TCP/NC performance in bi-directional loss environments

著者	Nguyen Viet Ha, Tsuru Masato
journal or publication title	2019 International Conference on Electronics, Information, and Communication (ICEIC)
year	2019-05-06
URL	http://hdl.handle.net/10228/00007453

doi: info:doi/10.23919/ELINFOCOM.2019.8706367

TCP/NC performance in bi-directional loss environments

Nguyen Viet Ha
Kyushu Institute of Technology
Fukuoka, Japan
nguyen.viet-ha503@mail.kyutech.jp

Masato Tsuru
Kyushu Institute of Technology
Fukuoka, Japan
tsuru@cse.kyutech.ac.jp

***Abstract*—Transmission Control Protocol with Network Coding (TCP/NC) is studied to improve the goodput performance of the standard TCP 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. Basically, *n*+*k* combination packets combined from *n* original packets are sent by the source. When the sink receives enough *n* combination packets, the sink can calculate *n* original packet even though *k'* packets are lost where *k'* is less than or equal to *k*. Most versions of TCP/NC consider the loss in only the direction of sending data; however, the loss in the reverse direction, i.e., the direction of sending an acknowledgment, affects seriously to goodput performance, especially in TCP/NC with an automatic estimation of Network Coding parameters (*n* and *k*) case. In this paper, we propose a new scheme for the bi-directional loss issue. 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 compared to the TCP NewReno and our previously proposed, TCP/NC with Loss Rate and Loss Burstiness Estimation (TCP/NCwLRLBE).**

Keywords—TCP/NC; TCP; Network Coding; Bi-directional loss; lossy networks; ns-3.

I. Introduction

Transmission Control Protocol (TCP) is currently used for reliable transmission in many applications due to its advantages on connection-oriented and congestion control features. However, the transmission performance of TCP is seriously degraded by packet loss through the lossy networks (e.g., wireless networks). It is also because of the TCP's congestion control. In this feature, a packet loss is considered as a congestion signal; hence, TCP will decrease the sending rate by decreasing the congestion window (CWND) when detecting a packet loss. The packet loss is caused not only by network congestion but also a channel. The CWND should be kept stable to overcome the temporary bad condition of the channel. But TCP decreases the CWND mistakenly affecting to transmission performance seriously. To overcome this issue, some TCP variants have been proposed, e.g., TCP Westwood+ [1]. Another promising approach is combining Network Coding with TCP (called TCP with Network Coding - TCP/NC) [2] which has more benefits than using only the TCP.

TCP/NC introduces a new NC sub-layer sandwiching between TCP and Internet layer shown as Fig. 1. Basically, NC sub-layer receives *n* TCP segments, combines them to *m* combination packets (referred to as encoding) with *m*>*n*. It is

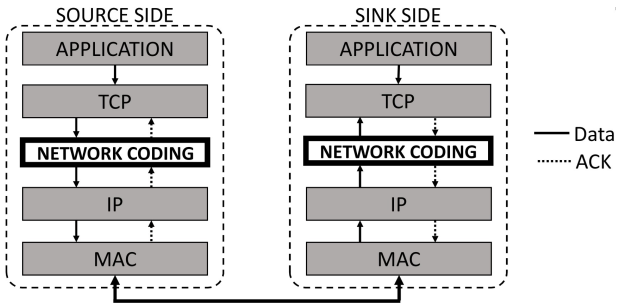

Fig. 1. Network coding sub-layer

supposed that *k'* combination packets are lost through a lossy channel. If *k'* is less than or equal *k* which equal subtraction of *m* and *n*, the sink is expected to recover the remaining combination packets to all original segments (referred to as decoding). While NC can apply on wide range area, we focus only one 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.

The network conditions change over time in the practical, especially wireless networks; thus, using the constant NC parameters (*n* and *m*) is not a good solution. Some variants of TCP/NC have been proposed to automatically adjust the NC parameters when the channel condition changes such as Self-Adaptive NC-TCP (SANC-TCP) [3], Adaptive NC (ANC) [4], Dynamic Coding (DynCod) [5], and TCP/NC with Loss Rate and Loss Burstiness Estimation (TCP/NCwLRLBE) [6]. However, most methods need to receive all acknowledgment (ACK) packets to estimate the correct channel conditions (link loss rate and loss burstiness) to determine the correct NC parameters. For example, in our previous study with TCP/NCwLRLBE, we use the Packet Identification (*Pid*) in the sending combination packet and *Pid-Echo-Reply* in the ACK packet to help the source estimate the channel conditions and adjust NC parameters. However, the practical channels are always a bi-direction loss environment; thus, these above methods cannot work well in the real. In this paper, we propose a new scheme which can help the sink notify to the source the status not only of the currently received packet but also of the previously received or lost packets. Consequently, the source can estimate correctly all the necessary values even though some ACK packets are lost.

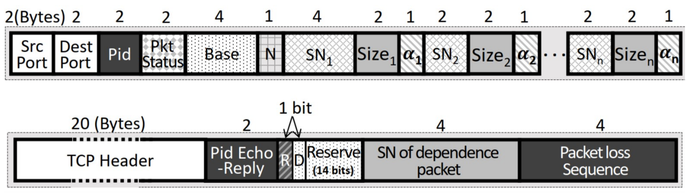

Fig. 2. NC header (above) and NC-ACK header (below)

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.

II. TCP/NC OVERVIEW

A. *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.

B. *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 sliding window) is k+1. Besides, α is selected randomly; thus, the coding algorithm is also called Random Linear Network Coding (RLNC [7]). All the computation is implemented in a Galois field (e.g., in a GF(2^8)). All operators are expressed to "exclusive or" (XOR) and lookup table; hence, the complexity of computation is small to apply to the real system.

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

C. *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 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.

III. PROPOSED SCHEME

The proposed scheme is introduced based on the previous study in [6] (TCP/NCwLRLBE). TCP/NCwLRLBE can automatically estimate the channel conditions (e.g., link loss rate, loss burstiness) and then it adjusts the NC parameters. Basically, the proposed scheme works as TCP/NCwLRLBE, but it does not decide the estimated value immediately. It will update the estimated values over time based on the previous information stored in each ACK packets. Therefore, the proposed scheme can estimate the correct channel conditions even though some ACKs are lost.

A. *NC header and NC-ACK header*

The NC header and NC-ACK header are retained from TCP/NCwLRLBE shown in Fig. 2 and described in Table. I. The content of the header fields is not scoped in this paper. The detail explanation is discussed in [5][8]. In this paper, we only focus on *Pid*, *Pid Echo-Reply*, and a new field added in NC-ACK header, "*Packet loss Sequence*." *Pid* are the continuous

TABLE I. HEADER FIELDS DESCRIPTION

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 the original packet in the combination packet
SN₁	The SN of the first original packet
SNₙ	Equal to the SN of the nth packet subtract to *SN₁*
Sizeₙ	The payload size of nth packet
αₙ	The nth NC coefficient
Pid Echo-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
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*.

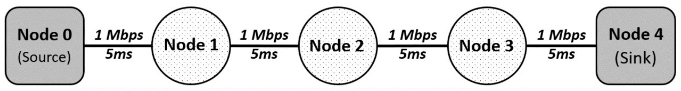

Fig. 3. Simulation topology

numbers assigned sequentially for every sending combination packets. *Pid Echo-Reply* is used in ACK packet to turn the acknowledgment from the sink to the source. It is set to the value of the *Pid* of the recently received combination packet. Based on the *Pid Echo-Reply*, the source can know which packet is received and which packet is lost.

The *Packet loss Sequence* field includes thirty-two binary number presenting to the status (received or lost) of recently thirty-two continuous 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 receive all the necessary information to estimate the channel condition even though some ACK packets are lost.

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 of this paper); thus, it does not affect the goodput performance.

B. Estimate the network condition

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 5 seconds (configurable parameter). Therefore, losing some ACK packets affects the estimation process of TCP/NCwLRLBE. TCP/NCwLR-LBE 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 (5 seconds) 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.

IV. Simulation Result

The simulation is accomplished by Network Simulator 3 (ns-3) [9] 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.

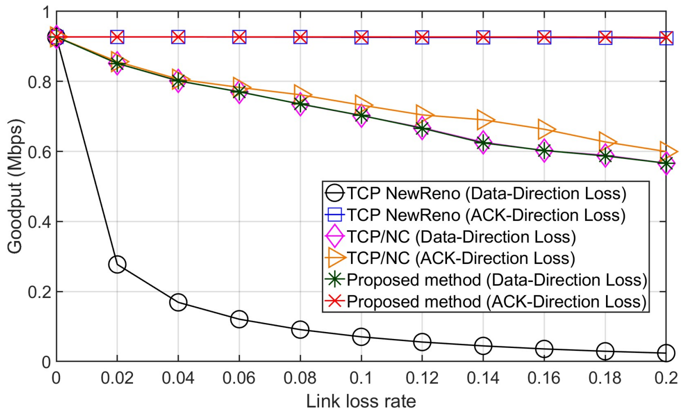

Fig. 4. Goodput comparison in one direction loss only (data sending direction or ACK receiving direction)

A. Simulation setup

The topology of the simulation consists of a backbone with three tandemly arranged routers. One source and one sink are on either side of the backbone shown in Fig. 3. 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. 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 link loss rate is adjusted from zero to 0.2. The simulations are run at least 20 times to obtain the average value.

B. Goodput evaluation

1) Loss at one direction

In this simulation, we consider two cases. First, the loss happens at the interface of Node 2 which connects to Node 1. And second, the loss happens at the interface of Node 1 which connects to Node 2.

In the first case, the goodput performance of TCP NewReno is decreased when the link loss rate increases shown in Fig. 4. The reason is the congestion control algorithm of TCP mistakenly decreases the CWND whenever it sees the packet loss. Meanwhile, the goodput performance of TCP/NC and the proposed scheme are completely the same and better than that of TCP NewReno.

In the second case, the goodput of TCP NewReno is maintained in high goodput at 0.93 *Mbps*. The ACK packet loss in TCP NewReno case is like the Delayed-ACK process; thus, it does not affect the goodput performance. However, in the TCP/NC case, the ACK packet loss makes the source receive lack information to estimate the channel condition. The source mistakenly recognizes the ACK packet loss as the data packet loss; hence, it increases sending the unnecessary redundancy packets causing the link bandwidth to be wasted.

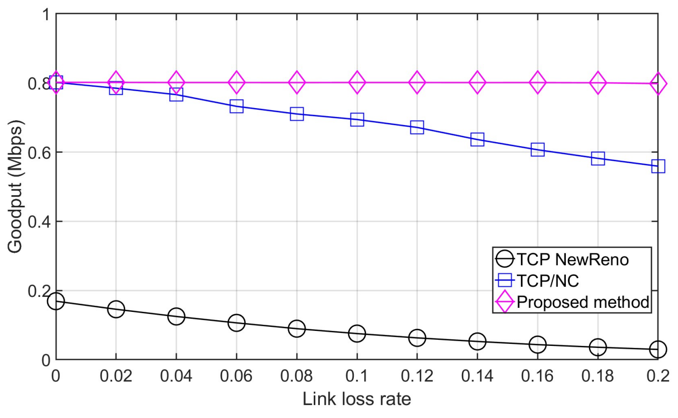

Fig. 5. Goodput comparison in the case that the link loss rate of data sending direction is 0.04

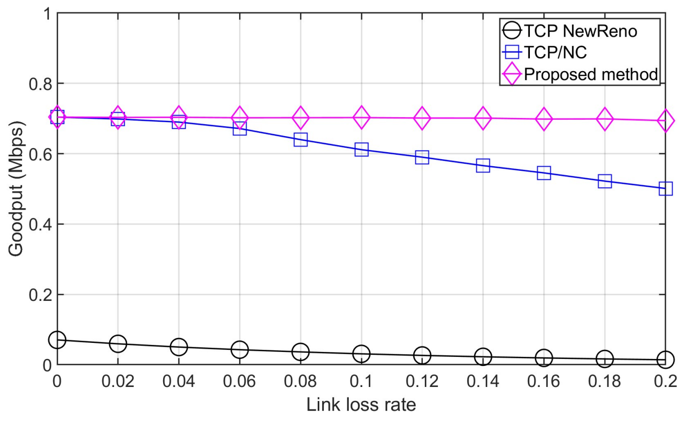

Fig. 6. Goodput comparison in the case that the link loss rate of data sending direction is 0.1

It means that the goodput performance is decreased. Meanwhile, the proposed scheme can update the correct estimation values using the set information of the previous packet in each ACK packet. The correct the channel condition is determined correctly. In this situation, no packet loss happens; thus, the proposed scheme does not send any redundancy packets. The goodput performance of the proposed scheme is the same that of TCP NewReno.

2) Loss at both directions

In this simulation, we consider the loss happens in both directions at the connection between Node 1 and Node 2. We keep the constant link loss rate of the sending data direction is 0.04 (Case 1) and 0.2 (Case 2) while adjusting the link loss rate of the ACK receiving direction from zero to 0.2. The results are shown in Fig. 5 and Fig. 6. In both cases, 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 is decreased.

V. CONCLUSION

In this paper, we have proposed the scheme to let the sink inform not only status (received or lost) of the latest packet but also the status of thirty-one previous packets. It helps the source know exactly the number of packet losses and the loss burstiness even though some ACK packets are lost. 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 bi-directional loss environments which are common in most practical channels.

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.

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, 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 Proceedings 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] 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 (ICSNC), Barcelona, Spain, pp. 20–25, Oct. 2011.

[4] C. Y. Cheng, and H. Y. Yi, “Adaptive Network Coding Scheme for TCP over Wireless Sensor Networks,” Journal of Computers, Communications and Control, vol. 8, no. 6, pp. 800–811, Dec. 2013.

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

[6] 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.

[7] 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

[8] 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.

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