On the characteristics of TCP/NC tunneling in heterogeneous environments

Nguyen Viet Ha, Masato Tsuru

Abstract Transmission Control Protocol (TCP) with a loss-based congestion control is still dominantly used for reliable end-to-end data transfer over diverse types of network although it is ineffective when traversing lossy networks. We previously proposed an IP tunneling system across lossy networks using the TCP with Network Coding (TCP/NC tunnel) and showed its potential to significantly mitigate the goodput degradation of end-to-end TCP sessions without any change of end-device's communications protocol stack, but it was shown only in homogeneous conditions. On the other hand, reliable end-to-end data transfer in diverse and heterogeneous IoT environments in a cost-efficient manner is an emerging challenge. Therefore, in this paper, we investigate the characteristics of the TCP/NC tunnel on heterogeneous networks with/without network congestions, to assess the applicability of the TCP/NC tunnel-based intelligent gateway system to IoT environments where end-devices are connected to a gateway with different link bandwidths or connected to different gateways in terms of network topology. The simulation results suggest the TCP/NC tunnel can efficiently utilize the bottleneck bandwidth in such heterogeneous situations even with congestion and achieve a significantly high goodput of end-to-end TCP sessions in a wide range of link loss degree especially when the tunnel link bandwidth is sufficient.

1 Introduction

Transmission Control Protocol (TCP), a transport layer communication protocol between end-nodes with a long history, is still dominantly used for reliable end-to-end (E2E) data transfer with network congestion control. However, TCP suffers a

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

Masato Tsuru
Kyushu Institute of Technology, Japan. e-mail: tsuru@cse.kyutech.ac.jp

low data transfer goodput on lossy networks especially with a long Round-Trip Time (RTT) because its loss-based congestion control reduces the sending rate mistakenly when non-congestion origin packet losses occur due to lossy links [1]. Actually, widely used TCP variants such as NewReno, CUBIC, and Compound-TCP mainly adopt a loss-based congestion control.

To mitigate the goodput degradation of TCP on lossy networks, a variety of solutions have been studied. In E2E solutions, one approach is to improve the congestion detection and the sending rate control in TCP adaptive to wireless networks such as TCP Westwood+ [2], and another approach is to mask packet losses possibly caused by lossy links based on proactively sending redundant packets such as TCP with Network Coding (TCP/NC) [3]. Furthermore, new types of E2E transport protocol on the top of UDP, e.g., QUIC, are also being developed and already in use [4]. Among them, the TCP/NC, on which the authors focused, enables a proactive recovery of lost packets with the redundant transmission of coded packets in cooperation with a reactive recovery of lost packets with retransmission based on the standard TCP-ACK mechanism, so as to maintain the goodput properly even in lossy networks. Among succeeding variants of TCP/NC, our developed one significantly improves the performance by an efficient retransmission and an adaptively optimized redundancy of coded packets based on online packet loss observation [5]. We showed the fundamental benefits of the improved TCP/NC over a wide range of loss rates and loss burstiness degrees with dynamic but slow changes. However, in general, any E2E solution requires a change on all involved end-nodes. In addition, TCP/NC is inefficient when a session traverses multiple heterogeneous networks (i.e., with different link loss rates). In middle-box solutions, the Performance Enhanced Proxy (PEP) approach, such as TRL-PEP [6] and D-Proxy [7], has been widely developed. While they do not require any change of end-nodes, complicated and costly per-TCP session management should be performed on the proxy (gateway) nodes.

On the other hand, the demand for Internet of Things (IoT) applications is growing rapidly with penetration of IoT devices in wild network environments, which are often characterized by a high loss rate and a long RTT. Especially for tiny end-devices, e.g., with less memory and power, it is an emerging challenge to realize E2E data transfer in diverse and heterogeneous IoT environments with no or few changes on end-device in cooperation with a scenario-specific application [8]. Please note that the use of packet-level coding for loss recovery in a variety of data transfer scenarios including IoT has attracted attention [9].

By considering the above issues, we proposed the TCP/NC tunnel to convey end-to-end TCP (E2E-TCP) sessions over lossy networks on a single TCP/NC session between two gateways [10] or on cascaded TCP/NC sessions involving more than two gateways [11] as shown in Fig. 1, without any change in TCP on each end-device. The TCP/NC tunnel is a kind of middle-box solution. However, in contrast to the PEP approach, the “tunneling” approach does not require a complicated per-session management on each gateway. On the other hand, the encapsulation overhead (e.g., header space and processing time) is introduced in general.

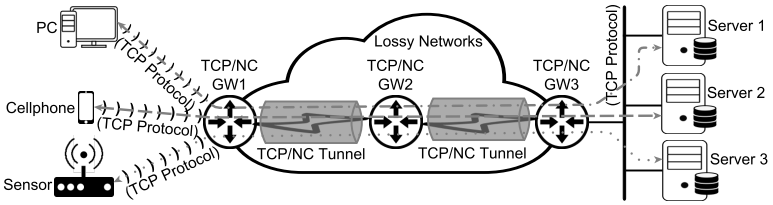


Fig. 1 Example of TCP/NC tunnel with three TCP/NC Gateways

In our previous papers [10, 11], we developed the TCP/NC tunnel system on Network Simulator 3 (ns-3) [12] and showed its potential to significantly mitigate the goodput degradation of E2E-TCP sessions in a wide range of link loss rates, but only in homogeneous conditions. Therefore, the purpose of this paper is to investigate the characteristics of TCP/NC tunnel on heterogeneous networks in terms of the link bandwidths and link positions of end-devices with/without network congestions. Such heterogeneous conditions are essential to IoT networking. Please note that we assume a stationary lossy network such as a large-scale Power Line Communications network or a multi-hop fixed long-distance wireless network. The TCP/NC tunnel for mobile ad-hoc networks remains as future work.

The remainder of this paper is organized as follows. In Section 2, the TCP/NC and its tunnel are briefly explained. Simulation evaluation is presented in Section 3 and a conclusion is given in Section 4.

2 Overview of TCP/NC Tunnel

2.1 TCP/NC

TCP/NC protocol introduces an idea of adding NC layer between TCP and IP layer. In every receiving n TCP segments (called original packets) from TCP layer, NC layer combines them to produce m combination packets (called combination) with $m \geq n$. The sink is expected to recover all the original packets without retransmission using the received combinations even some combinations are lost over a lossy network. TCP layer does not detect any loss events; thus, it maintains the CWND appropriately to stable the goodput performance. The processes of creating m combinations and regenerating n original packets are called encoding and decoding, respectively. Encoding and decoding processes are separated in every n original packets; hence, n also corresponds to Coding Window (CW) size. Redundancy factor $R=\frac{m}{n}$ and the recovery capacity $k=m-n$ in one CW are two parameters to express the recovery ability of the system and must be chosen carefully. Too large R incurs

an unnecessary redundancy causing a small goodput and too large k affects to the decoding delay and the hardware limitations.

NC layer uses the degree of freedom concept and the seen/unseen definition [3] for returning ACK process to avoid TCP layer seeing the losses which can be recovered. The sink sets an ACK number as a sequence number of the oldest "unseen" packet that it needs more information for the decoding process. Fig. 2 is an example of the encoding, decoding and acknowledgment processes. The packets p_1 to p_4 are encoded to the combinations $C[1]$ to $C[6]$. Due to the two lost combinations, the NC layer cannot decode any combination until receiving $C[6]$. For each received combinations, NC layer returns an ACK packet whose ACK number corresponds to the smallest sequence number of the unseen packet. During the process, the TCP layer is unaware of any loss events; thus, the CWND keeps increasing to stable the performance. In this example, R equals to $\frac{6}{4}$, k equals to 2. The packets from p_1 to p_4 are in the same CW. If a new packet comes e.g., p_5, it will be in the next CW.

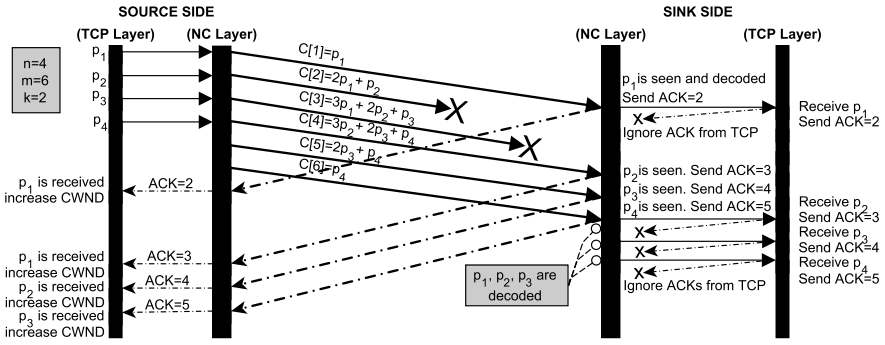


Fig. 2 Network coding process

In our previous work, we did a significant enhancement on TCP/NC in terms of a retransmission mechanism (retransmitting more than one lost packet quickly and efficiently, allowing encoding the retransmitted packets for reducing the repeated losses, and handling the dependent combination packets for avoiding the decoding failure), and an adaptation mechanism for encoding parameters (k and R) to bursty and time-varying losses (estimating both packet loss rate and burstiness by observing transmitted packets, computing appropriate parameters based on a mathematical model of packet losses, and updating the parameters of the current CW promptly) [5]. In this paper, our enhanced version of TCP/NC is used as "TCP/NC".

2.2 TCP/NC Tunnel

We also proposed an IP tunneling system using TCP/NC in which two or more TCP/NC gateways in the middle of the lossy network, called TCP/NC tunnel

[10, 11]. Without any change of end-device's communication protocol stack, the TCP/NC tunnel has brought the benefits of TCP/NC that maintains TCP goodput properly even in lossy networks. The protocol stack and structure of TCP/NC tunneling are illustrated in Fig. 3.

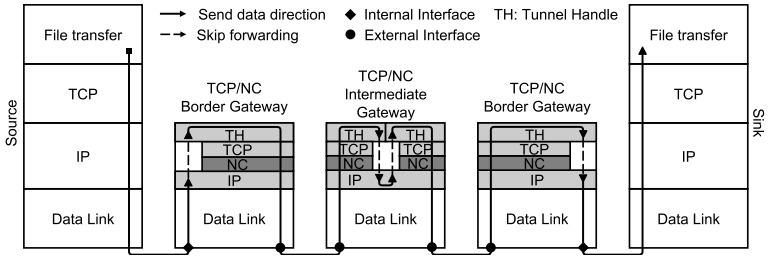


Fig. 3 Tunnel handler

TCP/NC gateway includes two interface types. The first is the "internal" interface which is connected to the local network. The second is the "external" interface which is connected to other networks. If a data packet comes from the internal interface and goes to the external interface, it will join TCP/NC flow. Otherwise (e.g., local delivery), TCP/NC gateway works as a normal router.

Basically, a TCP/NC gateway has two buffers to store a data including link buffer and TCP sending buffer which can be congested. Other buffers (TCP-small-queue, Traffic control queue) are not covered on this paper and are not enabled or available in the simulation. At the sending TCP/NC gateway, an IP packet from the E2E-TCP flow is firstly stored in TCP sending buffer until it is ACKed. Assume that this IP packet is in TCP sending window, a copy of this packet will be sent to the link and stored in link buffer. Consequently, there are two congestion cases of two buffers. If the receiving data rate at TCP sending buffer is larger than the ACKing rate, the congestion will happen at TCP sending buffer. If the sending rate of the higher layer is larger than that of link buffer, the congestion will happen at link buffer.

Packet dropping in network congestion is necessary to inform for TCP layer to decrease the sending rate. But dropping the packet at TCP/NC flow causes goodput degradation because of the lack of function to distinguish the congestion loss, resulting in the NC-parameters estimation and adaptation work inefficiently. The link buffer size should be chosen to limit the packet loss by network congestion. The maximum packets storing in link buffer are $R_{max} \times CWND_{max}$ where R_{max} is the maximum estimated value of redundancy factor at a maximum link loss rate and $CWND_{max}$ is the maximum value of CWND. In the simulation, we consider the random loss channel with the maximum link loss rate of 0.2 and TCP window scaling option is disabled due to the small bandwidth connection; hence, R_{max} is about 1.5 and $CWND_{max}$ is 63 packets (a TCP segment size at TCP/NC gateway is 1000 bytes plus IP and TCP header). Therefore, the link buffer size in this paper is set to 100 packets. The network congestion packet dropping behavior now moves to TCP sending buffer which is set to 64 packets plus the link buffer size (100 packets).

3 Simulation Evaluation

3.1 Simulation Settings

The performance of TCP/NC tunnel in heterogeneous environments has been evaluated on ns-3 with two topologies shown in Fig. 4 and Fig. 5. Both topologies have three routers or TCP/NC gateways (called GW) dependent on the simulation case and accommodate three E2E TCP sessions from source j to sink j (j=1,2,3). The delay of the links among GWs and the delay between the sinks and GW3 were set to 10ms and 1ms, respectively. The links connected to the sources were set to 1.5ms to avoid the bias based on an artificial synchronization of packet arrival times.

Topology 1 shows the mix of different edge link bandwidths connecting three sources to GW1. Sources 1 and 2 are connected with 1$Mbps$ while source 3 with 3$Mbps$. GW1 and GW2 are connected with 3, 5, and 7$Mbps$ in different cases. GW2 and GW3 are connected with 5$Mbps$. Topology 2 is asymmetric where sources 1 and 2 are connected to GW1 while source 3 is connected to GW2. All sources and sinks are connected to GWs with 1$Mbps$, respectively. GW1 and GW2 are connected with 2$Mbps$. GW2 and GW3 are connected with 2, 3, or 4$Mbps$ in different cases.

Fig. 4 Simulation topology 1

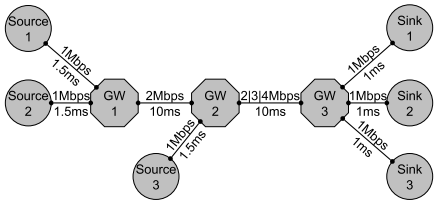


Fig. 5 Simulation topology 2

On both topologies, we evaluate the goodput of each of three E2E-TCP sessions with TCP/NC tunnel (called TCP/NC Tunnel option) and without TCP/NC tunnel (called E2E-TCP option). The TCP type is NewReno which the default settings are used except the Window Scaling, Time-stamp are disabled. One of default settings is Delayed-ACK which its delay is set to 200ms and the number of packets to wait before sending an ACK is set to two packets. The TCP segment size is 1000 bytes. The link buffer and TCP sending buffer size of each TCP/NC gateway is set to 100 and 163 packets. The random lossy channel in the transferred data direction is enabled at the intermediate links (between GWs) with link loss rate per link ranging from 0 to 0.2% (r_0); the total link loss rate from GW1 to GW3 equals $r_0+r_0\times(1-r_0)$. Note that, the link loss rate parameter in the x-axis of the result figures is the link loss rate per link. All the simulations are run 20 times to get the average results (goodput).

3.2 Performance with "mix edge device links" (Topology 1)

The bandwidth of the link between GW1 and GW2 is set to 3*Mbps* (Case 1), 5*Mbps* (Case 2), and 7*Mbps* (Case 3). On this topology, session 3 has a higher bandwidth edge link compared with sessions 1 and 2. With lossy links, unacceptable goodput of all sessions is seen in E2E-TCP option as predicted.

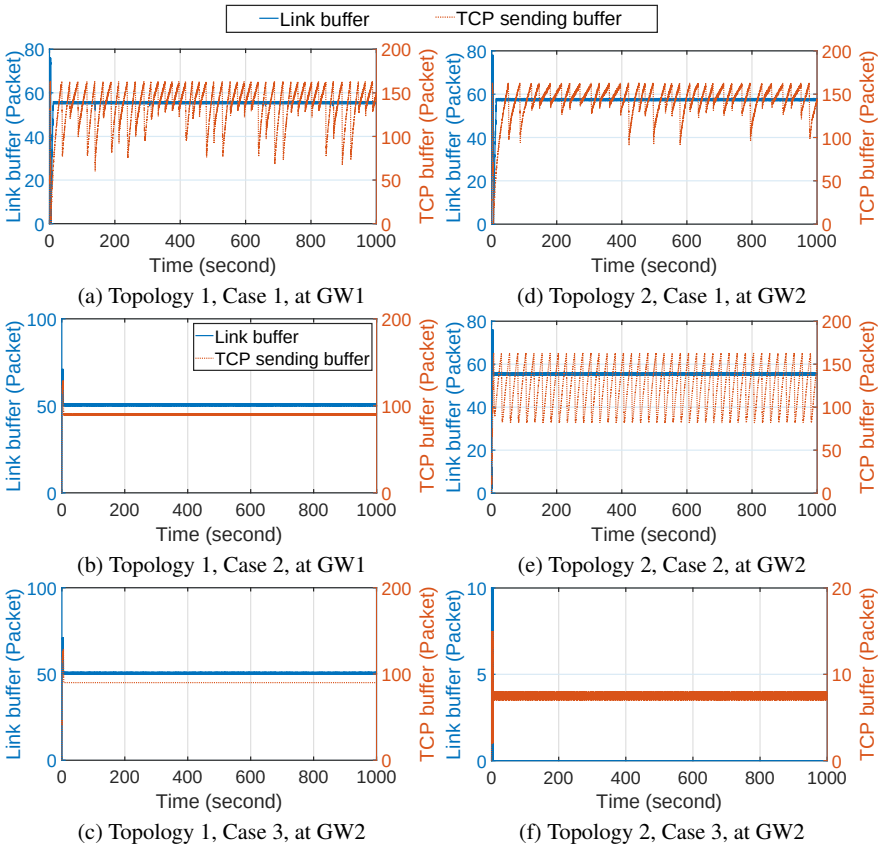
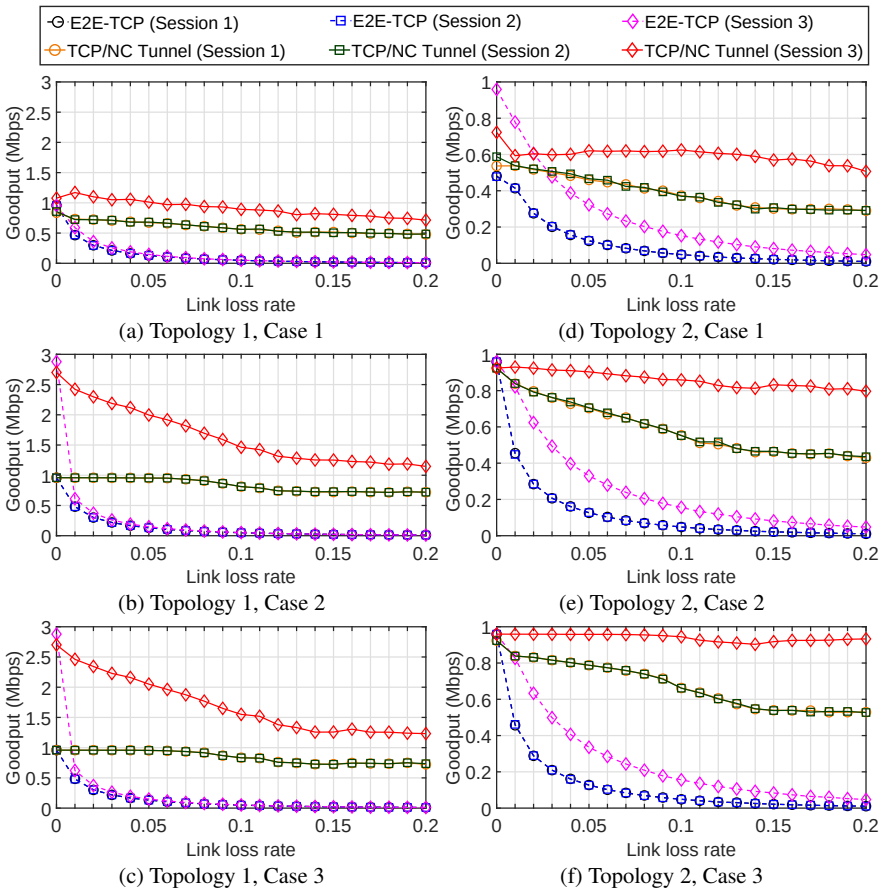


Fig. 6 Queue sizes of buffers in TCP/NC Tunnel option

In Case 1, the congestion happens on GW1 in both options. While the packet loss happens at link buffer in E2E-TCP option, the packet loss happens at TCP sending buffer in TCP/NC Tunnel option shown in Fig. 6(a). Without lossy links (r_0=0), regarding session 3's goodput, E2E-TCP option outperforms TCP/NC Tunnel while TCP/NC Tunnel outperforms E2E-TCP in sessions 1 and 2 shown in Fig. 7(a). Actually the total goodput of three sessions is 2.87*Mbps* in E2E-TCP option and 2.77*Mbps* in TCP/NC Tunnel option shown in Fig. 8(a), indicating a comparably effective utilization of the bottleneck bandwidth of 3*Mbps* with a small difference

resulting from the TCP/NC tunneling overhead (about 4%). As the link loss rate increases with lossy links, TCP/NC Tunnel option can keep the goodput of each session high and stable in a relatively fair manner.

In Case 2, the network congestion does not happen in E2E-TCP option, but it lightly happens in TCP/NC Tunnel option on GW1 due to the TCP/NC tunneling overhead shown in Fig. 6(b). The congestion in TCP/NC option still happens on GW2 in Case 3 because the bandwidth of the link between GW2 and GW3 remains of $5Mbps$ shown in Fig. 6(c). However, even without lossy links, the goodput in TCP/NC Tunnel option is comparable to E2E-TCP option; the difference is negligible as shown in Fig. 7(b,c) and Fig. 8(a). In both cases, as the link loss rate increases, in TCP/NC Tunnel option, the goodput of session 3 decreases gradually while those of sessions 1 and 2 keep $1Mbps$ because the advantage of $3Mbps$ edge link of session 3 becomes less.

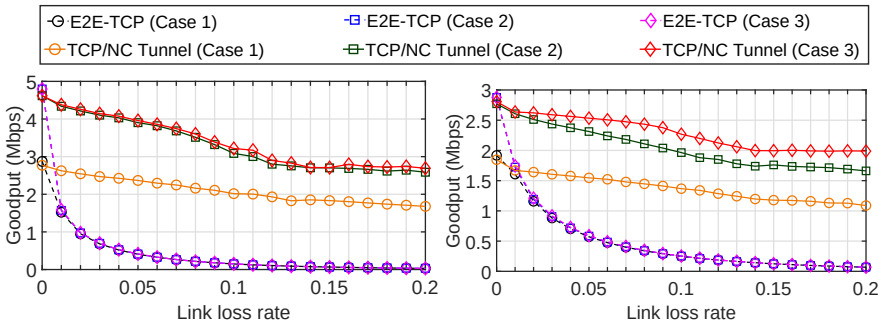


(a) Topology 1, Case 1 (d) Topology 2, Case 1

(b) Topology 1, Case 2 (e) Topology 2, Case 2

(c) Topology 1, Case 3 (f) Topology 2, Case 3

Fig. 7 Per-session Goodputs in three Cases on two topologies



(a) Topology 1 (b) Topology 2

Fig. 8 Total Goodputs in three Cases on two topologies

3.3 Performance on asymmetric topology (Topology 2)

The bandwidth of the link between GW2 and GW3 is set to $2Mbps$ (Case 1), $3Mbps$ (Case 2), and $4Mbps$ (Case 3). On this topology, the network congestion happens at GW2 in both of two options in Case 1 (Fig. 6(d)), while it happens only in TCP/NC Tunnel option in Case 2 due to the TCP/NC tunneling overhead (Fig. 6(e)). But there is no network congestion in Case 3 (Fig. 6(f)). Besides, session 3 traverses a fewer number of lossy links compared with sessions 1 and 2. Therefore, the goodput of session 3 is always greater than that of sessions 1 and 2 in any cases.

In Case 1, Fig. 7(d) shows the goodput of session 3 is greater than that of sessions 1 and 2 in both options at network congestion because of the shorter RTT. In Case 2 (less congestion with no packet dropped) and Case 3 (no congestion), as the link loss rate increases, an E2E-TCP option does not perform suddenly again. In contrast, in TCP/NC Tunnel option, the goodput of sessions 1 and 2 decreases gradually while session 3 keeps a high goodput because the advantage of passing only one lossy links becomes greater.

In any cases, TCP/NC Tunnel option achieves a fairness among sessions like E2E-TCP option. But it has a higher bandwidth utilization compared to E2E-TCP option as shown in Fig. 8(b).

4 Conclusions

We have investigated the characteristics of the TCP/NC tunnel on heterogeneous networks eventually aiming at the TCP/NC tunnel-based “intelligent gateway” system distributed over IoT environments. We have shown that the TCP/NC tunnel can efficiently utilize the bottleneck bandwidth even with congestion, and achieve a significantly high goodput of E2E-TCP sessions in a wide range of link loss degree especially when the tunnel link bandwidth is sufficient.

In future work, the issues to address include (i) network congestion detection by utilizing additional cross-layer information; (ii) modest redundancy control on links with a narrow bandwidth; and (iii) prediction and fast adaptation to rapid changes of network conditions. To solve those issues, more precise environment-dependent models on packet loss and its burstiness are required. An adaptive gateway selection and an integration of possible multiple network paths among gateways are also essential.

Acknowledgements The research is supported by JSPS Grant-in-Aid for Scientific Research 16K00130 and "Resilient Edge Cloud Designed Network", the Commissioned Research of NICT, Japan.

References

1. Leung, K-C. & Li, V.O.K.: Transmission control protocol (TCP) in wireless networks: issues, approaches, and challenges. IEEE Communications Surveys & Tutorials, 8(4),64–79 (2006).
2. Grieco, L.A., & Mascolo, S.: Performance evaluation and comparison of Westwood+, New Reno and Vegas TCP congestion control. ACM Computer Communication Review, 34(2), 25–38 (2004).
3. 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).
4. Langley, A., Riddoch, A., Wilk, A. et al.: The QUIC Transport Protocol: Design and Internet-Scale Deployment. Proceedings of the ACM SIGCOMM, 183–196 (2017).
5. 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).
6. Ivanovich, M., Bickerdike, P., & Li, J.: On TCP performance enhancing proxies in a wireless environment. IEEE Communications Magazine, 46(9), 76–83 (2008).
7. Murray, D., Koziniec, T., Dixon, & M.: D-Proxy: Reliability in wireless networks. Proceedings of 16th Asia-Pacific Conference on Communications (APCC), 129–134 (2010).
8. Gomez, C., Arcia-Moret, A., & Crowcroft, J.: TCP in the Internet of Things: From Ostracism to Prominence. IEEE Internet Computing, 22(1), 29–41 (2018).
9. Sandell, M., & Raza, U: Application Layer Coding for IoT: Benefits, Limitations, and Implementation Aspects. IEEE Systems Journal, 8 pages (Jan 2018; early access).
10. 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), 1292–1297 (2017).
11. Ha, N.V., Kumazoe, K., Tsukamoto, K., & Tsuru, M.: Benefits of Multiply-cascaded TCP Tunnel with Network Coding over Lossy Networks. Proceedings of 15th International Conference on Wired/Wireless Internet Communications (WWIC), 247–258 (2017).
12. Network simulator (ns-3). *https://www.nsnam.org/*. Accessed in March 1, 2018.