TCP with Network Coding Performance Under Packet Reordering

TCP with network coding performance under packet reordering

Nguyen Viet Ha, Masato Tsuru

Abstract The adverse impact of packet reordering besides packet loss is significant on the goodput performance of TCP (Transmission Control Protocol), a dominant protocol for reliable and connection-oriented transmission. With the primary purpose of improving the TCP goodput in lossy networks, the Network Coding technique was introduced. TCP/NC (TCP with Network Coding) is a promising approach which can recover lost packets without retransmission. However, the packet reordering has not been considered, and no study on that issue is found for TCP/NC. Therefore, in this paper, we investigate the goodput performance degradation due to the out-of-order reception of data or acknowledgment packets and propose a new scheme for TCP/NC to estimate and adapt to the packet reordering. The results of our simulation on ns-3 (Network Simulation 3) suggest that the proposed scheme can maintain the TCP goodput well in a wide range of packet reordering environments compared to TCP NewReno as well as TCP/NC.

1 Introduction

Transmission Control Protocol (TCP) has a long history and is still used widely in many applications as a primary connection-oriented transport protocol for reliable and in-order transmission of a byte sequence. One of the important features of TCP is the congestion control based on the congestion window (CWND). TCP will decrease the sending rate by reducing CWND when detecting a packet loss, which is necessary for fair bandwidth sharing and works almost correctly on conventional

Nguyen Viet Ha
Kyushu Institute of Technology
680-4 Kawazu, Iizuka-shi, Fukuoka, 820-8502 Japan. e-mail: nguyen.viet-ha503@mail.kyutech.jp

Masato Tsuru
Kyushu Institute of Technology
680-4 Kawazu, Iizuka-shi, Fukuoka, 820-8502 Japan. e-mail: tsuru@cse.kyutech.ac.jp

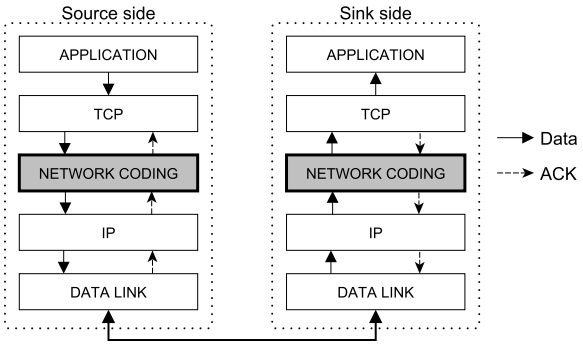

Fig. 1 NC layer in TCP/IP model

wired single-path networks. However, when being applied to complex and challenging environments, TCP exposes a weakness in maintaining the goodput performance due to lossy links as well as packet reordering.

On lossy links, packets are lost not by congestion but by physical link errors. In such conditions, since TCP cannot distinguish the type of losses and consider any packet loss as a congestion signal, TCP decreases the CWND mistakenly, resulting in a seriously lower goodput. Some TCP variants have been proposed to overcome this issue, e.g., TCP Westwood+ [1], but they are not useful in heavy loss environments. Another approach is combining Network Coding with TCP (called TCP with Network Coding - TCP/NC) [2].

In TCP/NC, a new NC sub-layer is added between TCP and Internet layer shown as Fig. 1 to control the packet loss issue. This sub-layer become an intermediate handler after and before the TCP layer sends and receives the packet, respectively. At sending side, NC sub-layer combines n original TCP segments to m combination packets with $m>n$. At receiving side, the sink is expected to recover all n original segments if the number of lost packets is no more than $m-n$.

The principal purpose of TCP/NC is improving the goodput performance in lossy networks. Therefore, many variants of TCP/NC have been developed for this demand. Some exemplary contributions are mentioned following. 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 lead 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 in adjusting the NC parameters without disrupting the current settings. Our study in [8] also solved the problem of Acknowledgment (ACK) packet loss by adding

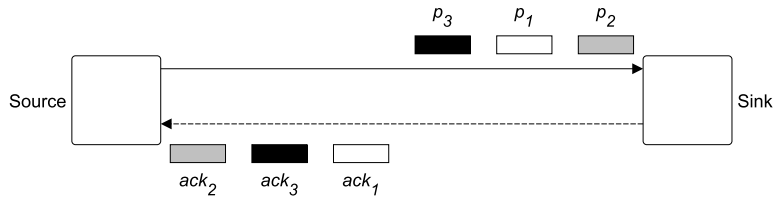

Fig. 2 Illustration of forward-path reordering and reverse-path reordering

new information in the NC-ACK header of the ACK packet to convey the reception information of not only the current packet but also the previous packets.

Besides the packet loss, packet reordering also causes goodput performance degradation. There are many reasons causes the packet reordering, such as multipath routing for load balancing, route fluttering, and Link-layer retransmission. Packet reordering (i.e., out-of-order packet arrival) happens not only on data packets (referred to as forward-path reordering) but also on ACK packets (referred to as reverse-path reordering). The simple illustration of packet reordering is shown in Fig. 2. p_1, p_2, and p_3 sent in the order, but they are received out of order at the sink. In the reverse side, ACK packets are transmitted in the order of ack_2, ack_1, and ack_3, but ack_1 arrives after ack_2 and ack_3 at the source. In the TCP operation, when the sink receives a packet out of order, the sink will put the same ACK number with the previous ACK packet into the current ACK packet. When the source receives many duplicated ACK packets, it enters to retransmission process and decreases CWND mistakenly. The goodput performance of not only the regular TCP but also the current TCP/NC variants will be affected. Another hand, reverse-path reordering will not impact the performance of the regular TCP, but it is dangerous in some TCP/NC variants. If the information conveyed in the ACK packet is used to estimate the channel conditions, e.g., calculating the loss burstiness, receiving out of order ACK packet will adversely affect the estimation process.

In this paper, we propose a new scheme to estimate the reordering conditions, such as reordering length, and the number of duplicated ACK to react the reordering affectation. This scheme can change the NC parameters based on not only loss/burstiness but also the reordering information. Besides adjusting the NC parameters, we introduce the Pause-ACK mechanism to helps the source delay some the duplicated ACK in the determined period to wait for the proper ACK packet. This mechanism can avoid the source receive not incorrect ACK packet and enter to retransmission mistakenly. Consequently, the goodput performance will maintain stable.

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 is proposed with the main responsibility of handling the packet loss to robustness in the lossy channel without any modifications of the TCP protocol. Therefore, new NC sub-layer is complemented and placed between TCP and network layer shown in Fig. 1. This sub-layer handles the incoming and outgoing packets from TCP and network layer, respectively. It works transparently with other layers; thus, TCP/NC can apply to any current devices. If NC sub-layer can recover all packet losses, the TCP layer is unaware of the loss events; thus, the goodput is not affected by the lossy channel. Besides, NC sub-layer will return ACK packet with ACK number determining based on the degree of freedom and the seen/unseen definition [2]. It means that the sink can return the different ACK number for every received packet without waiting for decoding all the packets. When the sink receives enough combination packet, all original packets will be decoded. Therefore, the CWND is kept increasing even though some combination packets are lost. Thus, the goodput 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 (encoding process). 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 (decoding process). 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 [10]). And the computation is implemented in a Galois field (e.g., GF(2^8)).

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

2.3 TCP functionality

As mentioned, TCP/NC must not interference the TCP operation; thus, it works transparently to other layers. Moreover, TCP functionalities have been studied and worked stably in a long history. TCP/NC should take all these advantages such as

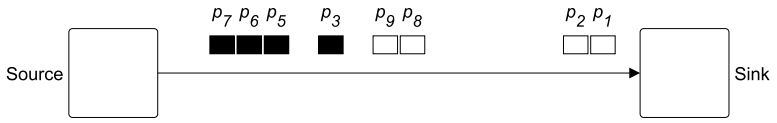

Fig. 3 Illustration of packet reordering length

retransmission and congestion control mechanisms. The source must retransmit the packets when the number of packet losses is larger than the recovery capacity of NC sub-layer. In that situation, both the TCP layer and NC sub-layer at the source receive many duplicated ACK equaling the oldest "unseen" packet. The NC sub-layer only needs to wait for the retransmission from the TCP layer. Increasing or decreasing the CWND is also controlled by TCP layers, not NC sub-layer.

3 Proposed scheme

3.1 Packet reordering estimation

Packet reordering issue causes goodput degradation as shown in the study [9]. When the sink receives packets out of order, all returning ACK packet will have the same ACK number with the ACK packet for the last in order packet. 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). TCP/NC relies on the same mechanism for congestion control; hence, like TCP, unwanted duplicated ACKs affect the goodput of TCP/NC. However, unlike TCP, TCP/NC returns the ACK number based on the seen/unseen packets. Thus, we suggest increasing the size of the NC window sliding, which equals the number of packets in one combination, to overcome this problem. If the largest distance of two out of order packets less than NC window sliding size, the packet reordering is not sensed by the sink-side TCP that can send an ACK packet with an incremental ACK number.

To estimate the largest distance of two out of order packets (referred to as reordering length), we use the existing field Packet-ID (*Pid*) in the NC header which is the sequential number assigning to the combination for loss estimation purposes. The sink will perform that responsibility. 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 out of order. And, if the $Pid_{current}$ is higher than the $Pid_{previous}$ at least two packets, the reordering length equal $Pid_{current} - Pid_{previous} - 1$. However, it may contain the packet loss; thus, the sink must record which packet in this length will be received. The final value of reordering length equals the number of the packets will be received. Another scheme will handle these packet loss such as TCP/NCwLRLBE [7] that does not scope in this paper. Fig. 3 shows a simple example to explain the reordering length. When the sink receives the p_8, the tem-

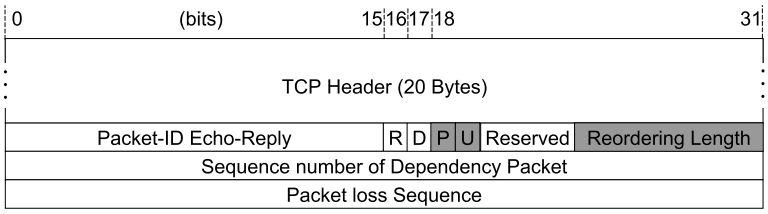

Fig. 4 NC-ACK header

Table 1 NC-ACK header fields description

Field name	**Description**
Packet-ID Echo-reply	The packet identity echo reply
R	The redundancy flag
D	The dependence flag
P	The Pause-ACK flag
U	The update indication flag
Reserved	Reserved for the future use
SN of the dependence pkt	The sequence number 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*.
Reordering Length	Average value of the reordering length

porary length is set to 5. After that, when the p_3, p_5, p_6, and p_7 arrives to the sink, the reordering length is set to 4 because the p_4 is lost. The reordering length in average (L_u) is calculated periodically in every pre-configuration period (e.g., 5 seconds in this paper). Besides, we also get the average value (l_u) using the Simple Moving Average (SMA) method with the window length of 5 for preventing the suddenly substantial change. After determining l_u, the sink will send the ceiling of l_u to the source via ACK packet by using the *"reordering length"* field in the NC-ACK header shown in Fig. 4 and Table. 1. The flag U (update indication) in the NC-ACK header is set to 1 also to let the source know the change to update the NC parameters. Noted that the NC header and NC-ACK header retained from our previous propose which is discussed in [3, 7, 8].

When the source receives the *"reordering length"* 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.

3.2 Pause-ACK

Since a large k results in a lower goodput due to too many redundant packets, the average value l_u is used to limit k. However, the instant packet reordering length some-

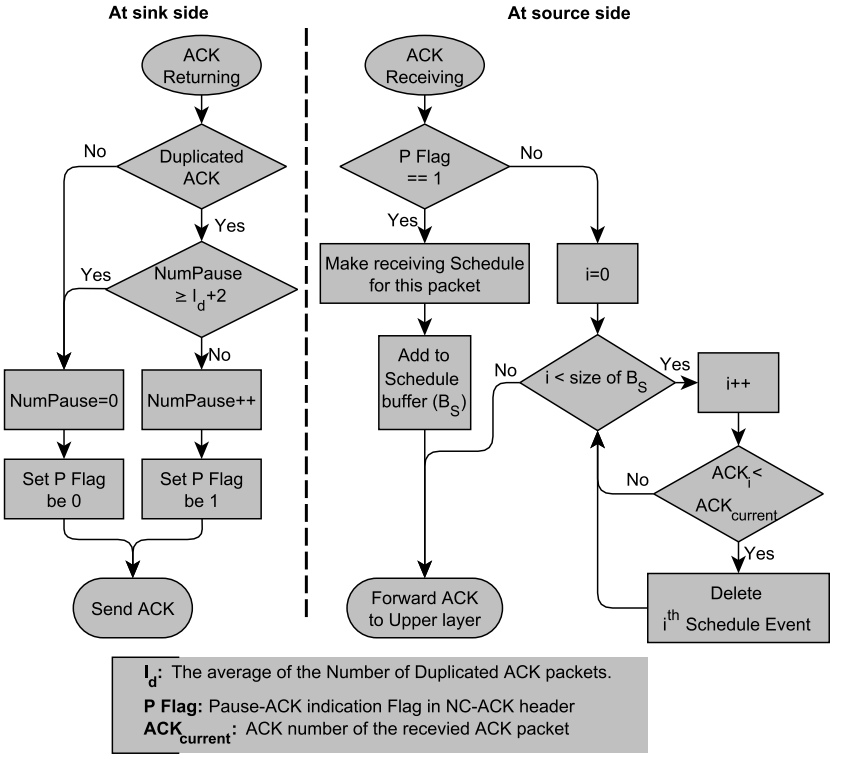

Fig. 5 Pause-ACK process

times becomes larger than the average value, which still causes many duplicated ACKs. It will reduce the goodput, leading to the sending rate degradation. Keeping the stable sending rate plays an important role in this scheme because the reordering length depends on the sending rate proportionally. When the sending rate is low, the estimated l_u will be not accurate. We propose the mechanism called Pause-ACK to let the source delay some duplicated ACK to wait for the proper ACK. This act can help the sink limit the number of mistakenly reducing the CWND.

The source has responsibility for estimating the loss condition; thus, it needs to receive as much as the number of ACK packets possible. Therefore, delaying the ACK packet to send to the upper layer will perform at the source. But, determining which the delayed ACK packet is complete at the sink as shown on the left side of Fig. 5 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 length of the duplicated ACK (referred to as the pause length). The pause length in average (L_d) is calculated periodically in every pre-configuration period. We also get the average value (l_d) using the SMA method with the window length of 5 for preventing the suddenly substantial change. Based on our simulation, we see that the number of the delayed packets (pause length) equal to l_d+2 having the good goodput performance.

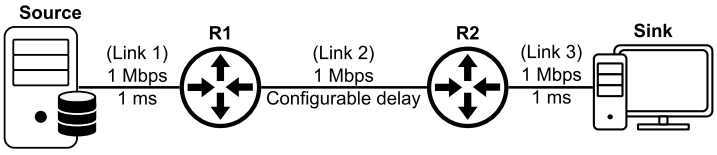

Fig. 6 Simulation topology

At the source, when it receives the ACK packet with a *P* flag, the source will store this packet to the Pause-ACK buffer and set the sending schedule to after 200 *ms* for this ACK packet. (at this point we following the setting of Delay-ACK of the regular TCP). Pause-ACK mechanism only works on duplicated ACK packets. Thus, if the source receives the new ACK packet having ACK number higher than that of ACK packets in the buffer, the source will erase these packet from the buffer. These process at the source is shown on the right flowchart in Fig. 5.

4 Simulation result

4.1 Simulation setup

The simulation is accomplished by Network Simulator 3 (ns-3) [11] which is a discrete-event network simulator for Internet systems. 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. 6. The simulation parameters is shown in Table. 2. The configuration propagation delay is changed dynamically based on the proposed topology in [9]. The path delay changes in every "Inter-switch time" (δ). The propagation delay changes randomly around $200\tau+50$ 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*.

TCP NewReno, TCP/NCwER, TCP/NCwLRLBE, TCP/NC with the Pause-ACK mechanism (TCP/NC with Pause-ACK), and the proposed scheme are compared in the time-averaged goodput. The detailed description is shown in Table 3. Note that TCP/NCwLRLBE protocol in this paper is an improved version developed in [8] for the ACK loss problem.

4.2 Goodput evaluation in reordering case

4.2.1 No link loss case

In this simulation (Fig. 7), we can see that only increasing *k* is not enough to overcome the packet reordering. TCP and TCP/NCwLRLBE have a worse goodput per-

Table 2 Simulation parameters

Parameter	**Value**
Bandwidth of all links	1 *Mbps*
Propagation delay of Link 1 and 3	5 *ms*
Propagation delay of Link 2	Configurable
Buffer size in a router	100 packets
TCP protocol	TCP NewReno
Maximum CWND	65535 bytes
Payload size	536 bytes
Minimum of TCP Timeout	1 second
TCP Delayed-ACK	2 packets
Loss model	Random loss model of ns-3 simulator
Simulation iteration	20 times

Table 3 Protocols description

Protocol	**Description**
TCP	TCP NewReno
TCP/NCwER	TCP/NC with Enhanced Retransmission [3]
TCP/NCwLRLBE	TCP/NC with Loss Rate and Loss Burstiness Estimation [8]
TCP/NCwLRLBE with Pause-ACK	TCP/NCwLRLBE combined with Pause-ACK mechanisms
Proposed scheme	TCP/NCwLRLBE combined with Reordering Estimation/ Adaptation and Pause-ACK mechanisms

formance compared to other protocols. Because of no link loss, TCP/NCwLRLBE keeps k is zero or a small value. In the TCP/NCwER cases, k is constant of 5 and 10; the goodput is stable in the low degree of packet reordering. But when τ increases, the goodput is decreased. For example of k=10, the goodput start decreasing at τ of 0.8, 1, 1.4, and 1.4 corresponding to δ of 50, 250, 500, and 1000 *ms*.

The goodput performance of the proposed scheme is stable but smaller than that of TCP/NCwLRLBE with Pause-ACK. It is because the proposed scheme increases k to overcome the out-of-order packet arrivals. But in this case, the only Pause-ACK mechanism is enough when τ less than about 1.4. In future work, this issue can be solved by using more information to decide the value of k. The additional information which is packet reordering rate may be solved this problem.

4.2.2 Link loss case

In this simulation, the link loss rate set to 0.05 and 0.1. The results show in Fig. 8, Fig. 9, respectively. The loss happens at the interface on R2 connected to R1 in the data sending direction.

Based on the results, we can see the advantage of the proposed scheme compared to the TCP/NCwLRLBE using only the Pause-ACK mechanisms. In Fig. 8 and Fig. 9, we can see that using only the Pause-ACK mechanism is not enough in

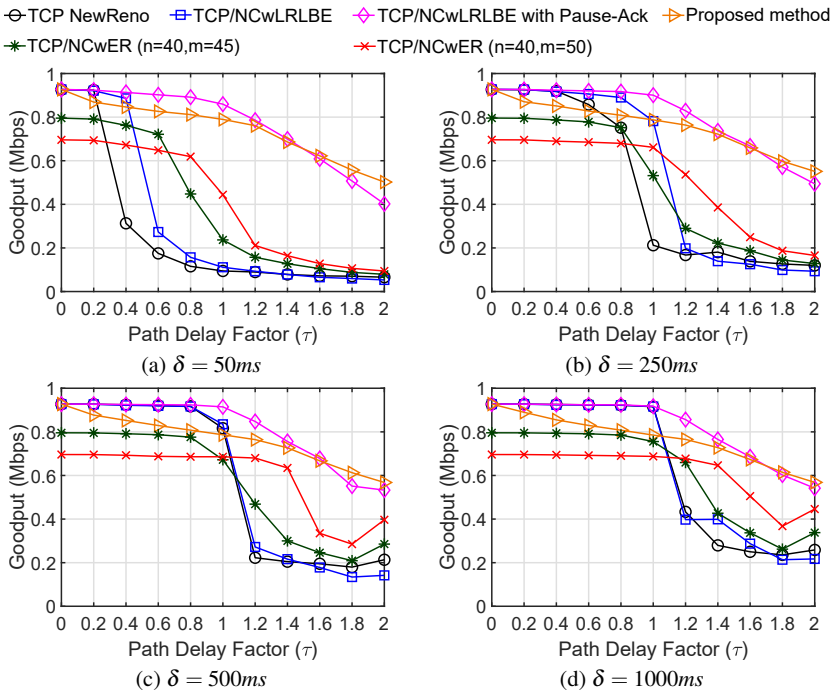

Fig. 7 Goodput comparison in different reordering cases (link loss rate is zero)

the lossy case. The Pause-ack mechanism can slow down the retransmission process of the TCP when it delays the proper duplicated ACK mistakenly. The goodput performance of TCP/NCwLRLBE with the Pause-ACK mechanism is better than TCP/NCwLRLBE but be not significant. Meanwhile, the proposed scheme gets a goodput performance to compare to the others. The proposed scheme increases k to overcome the packet reordering problem. This act can limit the number of duplicated ACK packets, resulting in decreasing the amount of Pause-ACK.

5 Conclusion

In this paper, we have proposed the scheme to help the TCP/NC work well on receiving out-of-order packets, which may sometimes happen in most practical complex network environments. 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. In the future, we will improve the scheme to well adapt to any packet reordering condition which depends on diverse network environments and affects seriously to the performance of the system. One possible approach is to consider

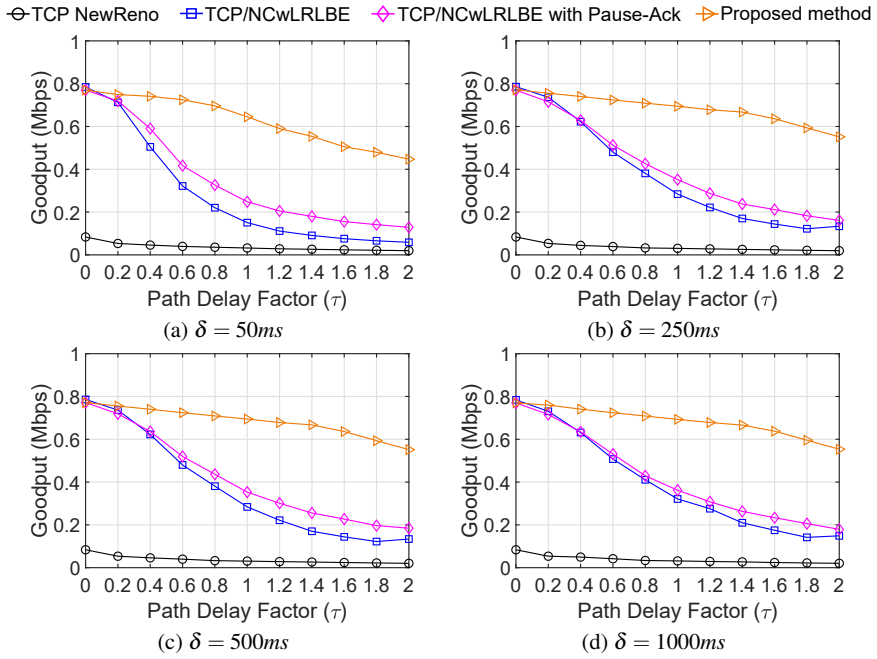

Fig. 8 Goodput comparison in different reordering cases (link loss rate is 0.05)

more additional information on packet reordering estimation such as the packet reordering rate.

Acknowledgements 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. Mascolo, S., Casetti, C., Gerla, M., Sanadidi, M.Y., WangR.: TCP westwood: Bandwidth estimation for enhanced transport over wireless links. Proceeding of the 7th annual International conference on Mobile computing and Networking, 287–297 (2001).
2. Sundararajan, J.K., Shah, D., Medard, M., Mitzenmacher, M., Barros, J.: Network coding meets TCP. Proceeding of the IEEE International conference on Computer Comunication, 280–288 (2009).
3. Ha, N.V., Kumazoe, K., & Tsuru, M.: TCP Network Coding with Enhanced Retransmission for heavy and bursty loss. IEICE Transactions on Communications, E100-B(2), 293–303 (2017).
4. Song, S., Li, H., Pan, K., Liu, J., Li, S.Y.R.: Self-adaptive TCP Protocol Combined with Network Coding Scheme, Proceeding of the 6th Conference on Systems and Networks Com-

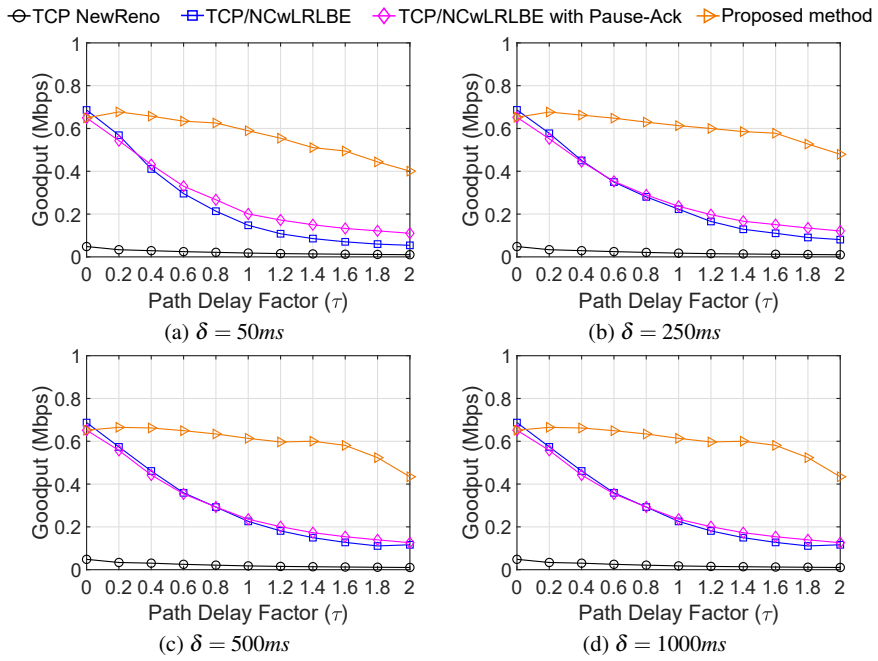

Fig. 9 Goodput comparison in different reordering cases (link loss rate is 0.1)

munications, 20–25, (2011).
5. Cheng, C.Y., Yi, H.Y.: Adaptive Network Coding Scheme for TCP over Wireless Sensor Networks, Journal of Computers, Communications and Control, 8(6), 800–811 (2013).
6. Vu, T.V., Boukhatem, N., Nguyen, T.M.T.: Dynamic Coding for TCP Transmission Reliability in Multi-hop Wireless Networks, Proceeding of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, 6 pages (2014).
7. Ha, N.V., Kumazoe, K., Tsuru, M.: TCP Network Coding with Adapting Parameters for bursty and time-varying loss. IEICE Transaction of Communications, E101-B(2), 476–488 (2018).
8. Ha, N.V., Tsuru, M.: TCP/NC performance in bi-directional loss environments. Proceeding of the International Conference on Electronics, Information, and Communication, 4 pages. (to appear in ICEIC 2019, January 2019)
9. Leung, K., Li, V.O., Yang, D.: An Overview of Packet Reordering in Transmission Control Protocol (TCP): Problems, Solutions, and Challenges. IEEE Transactions on Parallel and Distributed Systems, 18(4), 522–535 (2007).
10. Ho, T., Koetter, R., Medard, M., Karger, D., Effros, M.: The benefits of coding over routing in a randomized setting, Proceeding of IEEE International Symposium on Information Theory, 442–447 (2003).
11. Network simulator (ns-3). *https://www.nsnam.org/*. Accessed in September 20, 2018.