Masking Lossy Networks by TCP Tunnel with Network Coding

doi: 10.1109/ISCC.2017.8024702

Masking Lossy Networks by TCP Tunnel with Network Coding

Nguyen Viet Ha, Kazumi Kumazoe, Kazuya Tsukamoto, Masato Tsuru
Kyushu Institute of Technology, Japan
nvha@infonet.cse.kyutech.ac.jp, kuma@ndrc.kyutech.ac.jp, tsukamoto@cse.kyutech.ac.jp, tsuru@cse.kyutech.ac.jp

***Abstract*—Transmission Control Protocol (TCP) with Network Coding (TCP/NC) was designed to recover the lost packets without TCP retransmission to improve the goodput performance in lossy networks. However, TCP/NC is too costly to be implemented in some types of end devices, e.g., with less memory and power. In addition, TCP/NC across loss-free but thin networks may waste scarce link bandwidth due to the redundant combination packets sacrificed for the lossy network. In this paper, we propose the TCP/NC tunnel to convey end-to-end TCP sessions on a single TCP/NC flow traversing a lossy network between two special gateways without per-flow management. We implemented and validated our proposal in Network Simulator 3, in which each gateway runs a reinforced version of TCP/NC that we previously developed. The results show that the proposed TCP/NC tunnel can mitigate the goodput degradation of end-to-end TCP sessions traversing a lossy network without any change in TCP on each end host.**

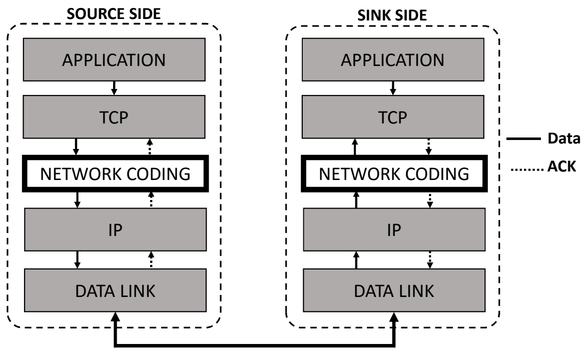


Fig. 1: NC layer in the TCP/IP model

I. Introduction

The conventional Transmission Control Protocol (TCP) recognizes all packet losses, even if they are caused by the lossy network, to be a sign of network congestion and cuts down the sending rate, hence its performance is considerably degraded. TCP with Network Coding (TCP/NC) was presented [1] to cope with this problem. The term Network Coding (NC), while being used in a broader sense, is mostly referred as a technique in which multiple original packets are combined into multiple coded packets to traverse a network and those packets are decoded to the original packets after traversing the network, in order to improve the throughput, delay, and/or resilience. In TCP/NC, the source sends the data as random linear NC combination packets (referred to as combination packets) to the sink across a lossy network. When some of combination packets are lost, the sink is expected to recover all data using the remaining combination packets without retransmission. It avoids unnecessary reduction of the sending rate according to TCP congestion windows (CWND) control. A new NC layer is added into the protocol stack between the TCP and IP layers to provide the recovery capability shown in Fig. 1. This layer operates transparently with upper and lower layer; thus, it can take the functionality of the original TCP protocol such as a congestion control and a retransmission mechanism with the benefit of the NC in recovering the packet loss. In other words, TCP/NC can recover the lost packets by combining forward erasure correction (FEC) and automatic repeat-request (ARQ) schemes.

Although NC in general and TCP/NC in particular are shown to have certain benefits in lossy networks, TCP/NC is not easy to be deployed because of several reasons. TCP/NC is required to be implemented at both ends of connection because it is incompatible with the existing TCP protocol. In addition, TCP/NC is costly to be implemented in some types of end devices e.g., with less memory and power. Furthermore, when a lossy network is located in the middle as an intermediate network (e.g., satellite network, long-distance fixed wireless backbone, and the lossy internet especially in the developing regions), performing the TCP/NC at end-hosts may waste scarce link bandwidth of local networks which are expected to be a low-loss environment.

To mitigate the performance degradation problem of conventional TCPs traversing lossy networks and/or wireless networks without change of end-host TCP, a variety of Performance Enhanced Proxy (PEP) approach have been studied in either split type (e.g., TRL-PEP [2]) or snoop type (e.g., D-Proxy [3]). While their effectiveness has been shown in some conditions, they require complicated per-TCP flow management on the proxy nodes.

On the other hand, TCP tunnel in general is a kind of proxy to provide a reliable virtual link that encapsulates and transparently conveys IP packets over one or more TCP sessions between two (ingress/egress) entities, e.g., interfaces, routers, or gateways. TCP tunnel is used for diverse purposes, including security by encryption, performance improvement by flow aggregation, flexible manageability by overlay, and so on. Since the majority of applications rely on end-to-end TCP sessions, TCP over TCP tunnel is commonly used as well in various cases, although its performance depends on conditions due to the complex interaction between upper-layer TCP and lower-layer TCP [4].

In this paper, a new solution called TCP/NC tunnel system to mitigate the end-to-end TCP performance degradation in lossy networks is proposed. TCP/NC tunnel system consists

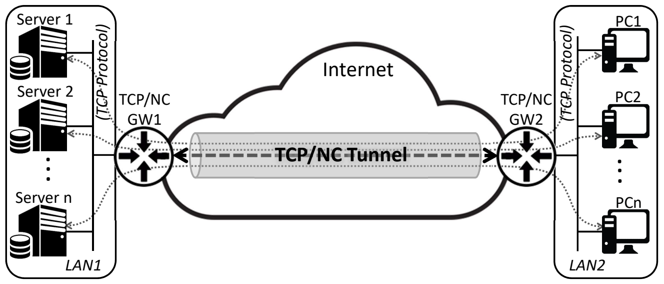


Fig. 2: Example of TCP/NC tunnel

of two gateways running TCP/NC protocol called TCP/NC gateway, which conveys end-to-end TCP sessions on a single TCP/NC session traversing a lossy network in the middle. A simple example of TCP/NC tunnel system is shown in Fig. 2 which consists of two local networks (LAN1 and LAN2) and a lossy network in the middle. All the data transferred between LAN1 and LAN2 through the lossy network are encoded and decoded at GW1 and GW2 to mask the packet losses happening on the lossy network.

In TCP/NC tunnel system, the TCP/NC gateway receives an IP packet with TCP segment from an end-host which runs the original TCP protocol. Then the gateway encapsulates this packet, and forwards it to the remote TCP/NC gateway through TCP/NC tunnel with the packet loss recovery capability. After recovering the packet if necessary, the remote TCP/NC gateway forwards it to an end-host as the original IP packet. TCP/NC gateways do not interfere to the TCP establishment phase as well as the ACK returning process between TCP end-hosts. When packet losses happen in local networks (before or after the tunnel), end-to-end TCP manage the lost packet recovery by a simple retransmission. TCP/NC tunnel system has been implemented and validated in Network Simulator 3. In the proposed system, to eliminate some limitations of the original TCP/NC, a reinforced version of TCP/NC is used, which includes a dynamic estimation and change of NC-related parameters (TCP/NCwLRE [5]) and an efficient retransmission of unrecoverable lost packets (TCP/NCwER [6]). They were previously developed by the authors as described later in Section II-B.

In contrast to PEP approach, the proposed "tunneling" approach does not require complicated per-flow management on each gateway. On the other hand, the tunneling approach must involve encapsulation overhead (e.g., header space and processing time) in general. In addition, the problem of TCP over TCP tunnel should be taken into consideration. As shown in latter sections, by introducing TCP/NC tunnel, the goodput of original TCP sessions across an intermediate lossy network between end-hosts without any change can be significantly improved in a wide range of packet loss rates on the lossy network.

The remainder of this paper is organized as follows. In Section II, TCP/NC is briefly described. The details of proposed TCP/NC tunnel is presented in Section III. Simulations and results are described in Section IV and the conclusions are discussed in Section V.

II. Overview of TCP/NC

A. TCP/NC scheme

TCP/NC protocol was presented in 2008 [1] which successfully implemented the NC into the protocol stack with a minor change by adding the NC layer between the TCP and IP layer, as shown in Fig. 1. The sender-side NC layer 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 on a certain Galois Field. If the number of the lost combination packets is no more than $k = m - n$, the sink-side NC layer is expected to recover all the original packets using the remaining combination packets without retransmission. Therefore, TCP layer is unaware of light loss events occurring and maintains the CWND appropriately to improve the goodput performance. The processes of creating m combination packets and regenerating n original packets are called encoding and decoding, respectively.

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

Besides executing the encoding/decoding process, NC layer allows a new interpretation of ACKs by using the degree of freedom concept and the seen/ unseen definition [7]. The ACK number in the ACK packet is set to the sequence number of the oldest "unseen" packet, which will be decoded when the sink receives the additional combination packets. The example of the coding process is shown in Fig. 3. The packets p_1, p_2, p_3 and p_4 are encoded to the combination packet $C[1]$, $C[2]$, $C[3]$, $C[4]$, $C[5]$ and $C[6]$. When a new packet comes to NC layer, the combination packets will be created and transported immediately. Due to the two lost combinations, the NC layer cannot decode any combination packets until receiving the combination packet $C[6]$. For each received combination packets, NC layer returns an ACK packet whose ACK number corresponds to the smallest "unseen" packet. During the process, the TCP layer totally unawares with any loss events; thus, the CWND keeps increasing and the performance is stable.

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 itself a linear combination involving only packets that arrived after p at the sender.*

If the number of lost combination packets exceed the recovery capability, one or some packets will be "unseen" in all received combination packets. Then TCP layer will receive duplicate ACK numbers from NC layer and retransmit the "unseen" packets to NC layer; NC layer simply forwards them to the lower layer, i.e., IP layer.

B. TCP/NCwLRE and TCP/NCwER

The following two problems hinder the potential of the original TCP/NC. The first one is about how to appropriately choose the NC-related parameters and how to change them in an online fashion. Two basic parameters affecting the efficiency of TCP/NC are the redundancy factor ($R = \frac{m}{n}$) and the recovery capacity (k), which should be chosen based on

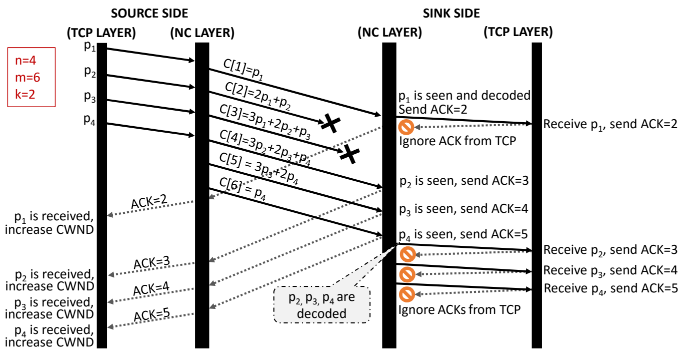


Fig. 3: NC process

TABLE I: Protocol terminology

Term	Definition
TCP NewReno	The basic TCP protocol
The original TCP/NC	The original TCP/NC based on [1]
TCP/NCwLRE	TCP/NC with Loss Rate Estimation [5]
TCP/NCwER	TCP/NC with Enhanced Retransmission [6]
TCP/NC or reinforced TCP/NC	Combine TCP/NCwLRE and TCP/NCwER

the link loss rate of the channel. In addition, those parameters should be able to change at any time if required. As the original TCP/NC does not provide any means for them, TCP/NCwLRE [5] was developed which includes mechanisms to estimate the link loss rate, choose an appropriate redundancy factor R based on the estimated link loss rate, and apply it without any adverse impact on the current encode/transmission/decode process. More specifically, after TCP/NCwLRE estimates the link loss rate r, it calculates the success probability of transmission in one coding window in case of random loss channels (its size is equal to n), $S(n,k)$, by using Eq. (2) with n and k as variable. In this study for TCP/NC tunnel, only the random loss channel is considered; thus, k=3 and the maximum value of n is chosen to satisfy $S(n,3)\geq 0.9$.

$$S(n,k) = \sum_{j=0}^{k} \binom{n+k}{j} r^j (1-r)^{n+k-j} \qquad (2)$$

The second problem is about retransmission of the unrecoverable lost packets. In general, TCP/NC relies on the TCP layer for the packet retransmission that are not recovered by redundant combination packets in NC layer. However, the original TCP/NC cannot simply accommodate an efficient retransmission of TCP layer such as Selective-ACK (SACK), and the lost packets are retransmitted one by one in each round trip time. In response to this problem, TCP/NCwER [6] was developed in which NC layer helps retransmission in an efficient way and the retransmitted packets are also encoded.

In the TCP/NC tunnel system, therefore, the reinforced version of TCP/NC is used instead of the original TCP/NC. The terminologies of the protocols are shown in Table I.

III. TCP/NC TUNNEL

As discussed in Section 1, although TCP performance on lossy networks is expected to be improved by introducing

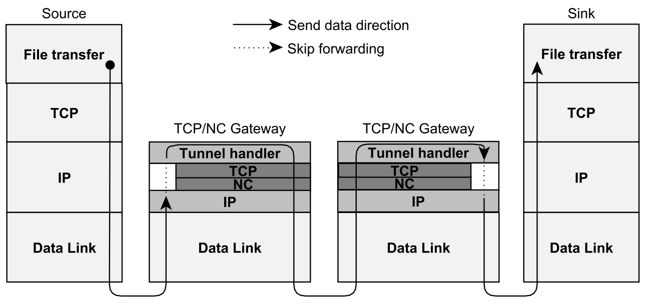


Fig. 4: Tunnel handler

TCP/NC, the deployment of TCP/NC is a challenging task with several difficulties. To avoid those difficulties and utilize the potential of TCP/NC, a simple but effective solution called TCP/NC tunnel is proposed. This proposal only requires the special TCP/NC gateways at the border of each networks which run the specific application called tunnel handler. They communicate together through lossy network using the TCP/NC protocol. In this paper, to make validation and evaluation simple, only one direction data transfer and only packet losses on this direction are considered; the returning ACK process between the sinks and the sources is transparent with the tunnel handler. In this condition, the TCP/NC gateway aggregates and transfers end-to-end TCP sessions between end-hosts into a single TCP/NC connection. Moreover, we focus on the application of transfering a large file in this paper; hence, only goodput performance is evaluated.

A. The operation of the TCP/NC gateway

The TCP/NC gateway equips two interface types, an internal interface and an external interface to distinguish the packets which receive in the local network or external network. The internal interface is connected to the local network and the external interface is connected to the external network. The protocol stack and structure of TCP/NC tunneling is illustrated in Fig. 4 and the packet processing at the TCP/NC gateway is shown in Fig. 5. When the IP packets from the end-hosts arrive at internal interfaces, they are moved to the tunnel handler to become the transferred data and forwarded to TCP layer and to NC layer. At NC layer, all the segments are encoded to the combinations on a TCP/NC session and sent to the remote gateway. When the combinations arrive at external interface of the remote gateway, the decoding process is performed by NC layer to recover the lost combinations if needed. A new decoded packet is forwarded to TCP layer for reordering. The data of the packet in the correct sequence in terms of tunnel TCP session is pushed to the tunnel handler. The tunnel handler converts the received data to an original TCP segment to be sent to the sink based on the IP address of the data. Finally, the sink receives the packet and returns an ACK packet without data to the source, which is transparently forwarded through the TCP/NC tunnel because only the single directional data transfer is considered. In case of the bi-directional data transfer, the system needs to handle an ACK packet with data on an opposite directional TCP/NC tunnel. However in any case, the system can maintain the end-to-end TCP ACK semantics for the retransmission process by end-hosts responsible to packet losses that happen outside the tunnel.

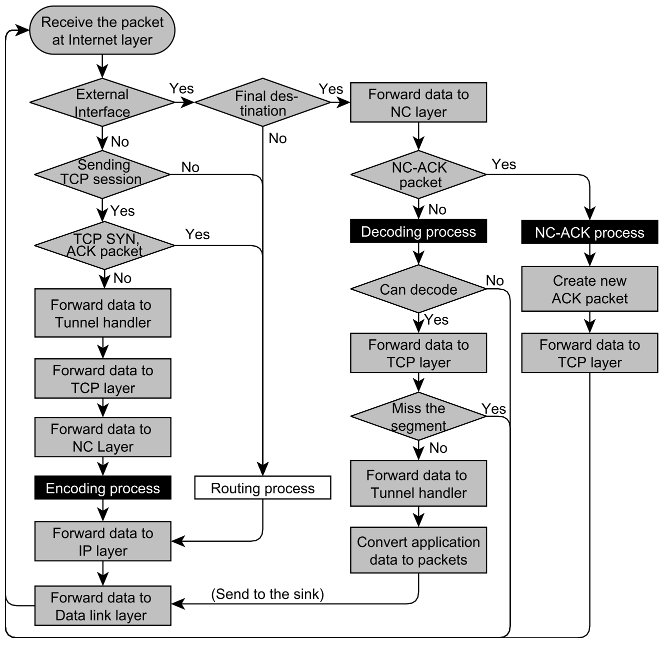


Fig. 5: Processing at TCP/NC gateway

B. The congestion control

The TCP/NC tunnel system includes three different buffers in which some kind of congestion may happen. The first one is the TCP sending buffer of the gateway GW1 (in Fig. 6) that accumulates all packets from end-to-end TCP sources and keeps on-the-fly packets for TCP retransmission. Even if the total incoming throughput at GW1 and the actual throughput from GW1 to GW2 are less than the intermediate bandwidth, the congestion can still occur, e.g., when GW1 cannot release the non-ACKed packets from the TCP sending buffer due to packet loss. The second one is the link buffer of external interface of GW1. The incoming packets may be amplified by encoding process to the combinations for redundancy when being forwarded into the TCP/NC tunnel. If the amplified data exceeds the bandwidth of the intermediate link, the congestion will happen. The last one is the link buffer of the internal interfaces of GW2. A number of lost packets belonging to the same end-to-end TCP session might be burstly recovered and forwarded, which may cause congestion.

The first case in which frequent congestion will impact the performance should be managed. The TCP sending buffer size is set to 64 KB plus the link buffer size of the external interface. Note that, the actual congestion window size is limited up to 64 KB because the TCP window scaling option is not used. In the preliminary comparison, a very large size of the TCP sending buffer decreased the goodput of end-to-end TCP sessions.

Another issue to be considered is the TCP retransmission timeout (RTO). In a heavily lossy network or network congestion, a retransmission timeout is unavoidable. If the RTO timer of an end-to-end session is smaller than that of the TCP/NC session, the end-to-end retransmission of a packet happens before the TCP/NC tunnel will recover the packet by in-tunnel retransmission. In such cases, the same packets may be stored in the TCP sending buffer multiple times. Therefore, the minimum RTO timer value of the TCP/NC tunnel session is set to a value (400 ms) less than a normal value (1000 ms) that is used for the end-to-end TCP session.

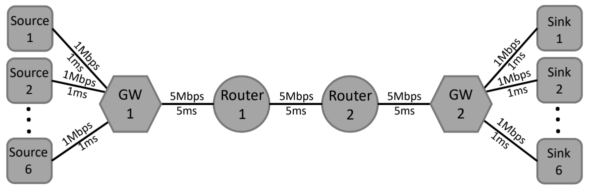


Fig. 6: Network topology

IV. Simulation and result

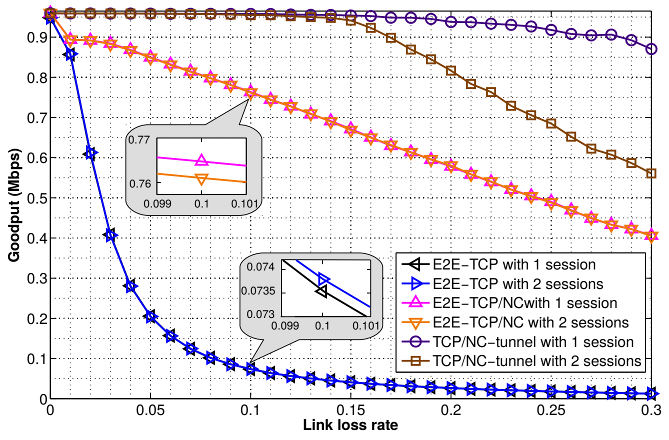
The implementation of TCP/NC tunnel was accomplished using Network Simulator 3 (ns-3) [8]. The topology of the simulation is a tandem network with two routers/gateways connect to at most five sources and five sinks, as shown in Fig. 6. Each edge link connects an end-host and a router has a bandwidth of 1 Mbps and a propagation delay of 1 ms. The intermediate link connecting two routers has a bandwidth of 4 Mbps and a propagation delay of 7 ms. The link buffer size is set to 100 packets, and the packet size is 1000 bytes. The size of TCP sending buffer in TCP/NC gateway is 164 packets. The transferred data size is 100 Mbytes. The TCP type is NewReno. The intermediate link is considered as lossy channel of random loss channel with a link loss rate ranging from 0.0 to 0.3. These losses happen only in the direction of transferred data. In all scenarios, each simulation was performed at least 20 times to obtain the average value.

The simulation is run in four cases, which has one, two, three and five active sessions. The sessions are started at the same time with the same data size and run the same protocol. There are three protocols which are used to compare. In the first case, all the sources and the sinks run TCP (E2E-TCP case). In the second case, all the sources and the sinks run TCP/NC (E2E-TCP/NC case). In the last case, all the sources and the sinks run TCP but over TCP/NC tunnel by configuring two TCP/NC gateways (TCP/NC tunnel case). The simulation result is shown in Fig. 7, the goodput performance is calculated by the average of the goodput of all sessions.

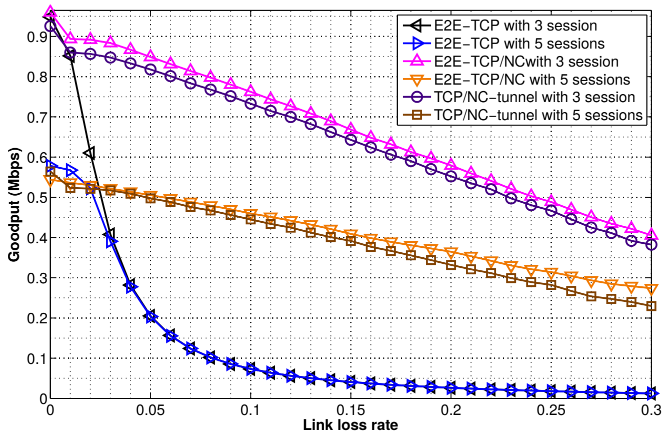
With packet loss recovery capacity, the goodput of E2E-TCP/NC and TCP/NC tunnel is always better than that of E2E-TCP over the lossy networks. If the number of sessions is less than four, the goodput of E2E-TCP/NC does not change because the bandwidth of intermediate link (4 Mbps) is always larger than the total throughput of three sessions (3 Mbps).

In E2E-TCP/NC case, as the link loss rate increases, the number of packet losses likely exceeds the recovery capacity of NC layer. Thus, the TCP layer needs to retransmit the “unseen” packets. The CWND is decreased and the goodput performance is also decreased.

By comparing E2E-TCP/NC and TCP/NC tunnel in goodput performance, two situations are seen. In cases of one or two end-to-end sessions, the performance of TCP/NC tunnel is clearly better than that of E2E-TCP/NC. However, in cases

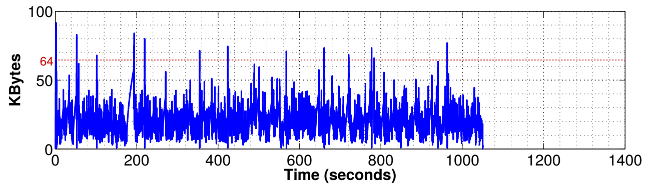


(a) One session and two sessions

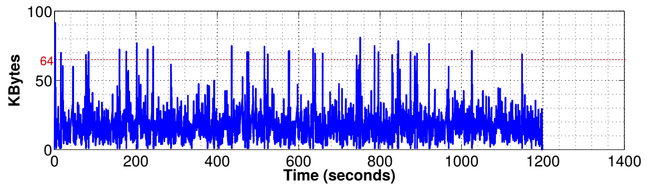


(b) Three sessions and five sessions

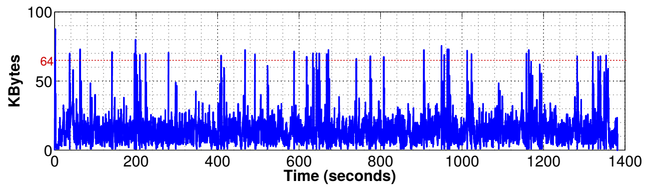
Fig. 7: Goodput comparison



(a) At link loss rate of 0.10



(b) At link loss rate of 0.15



(c) At link loss rate of 0.20

Fig. 8: End-to-end CWND evolution of one of two coexisting E2E-TCP/NC sessions

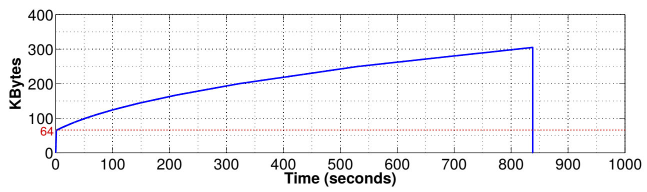
of three or five end-to-end sessions, the performance of E2E-TCP/NC is slightly better than that of TCP/NC tunnel.

In the case of one session, the bandwidth of intermediate link (4 Mbps) is sufficient to convey one end-to-end session's throughput amplified by NC up to the loss rate of 0.3. In the TCP/NC tunnel, therefore, no congestion happens in the TCP sending buffer at TCP/NC gateway. All the lost packets are successfully recovered and received in the end-to-end TCP sink before the end-to-end TCP source retransmits the packet by TCP timeout; thus, the source does not know the loss event and keeps a high CWND to get a high goodput performance.

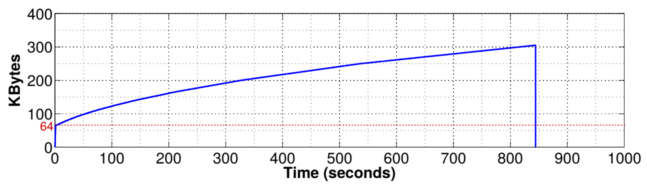
In the case of two sessions, congestion at the TCP/NC tunnel can happen. When the link loss rate is less than 0.15, all losses can be recovered before the TCP sending buffer is overflow. However, when the link loss rate is greater than 0.15, the number of retransmissions is increased, resulting in the shrinking CWND of TCP/NC session on the tunnel while the CWND of each end-to-end TCP session keeps increasing. This issue makes the TCP sending buffer be increased when the packet loss happens. There are two problems when this buffer increases. First, due to an overflow of this buffer, the newly incoming packets will be dropped. Since these packets cannot be recovered by TCP/NC tunnel, the end-to-end TCP source should recognize these lost packets by TCP timeout to retransmit them. The CWND at the end-to-end TCP source decreases to one, resulting in the goodput performance degradation. Second, due to a long waiting time in this buffer, the number of dead packets increases. The dead packet is the packet which is still stored in the TCP sending buffer but is expired by TCP timeout. The CWND at the end-to-end source decreases to one again and "unnecessary" retransitions happen. Fig. 8 and Fig. 9 show the end-to-end CWND of one of two sources in E2E-TCP/NC case with two sessions and in TCP/NC tunnel case with two sessions, respectively, at link loss rate of 0.1, 0.15 and 0.2. The CWND of the sources in TCP/NC tunnel case is increased in the link loss rate 0.1 and 0.15 but it is decreased at some times in the link loss rate of 0.2 while the CWND of the sources in E2E-TCP/NC is decreased many times in all cases of link loss rate. Fig. 10 shows the TCP sending buffer evolution of the TCP/NC gateway. At the high link loss rate (0.2), the queue length of the TCP sending buffer of TCP/NC gateway is almost always longer than the upper limit of its actual TCP congestion window size (around 64 packets). This suggests many packets (i.e., dead packets) are stored in the TCP sending buffer and retransmitted by the source due to end-to-end TCP timeout.

In the case of three sessions, the congestion on the TCP/NC tunnel happens more often. It decreases the goodput performance of TCP/NC tunnel to nearly the same level of E2E-TCP/NC. More precisely, an encapsulation overhead (4%) of TCP/NC tunnel makes its goodput slightly lower compared with E2E-TCP/NC.

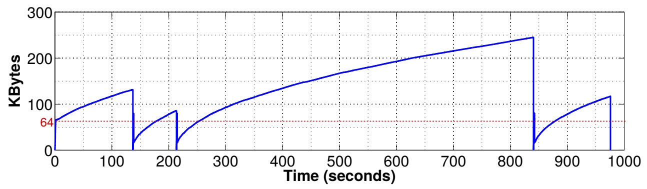
In case of five sessions, the congestion always happens



(a) At link loss rate of 0.10

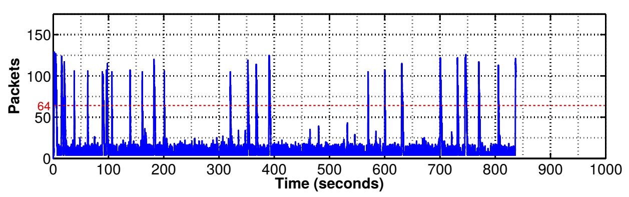


(b) At link loss rate of 0.15

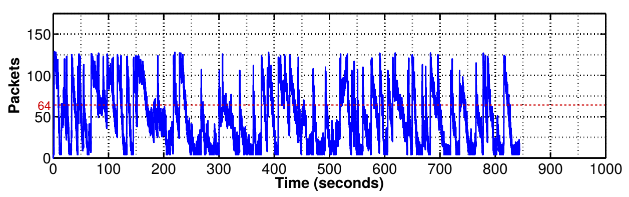


(c) At link loss rate of 0.20

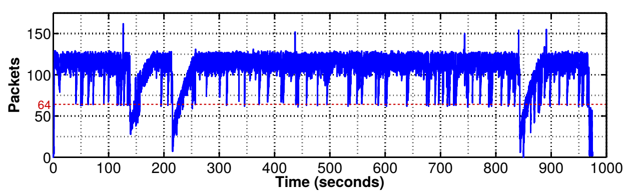
Fig. 9: End-to-end CWND evolution of one of two coexisting TCP sessions on the TCP/NC tunnel



(a) At link loss rate of 0.10



(b) At link loss rate of 0.15



(c) At link loss rate of 0.20

Fig. 10: TCP sending buffer evolution at TCP/NC gateway in the case of 2 sessions

even with non-lossy links. Both of E2E-TCP/NC and TCP/NC tunnel exhibit a similar goodput degradation.

V. Conclusions

In this paper, the TCP/NC tunnel system, consisting of two TCP/NC gateways that run TCP/NC protocol, has been proposed. The TCP/NC tunnel conveys end-to-end TCP sessions on a single TCP/NC session between the gateways traversing a lossy network in the middle. TCP end-hosts can use this tunnel to take advantage of the recovery capacity of NC without running TCP/NC on each end-host.

The simulation results on ns-3 show that the proposed TCP/NC tunnel achieves a better performance in goodput compared to the E2E-TCP and a comparable performance to the E2E-TCP/NC when traversing a lossy network. In addition, if the TCP sending buffer in the TCP/NC gateway is not congested, the TCP/NC tunnel gets a higher goodput than E2E-TCP/NC. Only if the bandwidth of intermediate link (i.e., the lossy network) is insufficient, the tunneling overhead causes a lightly smaller goodput compared to E2E-TCP/NC case.

We simply assume TCP/NC gateways are deployed at the edges of the targeted lossy network and work collaboratively for greedy end-to-end TCP sessions. Although the proposed system and the performance evaluation are preliminary and with limited assumptions, the results suggest the potential of the TCP/NC tunnel that can provide a goodput performance of end-to-end TCP sessions comparable to E2E-TCP/NC with no change at the end-hosts as well as no per-flow management at the gateways. Since TCP/NC tunnel may not necessarily be good for all types of applications on end-to-end TCP sessions, discriminating session types and selecting tunneling types (TCP/NC tunnel or other ones) should be one of our future work.

VI. Acknowledgments

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," in Proc. *IEEE Intertional conference on Computer Comunication (INFOCOM),* Rio de Janeiro, Brazil, no. 2, pp. 280–288, April 2009.

[2] M. Ivanovich, P. Bickerdike and J. Li, "On TCP performance enhancing proxies in a wireless environment," *IEEE Communications Magazine,* vol. 46, no. 9, pp. 76–83, 2008.

[3] D. Murray, T. Koziniec and M. Dixon, "D-Proxy: Reliability in wireless networks," in Proc. *16th Asia-Pacific Conference on Communications (APCC),* Auckland, New Zealand, pp. 129–134, 2010.

[4] O. Honda, H. Ohsaki, M. Imase and J. Murayama, "Understanding TCP over TCP: Effects of TCP Tunneling on End-to-End Throughput and Latency," in Proc. *SPIE,* vol. 6011, 9 pages, October 2005.

[5] N. V. Ha, K. Kumazoe and M. Tsuru, "Making TCP/NC adjustable to time varying loss rates," in Proc. *Proc. 8th International Conference on Intelligent Networking and Collaborative Systems (INCoS),* Ostrava, Czech Republic, pp. 457–462, September 2016.

[6] N. V. Ha, K. Kumazoe and M. Tsuru, "TCP Network Coding with Enhanced Retransmission for heavy and bursty loss," *IEICE Transaction of Communications,* vol. E100-B, no. 2, pp. 293–303, February 2017.

[7] J. K. Sundararajan, D. Shah and M. Medard, "ARQ for network coding," in *Proc. IEEE International Symposium on Information Theory (ISIT),* Toronto, Canada, pp. 1651–1655, July 2008.

[8] "Network simulator (ns-3)," in *https://www.nsnam.org/*, accessed Mar. 1. 2016.