Benefits of Multiply-cascaded TCP Tunnel with Network Coding over Lossy Networks

Nguyen Viet Ha, Kazumi Kumazoe, Kazuya Tsukamoto and Masato Tsuru

Kyushu Institute of Technology, Japan

Abstract. Transmission Control Protocol (TCP) is still dominant for reliable end-to-end data transmission with congestion control over diverse types of networks although it does not perform well in goodput on lossy networks. To mitigate the goodput degradation of TCP on lossy networks, TCP with Network Coding (TCP/NC) was proposed. But it has not been well deployed because TCP/NC should be implemented in both sides of end-to-end connection; it requires considerable costs and is sometimes difficult in tiny end devices, e.g., with less memory and power. In this paper, to utilize the potential of TCP/NC more practically with no change on end-host TCP, we consider the TCP/NC tunnel that simply conveys end-to-end TCP sessions not only on a single TCP/NC session but also on cascaded TCP/NC sessions traversing a lossy network in the middle without per-session management. The simulation results by Network Simulator 3 clearly show the benefit of the multiply-cascaded TCP/NC tunnel. In congestion scenarios with a wide range of link loss rates, the end-to-end standard TCP with multi-cascaded TCP/NC tunnel can achieve a significantly higher goodput compared to both the end-to-end TCP/NC without tunnel and the end-to-end standard TCP with single TCP/NC tunnel.

1 Introduction

Transmission Control Protocol (TCP) is ineffective in lossy networks, e.g., wireless networks [1]. TCP always cuts down the sending rate for each loss because it recognizes all loss are interpreted as a congestion signal. To mitigate this performance degradation issue without change of end-host TCP, a variety of Performance Enhanced Proxy (PEP) approaches 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 session management on the proxy nodes. From a different viewpoint, IP over TCP (we call it TCP tunnel in this paper) 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 session aggregation, flexible manageability by overlay, and so on. Since many applications rely on end-to-end TCP sessions, TCP over TCP tunnel can be seen commonly in

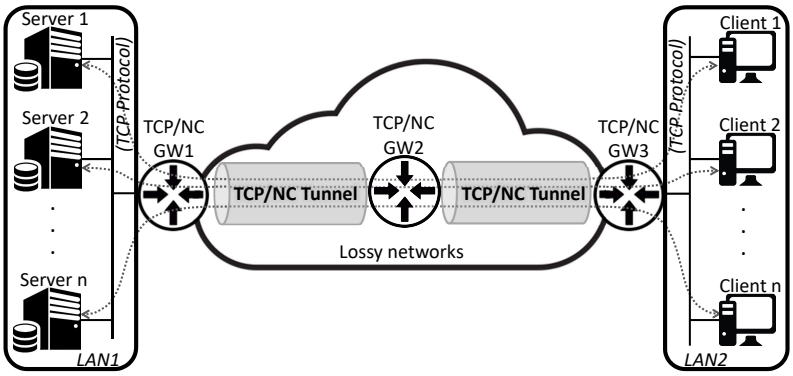


Fig. 1. The example of TCP/NC tunnel with two TCP/NC Gateways

various cases, although its performance depends on conditions due to the complex interaction between upper-layer TCP and lower-layer TCP [4]. In addition, since TCP itself is ineffective as mentioned above, TCP tunnel also does not perform well in lossy networks.

As an alternate solution for TCP goodput degradation in lossy networks, TCP with Network Coding (TCP/NC) was presented in 2009 [5]. The sink recovers all data even though some of packets are lost by allowing the source to send the data as random linear NC combination packets (referred to as combination packets) to the sink. Although TCP/NC is shown to have certain benefits in lossy networks, TCP/NC is not easy to be deployed because of several reasons. First is the incompatibility with the existing TCP protocol; thus, TCP/NC is required to be implemented at both ends of connection. Second, TCP/NC is costly to be implemented in some end devices e.g., with less memory and power.

To take advantage of TCP tunnel and TCP/NC, we propose TCP/NC tunnel system to mitigate the end-to-end TCP performance degradation in lossy networks without any change at end-host TCP. TCP/NC tunnel system consists of at least two gateways running TCP/NC protocol called TCP/NC gateways. In this paper, by extending the simplest TCP/NC tunnel with two gateways [6], we develop and evaluate the multi-cascaded TCP/NC tunnel which convey end-to-end TCP sessions on multi-cascaded TCP/NC sessions traversing multiple loss channels in the middle. An example of TCP/NC tunnel system is shown in Fig. 1. Data transmissions by end-to-end TCP sessions between LAN1 and LAN2 are encoded, decoded, and/or retransmitted at GW1, GW2, and GW3 to mask the packet losses happening in the lossy networks.

In TCP/NC tunnel system, adjacent TCP/NC gateways establish TCP/NC sessions together which can have the different NC-related parameters (NC parameters) based on the condition of each loss channel. The TCP/NC gateway of the source side (e.g., GW1 in Fig. 1) receives an IP packet from an end-host which runs the original TCP protocol. After that, the gateway encapsulates this packet, and forwards it to the next TCP/NC gateway through TCP/NC tunnel

with the packet loss recovery capability. The TCP/NC gateway of the sink side (e.g., GW3 in Fig. 1) forwards the received packet to an end-host as original IP packet after recovering the packet if necessary. Besides, some intermediate TCP/NC gateways (e.g., GW2 in Fig. 1) can be placed in the middle which has different lossy networks. The TCP/NC gateways do not interfere to the TCP establishment phase as well as the ACK returning process among TCP end-hosts. When the packet losses happen in local networks (before or after the tunnel), the end-to-end TCP manages the lost packet recovery by a simple retransmission.

In contrast to PEP approach, the proposed "tunneling" approach does not require a complicated per-session management on each gateway. On the other hand, the tunneling approach must involve the 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.

TCP/NC tunnel system has been implemented and validated in Network Simulator 3. In the proposed system, to eliminate some limitations of original TCP/NC, a reinforced version of TCP/NC is used, which includes a dynamic estimation and a change of NC parameters (TCP/NCwLRLBE [7]) and an efficient retransmission of unrecoverable lost packets (TCP/NCwER [8]). They were previously developed by the authors as reviewed later in Section 2.2.

The remainder of this paper is organized as follows. In Section 2, TCP/NC is briefly described. The details of proposed TCP/NC tunnel is presented in Section 3. Simulations and results are described in Section 4 and the conclusions are discussed in Section 5.

2 Overview of TCP/NC

2.1 TCP/NC scheme

TCP/NC protocol [5] was proposed to implement a NC into the protocol stack with a minor change by adding a new NC layer between TCP and IP layer. The sender-side NC layer allows the source to send m combination packets (C) created from n original packets (p) ($m \geq n$) using Eq. (1) where α is the coefficient on a certain Galois Field, e.g., GF(2^8). When 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 loss events and maintains the Congestion Window (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. ACK number in ACK packet is set to the sequence number

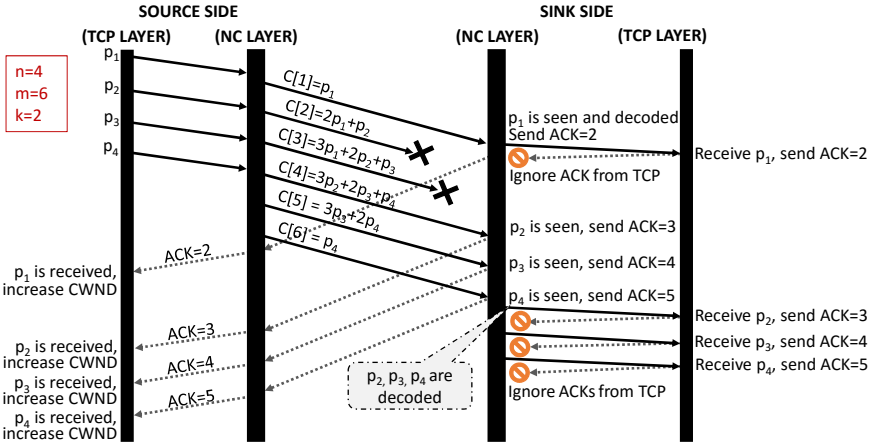


Fig. 2. NC process example

of the oldest "unseen" packet, which can be decoded when the sink receives the additional combination packets. Example of the coding process is shown in Fig. 2. 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, TCP layer totally unawares with any loss events; thus, the CWND keeps increasing and the performance is stable.

If the number of lost combination packets exceeds 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 lower layer, i.e., IP layer.

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.*

2.2 Reinforced version of TCP/NC

The TCP/NCwLRLBE was developed in [7] to solve the issues of the original TCP/NC which are how to choose the appropriate NC parameters (n and m) and how to change them in an online fashion. TCP/NCwLRLBE considers the channel burst loss channel; thus, the link loss rate (r) and the length of continuous losses (l) are estimated from continuous observation of the packet

transmission between the source and the sink. The probability of successful recoverable transmission (Pr) for each pair n and m is calculated based on the loss rate, loss burstiness conditions. Finally, n and m are selected based on Pr which is minimum value and greater than or equal 0.9.

Another problem is retransmission of the unrecoverable lost packets. In the original TCP/NC, the packets which cannot recovered by redundant combination packets in NC layer at the sink will be retransmitted by TCP layer at the source. The retransmitted packets are transported one by one in each Round Trip Time (RTT), resulting in a poor goodput performance. In response to this problem, TCP/NCwER [8] was developed in which NC layer helps retransmission in an efficient way. Multiple lost packet can be retransmitted in one RTT and the all of them are also encoded to avoid the repeated loss. In TCP/NC tunnel system, therefore, the reinforced version of TCP/NC, which is a combination of TCP/NCwLRLBE and TCP/NCwER, is used instead of the original TCP/NC.

3 Masking packet losses with Cascaded TCP/NC Tunnel

The TCP/NC tunnel system requires at least two special TCP/NC gateways at the border of each networks which run the specific application called tunnel handler. TCP/NC gateways can also be installed in the middle networks to increase the performance. The TCP/NC gateways at border aggregate and transfer end-to-end TCP sessions between end-hosts into a single TCP/NC session or cascaded TCP/NC sessions. TCP/NC tunnel system does not interfere the returning ACK process between the sinks and the sources; thus, this process is completely transparent with the tunnel handler. Note that, only one direction data transfer and only packet losses on this direction are considered in this paper to make validation and evaluation simple.

We investigated the benefit of single TCP/NC tunnel [6], i.e., with only two TCP/NC gateways, and showed that it could support tiny end devices which are unable to run TCP/NC on a lossy network. More specifically, in a wide range of link loss rates, the standard end-to-end TCP with single TCP/NC tunnel (Single TCP/NC tunnel) outperforms in goodput significantly compared to the standard end-to-end TCP without tunnel (E2E-TCP) and performed comparably to the end-to-end TCP/NC without tunnel (E2E-TCP/NC). On the other hand, in this paper, we focus on more a general case, i.e., multi-cascaded TCP/NC tunnel with more than two TCP/NC gateways, and will show that the standard end-to-end TCP with multi-cascaded TCP/NC tunnel can significantly outperform Single TCP/NC tunnel and E2E-TCP/NC in congestion cases on a lossy network. Moreover, we focus on the application of a large file transfer in this paper; hence, we evaluate the goodput performance only. Consideration on the end-to-end delay for other applications will be our future work.

3.1 The operation of the TCP/NC gateway

There are two type of TCP/NC gateway. Border TCP/NC gateway connects to the local networks and the intermediate TCP/NC gateway located in the

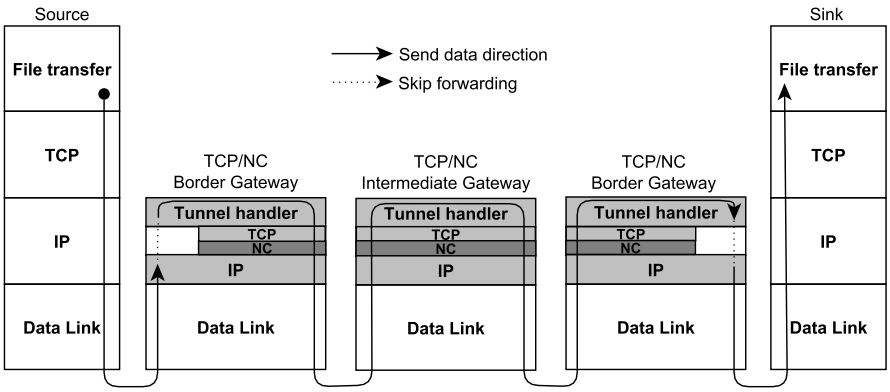


Fig. 3. Tunnel handler

middle. Each TCP/NC gateway can be equipped 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. Consequently, the intermediate TCP/NC gateway has only external interfaces.

The protocol stack and structure of TCP/NC tunneling are illustrated in Fig. 3 and the packet processing at the TCP/NC gateway is shown in Fig. 4. When an IP data packet from end-hosts arrives at internal interfaces of the border TCP/NC gateway, it is moved to the tunnel handler to become the transferred data to forward to TCP layer and NC layer. At NC layer, all the segments are encoded to the combination packets and sent to an adjacent TCP/NC gateway via a TCP/NC session. When the combination packets arrive at the external interface of the adjacent TCP/NC 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. Data of the packet in the correct sequence in terms of tunnel TCP session is pushed to the tunnel handler. After that, if this is an intermediate TCP/NC gateway, the tunnel handler converts the received data to the packets and sends to the next TCP/NC gateway into another TCP/NC tunnel. This process is like the process at the border TCP/NC gateway described above. Otherwise, this is a border TCP/NC gateway. The tunnel handler converts the received data to an original TCP segment to be sent to the sink based on IP address of the data. Finally, when the sink receives the packet, it returns an ACK packet with null 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.

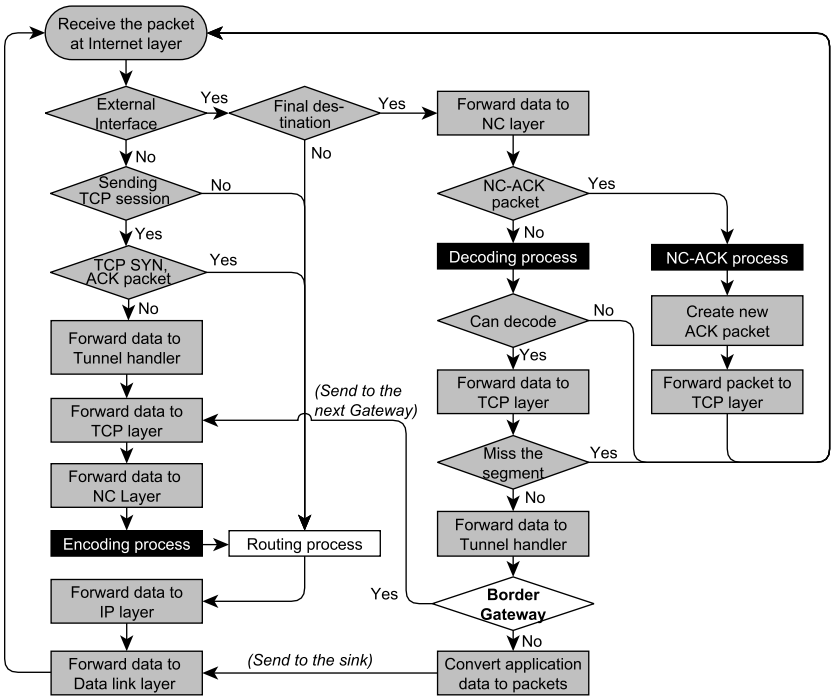


Fig. 4. The processing at TCP/NC gateway

3.2 Congestion control

The TCP/NC gateway includes two different buffers in which some type of congestion can happen. Those are TCP sending buffer and link buffer. First, the TCP sending buffer of the TCP/NC gateways (e.g., GW1 and GW2 in Fig. 1) accumulates all packets from end-to-end TCP sources or the adjacent TCP/NC gateway and keeps on-the-fly packets for TCP retransmission. Therefore, when the non-ACKed packets cannot be released from the TCP sending buffer due to packet loss, the congestion can still occur, even if the total incoming throughput into GW1 and the incoming throughput into GW2 are less than the bandwidth of the inter-gateway link. Second, the incoming packets can be amplified to the combinations for redundancy by encoding process when TCP/NC gateways communicate together via TCP/NC tunnel. If the amplified data exceeds the bandwidth of the intermediate link, the congestion will happen at the link buffer of external interface of the GW1 and GW2. Besides, the link buffer of the internal interfaces of GW3 can be congested if a number of lost packets belonging to the same end-to-end TCP session are burstly recovered and forwarded.

To mitigate congestions in the first case that will impact the performance, 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 this paper. In the

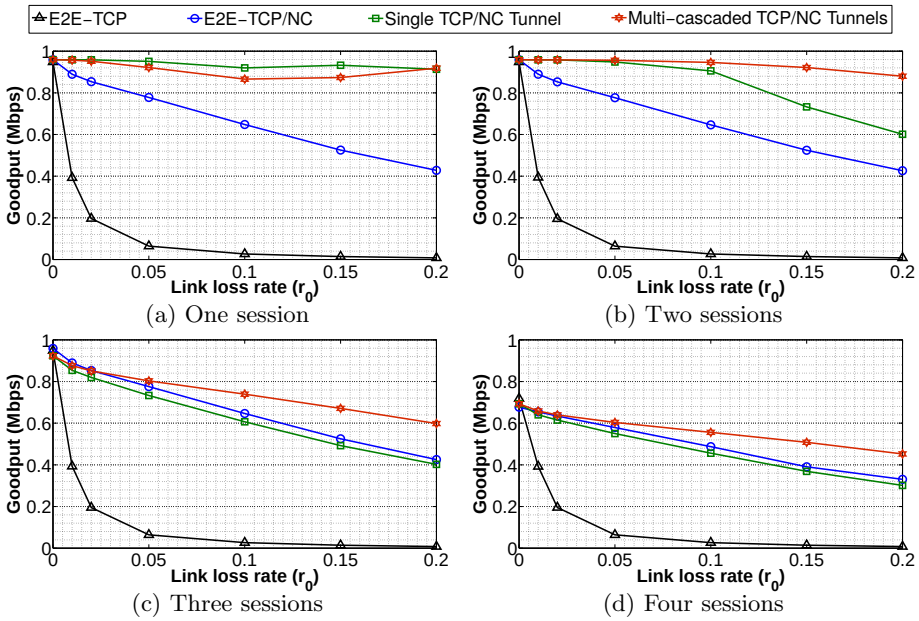
Fig. 5. The network topology

preliminary comparison, a very large size of the TCP sending buffer decreases the goodput of end-to-end TCP sessions.

Another issue to be considered is the TCP retransmission timeout (RTO). Retransmission timeout is unavoidable in heavily lossy network or network congestion. 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 recovers the packet by in-tunnel retransmission. In such cases, the same packets are stored in the TCP sending buffer multiple times (dead packets). Therefore, the minimum RTO timer value of each TCP/NC tunnel session should be set so that the sum of those values is less than the value that is used for each end-to-end TCP session. On the other hand, the minimum RTO value of each TCP/NC tunnel should not be too small to avoid unnecessary time out. From those conditions, the maximum number of cascaded gateways is limited based on the minimum RTO requirement of each application. To reduce the deployment cost, a small number of gateways is also important. However, the optimal number of gateways remains as future work. In the simulation below, the minimum RTO of all the TCP/NC tunnel sessions is set to 400 ms and the minimum RTO of end-to-end TCP session is set to 1000 ms.

4 Simulation and result

The simulation of TCP/NC tunnel was accomplished using Network Simulator 3 (ns-3) [9]. The topology is a tandem network with three routers/gateways connect to at most four sources and four sinks, as shown in Fig. 5. Each edge link connects an end-host and a router has a bandwidth of 1 Mbps and a propagation delay of 5 ms. The intermediate links connecting between routers has a bandwidth of 3 Mbps and a propagation delay of 10 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. Losses happen at GW2 and GW3 in the direction of transferred data. The link loss rate in each link is set from 0.0 to 0.3 (r_0). Therefore, the total link loss rate from GW1 to GW3 equals $r_0+r_0\times(1-r_0)$. Note that, the



(a) One session (b) Two sessions

(c) Three sessions (d) Four sessions

Fig. 6. The goodput comparison

link loss rate parameter in X-axis in the result figures is r_0. In all scenarios, each simulation was performed 10 times to obtain the average value. The simulation is run in four cases, which has one, two, three and four active sessions. The sessions are started at the same time with the same data size and run the same protocol. There are four protocols which are used to compare:

- E2E-TCP: all sources and sinks run TCP.
- E2E-TCP/NC: all sources and sinks run TCP/NC.
- Signal TCP/NC tunnel: all sources and sinks run TCP but over a single TCP/NC session by configuring two TCP/NC gateways at GW1 and GW3.
- Multi-cascaded TCP/NC tunnel: all sources and sinks run TCP but over two cascaded TCP/NC sessions between GW1 and GW2, GW2 and GW3 by configuring three TCP/NC gateways at GW1, GW2 and GW3.

The simulation result is shown in Fig. 6, the goodput performance is calculated by the average of the goodput of all sessions. Both the congestion and non-congestion cases are considered in this simulation. In all cases, the goodput of E2E-TCP/NC and two types of TCP/NC tunnel with the packet loss recovery capacity are always better than that of E2E-TCP over the lossy networks. Therefore, it is shown that implementing TCP/NC tunnel can mitigate the goodput degradation of TCP sessions on lossy networks without any change on end-host TCP.

E2E-TCP/NC and TCP/NC tunnel are different in the cause of congestion and the reaction to it. In E2E-TCP/NC, the congestion happens when total

throughput of all sessions exceeds the bandwidth of intermediate link. All packet losses caused by congestion are aware and retransmitted by TCP/NC end-host through Triple-Duplicate ACK and TCP timeout mechanisms. In contrast, the congestion in TCP/NC tunnel mainly happens at TCP sending buffer, as investigated in Section 3.2. TCP/NC gateway is unaware of the packets lost by the TCP sending buffer congestion; such lost packets should be retransmitted from TCP end-host after TCP timeout.

The goodput of E2E-TCP/NC does not change in the case of the number of sessions less than four because the total throughput of all end-to-end sessions (from 1 Mbps link) does not exceed 3 Mbps bandwidth of intermediate link. As the link loss rate increases, the number of packet losses likely exceeds the recovery capacity of NC layer, resulting in retransmission and sending rate reduction by TCP layer. Thus, the goodput is decreased. In case of 4 sessions, the congestion happens. NC parameters are also increased to mitigate these losses.

When comparing the goodput between E2E-TCP/NC and Single TCP/NC tunnel, two situations are seen. First, the performance of TCP/NC tunnel is clearly better than that of E2E-TCP/NC in case of one or two end-to-end sessions. This is because TCP/NC tunnel can fully utilize 3 Mbps bandwidth of intermediate link to transmit and retransmit the data including redundancy for one or two end-to-end sessions. Meanwhile, E2E-TCP/NC is limited to 1 Mbps link for each end-to-end session. Second, the performance of E2E-TCP/NC is slightly better than that of Single TCP/NC tunnel in case of three or four end-to-end sessions. In this situation, the congestion on the TCP/NC tunnel happens. This leads to the goodput performance decrease 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 than E2E-TCP/NC.

Next, the goodput performance of Single TCP/NC tunnel and Multi-cascaded TCP/NC tunnel are compared. In cases of one session, the goodput performance are similar in general but some interesting observations are seen. First, in one session case, the goodput of Single TCP/NC tunnel is slightly better than that of Multi-cascaded TCP/NC tunnel. Second, the goodput of Multi-cascaded TCP/NC in two sessions case is better than in one session case. Both issues mainly come from choosing the value of n. In NC process, TCP/NC gateway must receive enough n original packets to finish one encoding process (to generate m combination packets). But the TCP/NC gateway is passive in receiving n original packets because this depends on the end-host or the preceding TCP/NC gateway. In one session case, only one source end-host provides original data. In addition, when it does not receive enough ACK packets due to packet losses, the packet sending rate is decreased. Hence, the number of packets received at TCP/NC gateway becomes sometimes insufficient compared with n and this "incoming packet suspending" happens more frequently with a larger n. In Single TCP/NC tunnel, the gateway of the source side estimates the total link loss rate r to choose n while Multi-cascaded TCP/NC chooses n based on r_0. Thus, n in Single TCP/NC tunnel is less than n in Multi-cascaded TCP/NC. Consequently, the incoming packet suspending less happens in Single TCP/NC tunnel

than Multi-cascaded TCP/NC tunnel. In fact, at link loss rate (r_0) of 0.1, the estimated (n, m) of GW1 in Single TCP/NC tunnel is (25,35) while the estimated (n, m) of GW1 and GW2 are (40,47) and (30,40), respectively in Multi-cascaded TCP/NC tunnel. The packets from GW1 to GW2 do not send consecutively; thus, GW2 does not estimate an accurate (n, m).

In cases of two, three, and four sessions, the goodput advantage of Multi-cascaded TCP/NC tunnel to Single TCP/NC tunnel is clearly shown. In two sessions case with two end-hosts, for example, the probability of continuously and timely receiving enough n original packets is double compared to the one session case. At each TCP/NC gateway, the incoming packet suspending will happen less and an appropriate (n, m) and thus an appropriate redundancy factor $R{=}m/n$ can be estimated. In such cases, Multi-cascaded TCP/NC tunnel is always better than that of E2E-TCP/NC and Single TCP/NC tunnel in goodput simply because the each link loss rate (r_0) to be responded by the former is lower than the total link loss rate (r) to be responded by the latter, which allows the former to use a smaller redundancy factor R. For example, in case of three sessions, the estimated (n, m) of E2E-TCP/NC, Single TCP/NC and Multi-cascaded TCP/NC tunnel are (25,35), (25,35) and (40,47), respectively at r_0 of 0.1. In case of four sessions, the estimated (n, m) of E2E-TCP/NC, Single TCP/NC and Multi-cascaded TCP/NC tunnel are (23,33), (25,35) and (40,47), respectively at r_0 of 0.1. As the loss rate increases, the goodput decreases gradually but the goodput advantage of Multi-cascaded TCP/NC tunnel increases as expected. Note that the goodputs of both Single and Cascaded TCP/NC tunnels with four sessions are significantly lower than those with three sessions because a sever congestion happens even in no link loss case.

In this paper, for simple and fair comparison between Single TCP/NC tunnel and Multi-cascaded TCP/NC tunnel, we used a homogeneous link scenario in which the same bandwidth and the same loss rate are set on the link between GW1 and GW2 and that between GW2 and GW3. Since the Multi-cascaded TCP/NC tunnel can choose suitable NC parameters for each link, it is expected to take more advantage of Multi-cascaded TCP/NC tunnel in case of more heterogeneous links. Therefore, the presented results can be considered as the baseline advantage of Multi-cascaded TCP/NC tunnel.

5 Conclusions

In this paper, the TCP/NC tunnel system was preliminarily implemented and evaluated on ns-3 simulator, which simply conveys end-to-end TCP sessions on a single TCP/NC session between two TCP/NC gateways or cascaded TCP/NC sessions involving more than two TCP/NC gateways traversing a lossy network in the middle without per-session management. The simulation results showed that by using TCP/NC tunnel, TCP end-hosts can take advantage of the recovery capacity of NC without running TCP/NC on each end-host.

In particular, we showed the benefit of multi-cascaded TCP/NC tunnel. More specifically, in congestion scenarios with a wide range of link loss rates, the

end-to-end standard TCP with multi-cascaded TCP/NC tunnel can achieve a significantly higher goodput compared to both the end-to-end TCP/NC without tunnel and the end-to-end standard TCP with single TCP/NC tunnel. Those results suggest the potential of the TCP/NC tunnel in goodput performance without change at the end-hosts as well as no per-session management at the gateways.

As future work, more sophisticated sizing method of TCP sending buffer at each TCP/NC gateway and more sophisticated "concatenation" in multi-cascaded tunnel should be investigated to decrease the number of packets dropped at the TCP sending buffer and the number of the dead packets that are stored multiple times in the TCP sending buffer. In this paper, we assume enough resources of each gateway. For practical use, the implementation and deployment costs of our proposed system should be estimated carefully.

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

References

1. Lefevre, F., & Vibier, G.: Understanding TCP's behavior over wireless links. Proceedings of IEEE Symposium on Computers and Communications, 123–130 (2000)
2. Ivanovich, M., Bickerdike, P., & Li, J.: On TCP performance enhancing proxies in a wireless environment. IEEE Communications Magazine, 46(9), 76–83 (2008)
3. Murray, D., Koziniec, T., Dixon, & M.: D-Proxy: Reliability in wireless networks. Proceedings of 16th Asia-Pacific Conference on Communications (APCC), 129–134 (2010)
4. Honda, O., Ohsaki, H., Imase, M., & Murayama, J.: Understanding TCP over TCP: Effects of TCP Tunneling on End-to-End Throughput and Latency. Proceedings of SPIE, 6011, 9 pages (2005)
5. Sundararajan, J.K., Shah, D., Medard, M., Mitzenmacher, M., & Barros, J.: Network coding meets TCP. Proceedings of the IEEE International conference on Computer Comunication (INFOCOM), 280–288 (2009)
6. Ha, N.V., Kumazoe, K., Tsukamoto, K., & Tsuru, M.: Masking Lossy Networks by TCP Tunnel with Network Coding. Proceedings of 22nd IEEE Symposium on Computers and Communications (ISCC), 6 pages (2017)
7. Ha, N.V., Kumazoe, K., & Tsuru, M.: TCP with Network Coding meets loss burstiness estimation for lossy networks. Proceedings of 11th International Conference on Broad-Band Wireless Computing, Communication and Applications (BWCCA), 303–314 (2016)
8. Ha, N.V., Kumazoe, K., & Tsuru, M.: TCP Network Coding with Enhanced Retransmission for heavy and bursty loss. IEICE Transaction of Communications, E100-B(2), 293–303 (2017)
9. Network simulator (ns-3). *https://www.nsnam.org/*. Accessed in March 1, 2016.