular TCP) for this ACK packet. When receiving a new ACK, the source will erase all the packets which having the ACK number less than that of the new ACK from the buffer. These processes are shown in the flowchart in the Fig. 3

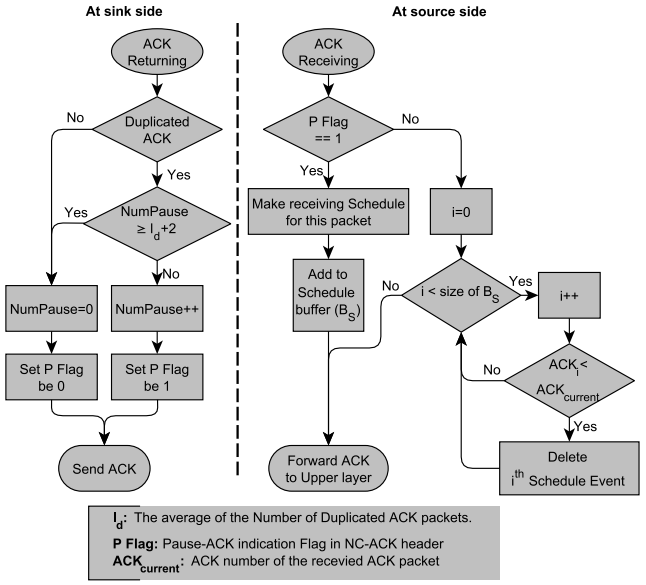


Fig. 3 Pause-ACK process

4. Simulation result

The simulation is accomplished by Network Simulator 3 (ns-3) [10] which is a discrete-event network simulator for Internet systems. We compare the transmission performance through goodput among the standard TCP NewReno, the previous study TCP/NCwLRLBE (refer to as TCP/NC from here), and our proposed scheme.

The topology of the simulation consists of a backbone with two arranged routers. One source and one sink are on either side of the backbone shown in Fig. 4. All links have a bandwidth of 1 $Mbps$. A propagation delay of the links connected to source and sink is 5 ms. A propagation delay on the link between two routers is configurable. The buffer size of the links is set to 100 packets. The TCP protocol type is NewReno. The payload size is 536 bytes. The minimum TCP timeout is 1 second. And, the number of Delayed-ACK is 2. The loss channel is the random loss channel for both two directions. The simulations are run at least 20 times to

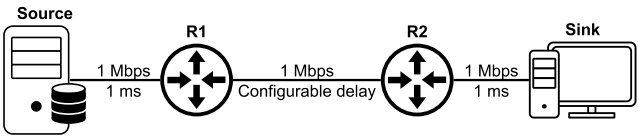
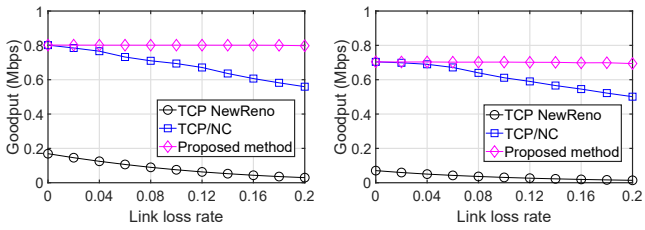


Fig. 4 Simulation topology



(a) Link loss rate of data sending direction is 0.04

(b) Link loss rate of data sending direction is 0.1

Fig. 5 Goodput comparison in the bi-direction loss

obtain the average value.

4.1 Goodput evaluation in bi-directional loss case

In this simulation, the propagation delay of the link between R1 and R2 is 10 ms. The loss happens in both directions at the intermediate link. The link loss rate of the sending data direction is constant of 0.04 (Case 1) and 0.2 (Case 2) while the link loss rate of the ACK receiving direction changes from 0 to 0.2. Based on the results in Fig. 5, we can see the advantage of the proposed scheme compared to TCP NewReno and TCP/NC. The goodput performance of the proposed scheme is kept stable even though the number of loss degree increases. While the goodput performance of both TCP NewReno and TCP/NC decreased when link loss rate increases.

4.2 Goodput evaluation in unordering case

In this simulation, the propagation delay is changed dynamically based on the proposed topology in [8]. The path delay changes in every "Inter-switch time" (δ). The propagation delay changes randomly around 200τ+50 with the standard deviation of $\frac{200\tau}{3}$ where $\tau \in [0, 2]$. We investigate the variation of the goodput performance on four values of δ that are 50 ms, 250 ms, 500 ms, and 1000 ms. Moreover, we show the results on three cases of link loss rate of zero, 0.05, and 0.1. The simulation results shows in Fig. 6, Fig. 6, Fig. 6, respectively. We compare the goodput performance among the protocols shown in Table 2 excluding the TCP/NCwLRLBE.

In Fig. 6, TCP/NC2 with Pause-ACK has the highest goodput performance where τ if from 0.2 to about 1.4 (δ=50ms), 1.6 (others). The goodput performance of TCP NewReno and TCP/NC2 is not stable when suddenly decreasing at τ of 0.4, 0.8, 1, and 1 corresponding to δ of 50 ms, 250 ms, 500 ms, and 1000 ms. The goodput performance of the proposed scheme is stable but smaller than that of TCP/NC2 with Pause-ACK. It is because the proposed scheme increases k to overcome the unordering problem. But in this case, the only Pause-ACK mechanism is enough when τ less than about 1.4. This issue can be solved when using more information to decide the value of k. The additional information which is unordering rate may be solved this problem. It will be our future work.

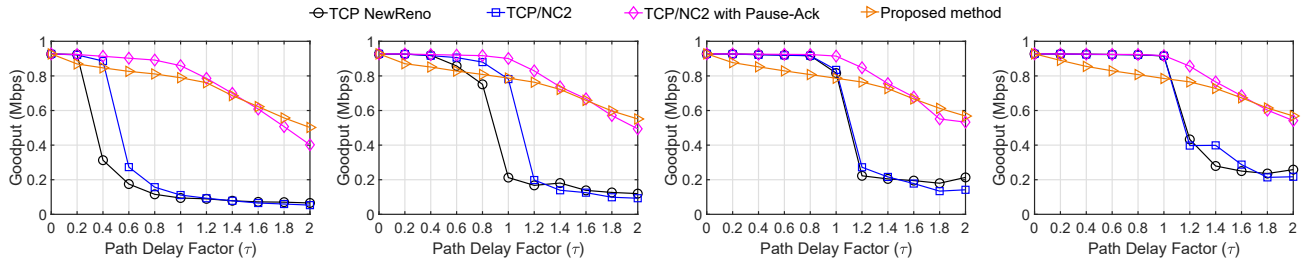
Table 2 Protocols description

Protocol	Description
TCP	TCP NewReno
TCP/NCwLRLBE or TCP/NC	TCP/NC with Loss Rate and Loss Burstiness Estimation
TCP/NC2	TCP/NCwLRLBE combine with ACK accumulation mechanism
TCP/NC2 with Pause-ACK	TCP/NCwLRLBE combine with ACK accumulation and Pause-ACK mechanisms
proposed scheme	TCP/NCwLRLBE combine with ACK accumulation, Unordering Estimation/ Adaptation, and Pause-ACK mechanisms

In Fig. 7 and Fig. 8, we can see that using only the Pause-ACK mechanism is not enough in the lossy case. The pause-ack mechanism can slowdown the retransmission process of the TCP when it delays the duplicated ACK unnecessarily. The goodput perfomance of TCP/NC2 with the Pause-ACK mechanism and TCP/NC2 is nearly the same in whole of τ. Meanwhile, the propose scheme get a goodput goodput perfomance compare to the others. The propose scheme increase k to overcome the unordering problem. This act can limit the number of duplicated ACK packets, resulting to decrease the number of Pause-ACK.

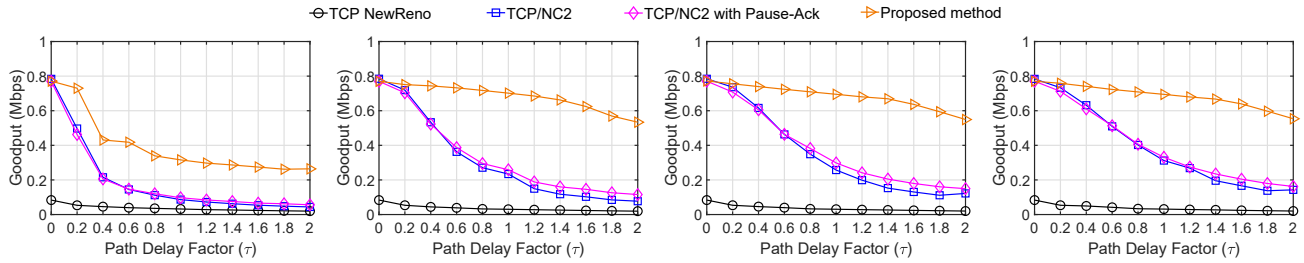
5. Conclusion

In this paper, we have proposed the scheme to help the TCP/NC work well on the bidirectional loss as well the receving not in order packet environments which are common in most practical channels. The simulation results on ns-3 have shown that the proposed scheme outperforms other protocols such as TCP NewReno and the recent variant of TCP/NC (TCP/NCwLRLBE). In the future, we will improve the scheme to adapt more heavy loss burstiness conditions where the number of continuous packets in both directions may exceed thirty-two packets. Instead of using a fixed length continuous packet information, we will use the random packet information. Another issue is to adapt unordered packet conditions which also affect seriously to the goodput performance of the system. Besides, we will consider more additional information on unordering estimation, such as considering the unordering rate.



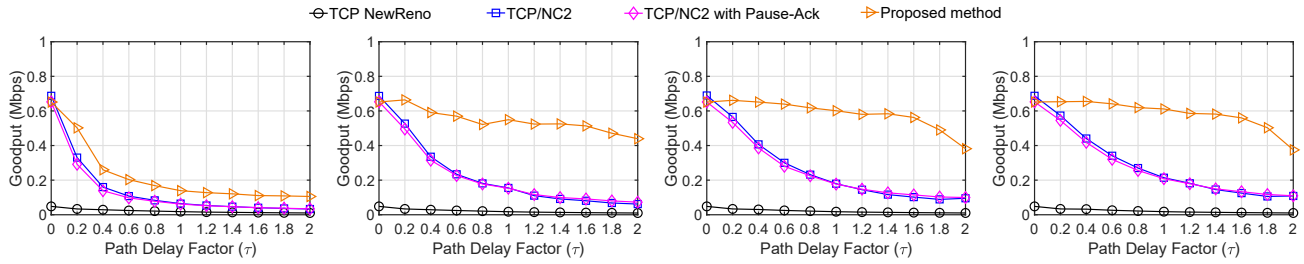
(a) $\delta = 50ms$ (b) $\delta = 250ms$ (c) $\delta = 500ms$ (d) $\delta = 1000ms$

Fig. 6 Goodput comparison in unordering case (link loss rate is zero)



(a) $\delta = 50ms$ (b) $\delta = 250ms$ (c) $\delta = 500ms$ (d) $\delta = 1000ms$

Fig. 7 Goodput comparison in unordering case (link loss rate is 0.05)



(a) $\delta = 50ms$ (b) $\delta = 250ms$ (c) $\delta = 500ms$ (d) $\delta = 1000ms$

Fig. 8 Goodput comparison in unordering case (link loss rate is 0.1)

Acknowledgment

The research results have been achieved by the "Resilient Edge Cloud Designed Network (19304)," the Commissioned Research of National Institute of Information and Communications Technology (NICT), and by JSPS Grant-in-Aid for Scientific Research (KAKENHI) Grant number JP18H06467 and JP16K00130, Japan.

References

[1] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, "TCP westwood: Bandwidth estimation for enhanced transport over wireless links," in Proc. of the 7th annual International conference on Mobile computing and Networking, Rome, Italy, pp. 287–297, Jul. 2001.

[2] J. K. Sundararajan, D. Shah, M. Medard, S. Jakubczak, M. Mitzenmacher, and J. Barros, "Network coding meets TCP: Theory and Implementation," Proceeding of the IEEE, vol. 99, no. 3, pp. 490–512, Mar. 2011.

[3] N. V. Ha, M. Tsuru, and K. Kumazoe, TCP Network Coding with Enhanced Retransmission for heavy and bursty loss, IEICE Transactions on Communications, vol. E100-B, no. 2, pp. 293–303, Feb. 2017.

[4] S. Song, H. Li, K. Pan, J. Liu, and S Y R Li, "Self-adaptive TCP Protocol Combined with Network Coding Scheme," in Proceeding of the 6th Conference on Systems and Networks Communications, Barcelona, Spain, pp. 20–25, Oct. 2011.

[5] C. Y. Cheng, and H. Y. Yi, "Adaptive Network Coding Scheme for TCP over Wireless Sensor Networks," Journal of Computers, Commu-nications and Control, vol. 8, no. 6, pp. 800–811, Dec. 2013.

[6] T. V. Vu, N. Boukhatem, and T. M. T. Nguyen, "Dynamic Coding for TCP Transmission Reliability in Multihop Wireless Networks," in Proceeding of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, Sydney, Australia, 6 pages, Oct. 2014.

[7] N. V. Ha, K. Kumazoe and M. Tsuru, "TCP Network Coding with Adapting Parameters for bursty and time-varying loss," IEICE Transactions on Communications, vol. E101-B, no. 2, pp. 476–488, Feb. 2018.

[8] K. Leung ; V. O. Li ; D. Yang, "An Overview of Packet Reordering in Transmission Control Protocol (TCP): Problems, Solutions, and Challenges," IEEE Transactions on Parallel and Distributed Systems, vol. 18, no. 4, pp. 522–535, Feb. 2007.

[9] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in Proceeding of IEEE International Symposium on Information Theory (ISIT), Yokohama, Japan, pp. 442-447, Jun. 2003

[10] Network Simulator 3 (ns-3), "https://www.nsnam.org/," accessed in Sep. 20th, 2018.