Improvement of Acknowledgment Mechanism for TCP with Network Coding

Yosuke Mitsuzumi, Shuhei Aketa, Eiji Takimoto, Shoichi Saito, Koichi Mouri

***Abstract*—The performance of traditional transmission control protocol (TCP) suffers from lossy wireless networks because of its inability to distinguish wireless link errors from congestion-induced losses. TCP with network coding (TCP/NC) is a promising solution to for this problem. TCP/NC masks packet losses from TCP and sends redundant coded packets to correct erasures instead of TCP. However, acknowledgment (ACK) packets for the redundant coded packets may lead to false TCP fast retransmits and degrade performance. To counter this issue, we propose a modification of the acknowledgment mechanism that can better decide whether to send an ACK packet. Simulation results showed that our method reduces fast retransmits by up to about 90% and helps to achieve near-capacity goodput.**

***Index Terms*—TCP, FEC, coding, wireless network.**

I. Introduction

WIRELESS communication is essential for mobile devices such as laptops, sensors, or smartphones. Transmission control protocol (TCP) is the predominant transport layer protocol on both wired and wireless communication networks. It is well known that conventional TCP exhibits two problems in wireless networks. One is erroneous congestion control: TCP cannot differentiate random loss by interference or fading on wireless links from packet drop caused by network congestion. Therefore, TCP erroneously views a random loss as congestion and decreases its sending rate. The other issue is an increase in communication delay caused by retransmission. TCP adopts automatic repeat request (ARQ) for error correction. In ARQ, when a sink node detects a lack of packets, it sends acknowledgment (ACK) packets to the source node with the sequence number of the packet. The ACKs cause the source node to recognize the packet loss; consequently, it retransmits the packet. This procedure requires more than one round-trip time (RTT) to recover a loss. Therefore, in a lossy wireless environment, TCP has many delays due to retransmissions.

TCP with network coding (TCP/NC), introduced by Sundararajan et al. [1], is a promising approach to address these problems. TCP/NC defines a network coding (NC) layer between TCP and IP, as illustrated in Fig. 1. The NC layer encodes the original packets coming from the upper TCP layer and decodes the coded packets it receives from the lower IP layer. The NC layer adopts forward error correction (FEC) by sending an extra coded packet (hereafter called a redundant packet) in advance. In addition, it acknowledges coded packets instead of TCP. The NC layer masks random losses from the upper TCP layer so that erroneous congestion control is suppressed. Moreover, FEC can recover from packet loss earlier than ARQ.

Manuscript received January 8, 2017; revised January 30, 2017.

Y. Mitsuzumi, S. Aketa, E. Takimoto, and K. Mouri are with Ritsumeikan University (e-mail:ymitsuzumi@asl.cs.ritsumei.ac.jp).

S. Saito is with Nagoya Institute of Technology.

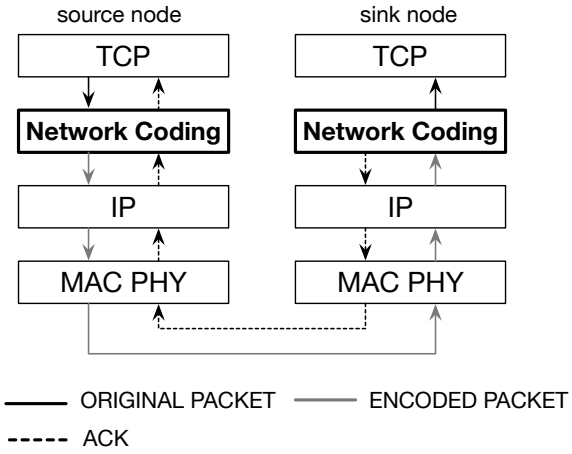


Fig. 1. TCP/NC protocol stack.

However, the acknowledgment mechanism of TCP/NC may degrade performance because of the ACKs for redundant packets. For example, suppose there is an error-free channel between a source and a sink node. The source NC creates three coded packets from three original packets and sends them with three additional redundant packets. Thus, the sink NC replies with six ACKs, including three duplicate ACKs, for all received packets. Although neither random loss nor packet drop have occurred, these duplicate ACKs cause the source TCP to invoke fast retransmit. Hence, this unnecessary retransmission and congestion control decreases the throughput.

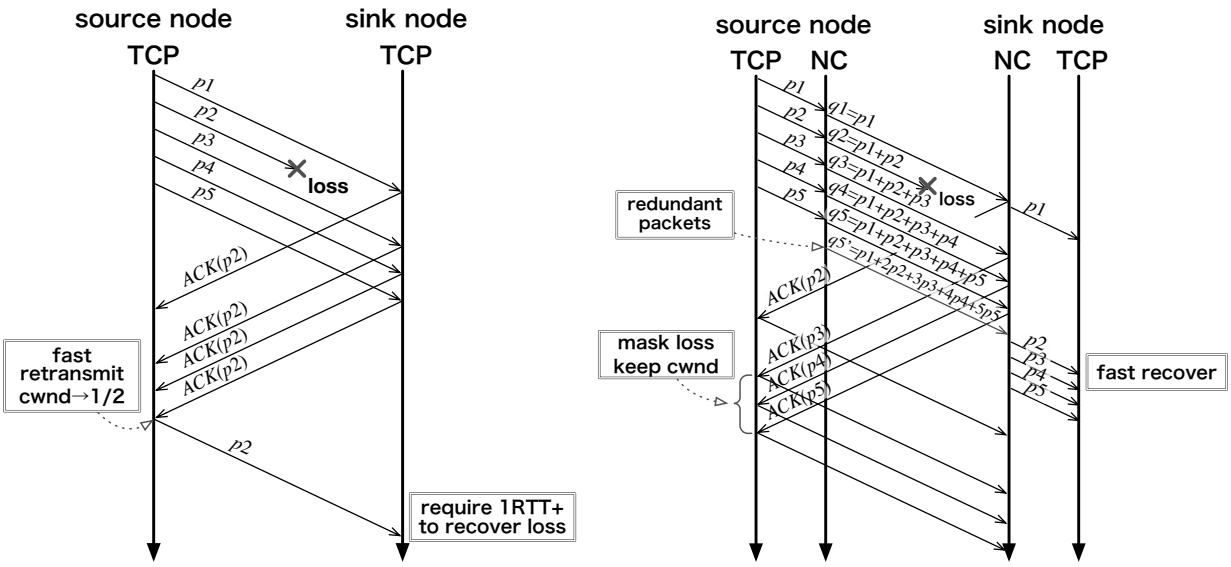
To solve this problem, we introduce an acknowledgment mechanism that cooperates with the source. Through this mechanism, the source NC specifies whether a coded packet is a redundant packet using its header field and the sink NC does not reply with an ACK to packets that are created as redundant packets and not useful for decoding (hereafter called useless redundant packets). This mechanism prevents duplicate ACKs being sent for useless redundant packets so that erroneous fast retransmits do not occur. As a result, the NC layer can transmit more redundant packets than minimally required without fast retransmits. These redundant packets help speed recover from losses and achieve near-capacity goodput.

The rest of this paper is organized as follows. An overview of TCP/NC is described in Section Ⅱ. The proposed acknowledgment mechanism with TCP/NC is presented in Section Ⅲ. The simulation results are described in Section Ⅳ, and the conclusions and future work are discussed in Section V.

II. TCP/NC

TCP/NC introduces an NC layer between the TCP and IP, as illustrated in Fig. 1. The source NC generates a

ISBN: 978-988-14047-7-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)



(a) TCP decreases cwnd and needs 1RTT

(b) TCP/NC keeps cwnd large and recovers quickly

Fig. 2. Comparison between conventional TCP and TCP/NC in a random loss senario.

coded packet that involves multiple TCP segments for every arriving segment and transmits it to the sink. The sink NC performs the decoding operation for each received coded packet, and then sends back a TCP ACK to the source TCP. If the sink NC collects enough coded packets to decode the segments, it sends them to the sink TCP and discards the ACKs from the sink TCP to the source.

One of the novelties of TCP/NC is the incorporation of an end-to-end coding scheme into the TCP without spoiling the TCP's functionality. TCP manages the sending rate using the principle of self-clocking for flow and congestion control. Encoding and acknowledging coded packets in an online manner, called online coding [2], enables the NC layer to be synchronized with TCP ACK-clocking. Therefore, TCP can transparently use the lower layer without any changes to the TCP.

The main advantages of TCP/NC are that it masks random losses from TCP and quickly recovers from losses by FEC. Fig. 2(a) and (b) depict conventional TCP and TCP/NC, respectively, are in a scenario in which TCP sends five segments p_1 to p_5, of which one is a random loss. Conventional TCP replies with duplicate ACKs for p_3, p_4, and p_5, because of the loss of p_2. They trigger a fast retransmit with erroneous congestion control, and it results in performance degradation. In contrast, TCP/NC sends back ACKs in order of sequence number for q_3, q_4, and q_5 despite of lack of q_2. Therefore, the random loss is hidden from TCP and the TCP maintains its sending rate.In TCP/NC, a lack of coded packet stalls the decoding of segments and this affects the upper layers as a decoding delay. The source NC periodically sends a redundant packet as q'_5. It increases the opportunities for the sink NC to gather enough coded packets for decoding. Therefore, TCP takes one RTT or more for loss recovery, whereas TCP/NC accomplishes it within one RTT.

We present the details of the coding operation and the acknowledgment with NC in the rest of this section.

A. Coding Operation

1) Encoding: TCP/NC adopts random linear coding [3] for the coding scheme. Random linear coding produces a linear combination of original packets as a coded packet using randomly chosen coefficients. In TCP/NC, TCP segments p_i, where $i \in \{1, 2, \ldots, n\}$, are combined into q_j, where $j \in \{1, 2, \ldots, m\}$. This encoding operation is presented in Eq. 1, where coefficients are given as $\alpha ji \in \mathbb{F}_{2^q}$. The coded packet consists of header $\vec{\alpha_j} = (\alpha_{j1}, \alpha_{j2} \ldots \alpha_{jn})$ and payload q_j. The whole of encoding operation is given as Eq. 2, where $\vec{p} = (p_1, p_2, \ldots, p_n)$, $\vec{q} = (q_1, q_2, \ldots, q_m)$, and $C = (\alpha_{ji}) \in \mathbb{F}^{m \times n}$.

$$q_j = \sum_{i=1}^{n} \alpha_{ji} p_i \tag{1}$$

$$\vec{q} = C\vec{p} \tag{2}$$

Each coefficient is randomly chosen from finite field $\mathbb{F}_{2^q}$. According to [3], If field size $\mathbb{F}_{2^q}$ is large enough ($q = 8$ in this paper), this system guarantees that each coded packet becomes linearly independent, that is, useful for decoding.

2) Decoding: A sink NC can obtain coefficients and linear combinations from coded packets and construct a linear equation (Eq. 2). The decoding operation is given as Eq. 3. The decoder solves the linear equations using Gauss-Jordan elimination to extract the segments $\vec{p}$. To solve the equations, matrix C must be $m \geq n$. This indicates that, to be decoded, the number of coded packets must be larger than the number of segments involved in them.

$$\vec{p} = C^{-1}\vec{q} \tag{3}$$

3) Forward Error Correction: A source NC includes redundant coded packets in the coded packets for error correction. The number of redundant packets is given by redundancy factor R, which is defined as the proportion of number of coded packets to original packets, namely, $\frac{m}{n}$.

ISBN: 978-988-14047-7-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

With $\rho\%$ loss rate, the theoretical value of R is determined to be $\frac{1}{1-\rho}$.

The maximum count of the original packets involved in a coded packet is presented by coding window size W. The value of W affects two kinds of performance, the overhead of the header and capacity for loss. As mentioned above, the coded packet's header contains the coefficients. The header size becomes large because of the lager number of coefficients as W is increased. Thus, a very large W increases the communication overhead of the header. At the same time, W guarantees that the system can correct losses within the continuous W losses with redundant packets. Therefore, larger W enhances the system's robustness for the losses.

B. Acknowledgment Scheme of the NC Layer

The NC layer has an acknowledgment mechanism for concealing loss and compatibility with TCP. The source NC returns TCP ACK packets for every reception of a coded packet. To acknowledge the coded packet with TCP ACK, TCP/NC introduces a $seen/unseen$ definition. According to the definition provided by [4], an original packet p_k is $seen$ by a node if it has enough information to form a linear equation into $(p_k + q)$, where $q = \sum_{i=k+1}^{n} \alpha_i p_i$. $Seeing$ p_k guarantees that the following coded packets do not need to contain it even if it has not yet been decoded. This can be regarded as a reception of p_k so that the sink NC can acknowledge the seen packet and send a TCP ACK with the sequence number of the oldest unseen packet.

III. Improvement of the Acknowledgment Mechanism in TCP/NC

Previous studies [1] have found that TCP/NC can achieve to close the capacity throughput in a lossy network with an appropriate R based on loss rate, and the value of R should slightly exceed the theoretical value on a practical network from empirical experiments. It can be considered that the extra redundant packets reduce the decoding delay and enhance the system robustness. However, the conventional TCP/NC acknowledgment mechanism may degrade the performance with the extra redundant packet. The rest of this section describes the drawbacks of the TCP/NC acknowledgment mechanism and proposes a solution.

A. Performance Degradation by Unnecessary Fast Retransmits

In TCP/NC, the sink NC acknowledges all received coded packets, and has no ability to separate redundant packets from coded packets. Hence, it replies with ACKs to useless redundant packets that are created as redundant packet at the source and are discarded at the sink. They become duplicate ACKs leading fast retransmits even in an error-free environment. Fig. 3 illustrates such an event. Suppose the source NC generates a redundant packet for every coded packet ($R = 2$) on an error-free channel. The source NC sends redundant packets q'_1, q'_2, and q'_3 adding to coded packets q_1, q_2, and q_3. The sink NC discards q'_1, q'_2, and q'_3, regarding them as useless redundant packets, and returns three duplicate ACKs to the source. These duplicate ACKs cause the source TCP to create an unnecessary fast retransmit.

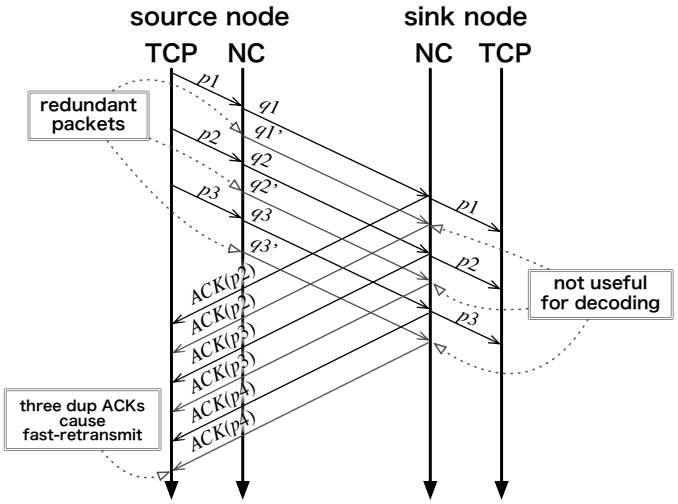


Fig. 3. TCP/NC with R=2 over a no-loss channel.

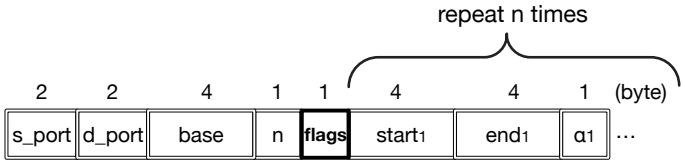


Fig. 4. New acknowledgment mechanism incorporating the *flags* field into the existing TCP/NC header.

On account of the drawback of the acknowledgment mechanism, TCP/NC may suffer on practical networks for two reasons. The first reason is that calculating appropriate R on real networks is a difficult challenge. Since packet loss rates always fluctuate, sending the minimally required redundant packets is difficult in practice. Therefore, R is estimated to be larger than the theoretical value, and it is inevitable that some redundant packets become useless redundant packets. The second reason is that a static FEC cannot adapt to dynamically occurring packet loss. As mentioned earlier, the source NC periodically sends a redundant packet. However, it does not always meet an occurrence of packet loss in practice. Therefore, such differences in timing produce useless redundant packets.

B. Source Cooperative Acknowledgment Mechanism

To address this limitation of the acknowledgment mechanism, we propose an acknowledgment mechanism that cooperates with a source. In this mechanism, the source NC explicitly specifies the redundant packet using the coded packet's header field. The sink NC determines useless redundant packets using the field and cancels the ACK message.

The proposed mechanism newly reserves a *flags* field on the conventional header [1], as depicted in Fig. 4. The source NC sets a bit on the *flags* if it is a redundant packet.

The sink NC decides whether to send an ACK using the *flags* field. Fig. 5 describes the algorithm of the sink NC. The sink NC always acknowledges the coded packet that is used for decoding. The other packets, which are useless for decoding, are divided into two types. One is that the *flags* field indicates the redundant packet, namely, the useless redundant packet. The sink NC does not acknowledge

ISBN: 978-988-14047-7-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

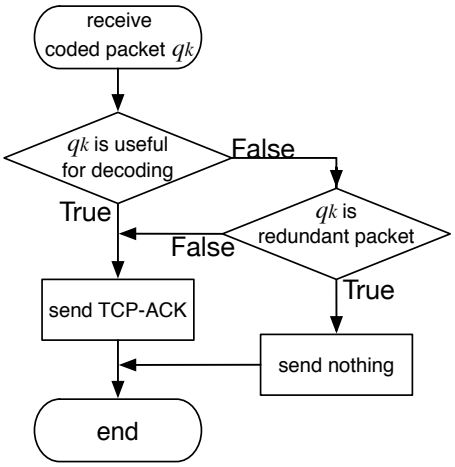


Fig. 5. Sink NC algorithm to determine whether to send TCP ACK.

TABLE I
SIMULATION PARAMETERS

Parameter	Value
802.11 protocol (Source-AP)	IEEE802.11b
Maximum transmission rate (Source-AP)	11Mbps
Maximum transmission rate (AP-Sink)	10Mbps
Application	FTP
Data size (In number of packet)	9.6MB (10,000)
MSS	1,000byte
TCP variants	TCP-NewReno
Coding window size (W)	5
Redundancy factor (R)	$x \sim x + 0.05$
Finite field	$\mathbb{F}_{2^8}$

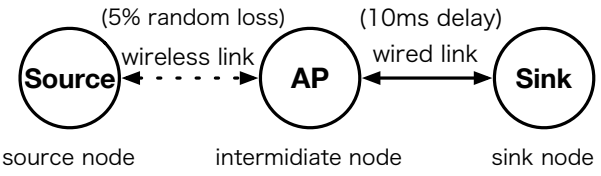


Fig. 6. Simulation topology assuming a wireless LAN system via an access point (AP).

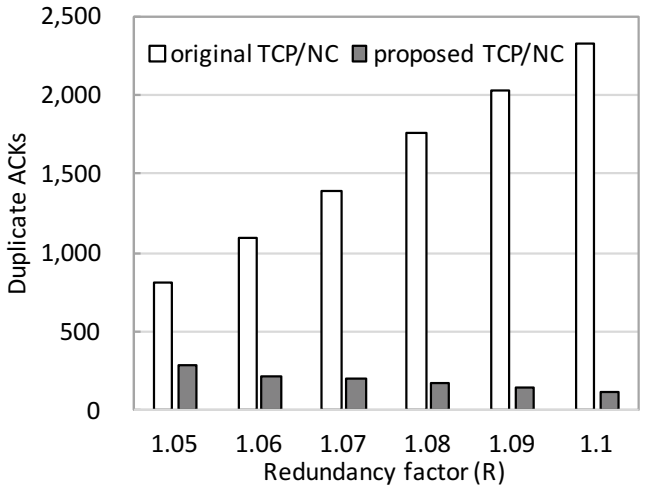


Fig. 7. Number of duplicate ACKs versus R.

it. The other is neither a redundant packet nor a useful packet. It corresponds to a retransmitted coded packet that is retransmitted because of the loss of ACKs. Assume that all sink ACKs are lost. The source NC resumes sending coded packets according to TCP's retransmission time out. However, the retransmitted coded packet has already been acknowledged by the sink, so that it will be a useless coded packet. Such a coded packet must be acknowledged; otherwise it results in a non-communication state.

IV. SIMULATION RESULTS

We implemented the proposed method in addition to the TCP/NC on the QualNet network simulator [5]. The simulation topology was assumed to be a general wireless LAN network that consists of a mobile (sink) node, AP, and server (sink) node, as depicted in Fig. 6. The basic simulation parameters are presented in Table I. The wireless link between the source and AP had a bandwidth of 11Mb/s and was set to cause random losses. The wired link between the AP and sink had a bandwidth of 10Mb/s and a propagation delay of 10 ms. The source transmitted 9.6 MB (10,000 packets) to the sink. TCP/NC's parameter coding window size W was 5 and redundancy factor R ranged from the theoretical value x, which is calculated using the loss rate, to $x + 0.05$. The simulation was repeated ten times to obtain an average value.

A. Effectiveness of the Mechanism

First, we provide comparisons between the original TCP/NC and the TCP/NC with our proposed method to establish the effectiveness of the method. A packet loss rate with the wireless link was set to 5%. We incremented R from its theoretical value $1.05 \approx \frac{1}{(1-0.05)}$ by 0.01 each time.

Fig. 7 illustrates the number of duplicate ACKs for each value of R. In the original TCP/NC, duplicate ACKs linearly increase with R or redundant packets due to useless redundant packets. In contrast, the number of duplicate ACKs in the proposed TCP/NC has no tendency to rise because of proposed method suppresses useless redundant packets. Furthermore, the proposed TCP/NC decreases duplicate ACKs for larger value of R because the extra redundant packets help fast recovery and enhance its capability to conceal losses.

Fig. 8 gives the number of fast retransmits for each value of R. As duplicate ACKs trigger fast retransmits, the original TCP/NC increases fast retransmits for larger value of R, whereas proposed TCP/NC reduces it.

The effect of reducing the fast retransmits is shown in Fig. 9 and Fig. 10. Fig. 9 plots the TCP congestion window (cwnd) for each value of R. Since fast retransmits trigger congestion control to reduce cwnd, the cwnd of the original TCP/NC declines, whereas the proposed TCP/NC keeps cwnd high.

Fig. 10 depicts the goodput for each value of R. As described earlier, unnecessary fast retransmits degrades performance. Therefore, the goodput of the original TCP/NC shrinks with R, and the goodput of the proposed TCP/NC increases.

The simulation results confirms that the proposed method suppresses duplicate ACKs and this reduces the number of fast retransmits and maintains near-capacity goodput. Another important finding is that if the problem is solved with too high a value of R, the proposed TCP/NC enhances

ISBN: 978-988-14047-7-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

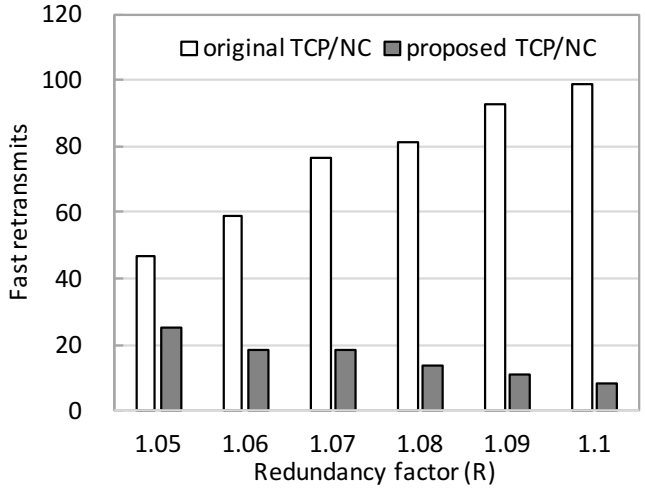


Fig. 8. Number of fast retransmits versus R.

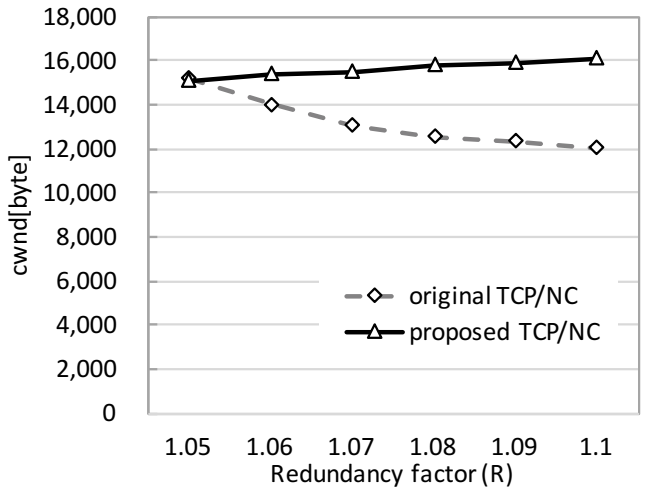


Fig. 9. Congestion window versus redundancy factor R.

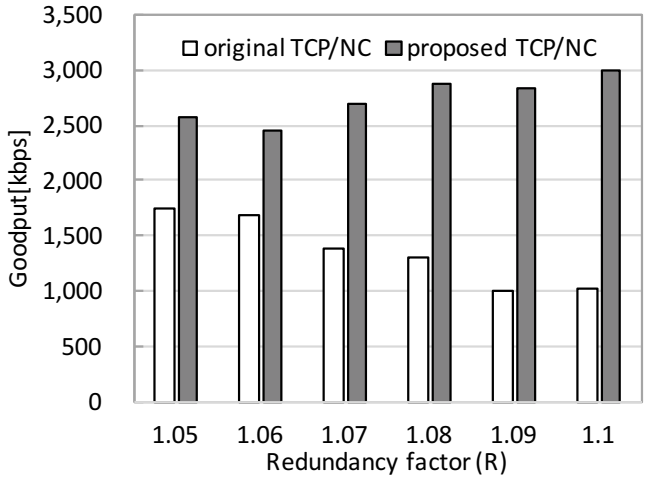


Fig. 10. Goodput versus redundancy factor R.

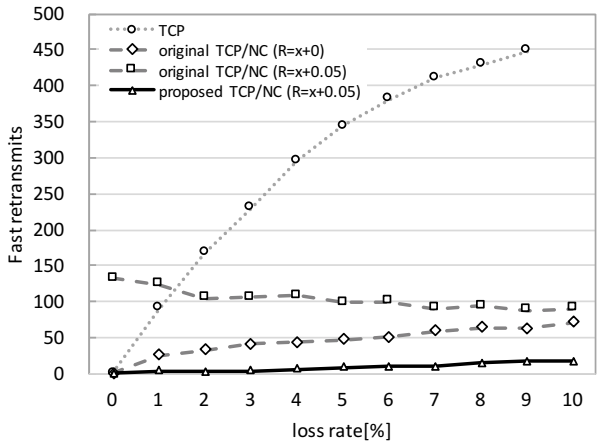


Fig. 11. Number of fast retransmits versus loss rate.

the effectiveness of FEC.

B. General Performance Evaluation

We evaluated performance with a variable loss rate for TCP, original TCP/NC, and the proposed TCP/NC. The value of R of the original TCP/NC was set to high $(x + 0.05)$ and low (x), and that of proposed TCP/NC was set to only high since higher R is better as mentioned above. In the simulations, normal TCP was not able to complete a simulation with a 10% loss rate due to heavy traffic.

Fig. 11 plots the number of fast retransmits for each loss rate. We can see that for almost all cases for TCP/NC, this value is lower than that of TCP because of the loss-masking function. The original TCP/NC with the high R has a greater number of fast retransmits than the other cases of TCP/NC because of the useless redundant packets. In contrast, the number of fast retransmits of TCP/NC with low R gradually increases with the loss rate due to a shortage of redundant packets. The number of fast retransmits for the proposed TCP/NC remains low in all cases.

Fig. 12 depicts the goodput for each loss rate. The results are inversely proportional to the fast retransmits. With a 5% loss rate, the goodput of all cases of TCP/NC exceeds that of TCP and that of the proposed TCP/NC increases by 70% compared with the original TCP/NC. The proposed TCP/NC exhibits a high goodput at all loss rates.

V. Conclusion

TCP/NC was proposed to overcome the performance degradation of conventional TCP in a lossy network. However, TCP/NC does not distinguish the redundant packets on acknowledgment. As a result, duplicate ACKs caused by useless redundant packets lead to unnecessary fast retransmits. To address this problem, we proposed an adequate acknowledgment mechanism using a coded header. The simulation results show that our method reduces duplicate ACKs and fast retransmits, and it achieves near-capacity goodput in a lossy network.

An other approach derived from TCP/NC is dynamic network coding (DNC) [6]. DNC replaced FEC with ARQ as the error correction on the NC layer. The system uses feedback that contains the number of required packets and retransmits the coded packets based on the feedback information. ARQ-based retransmission can prevent useless redundant packets. However, this approach does no have benefits of FEC. Our method can be applied for other proposed TCP/NC variants that adopt FEC.

ISBN: 978-988-14047-7-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

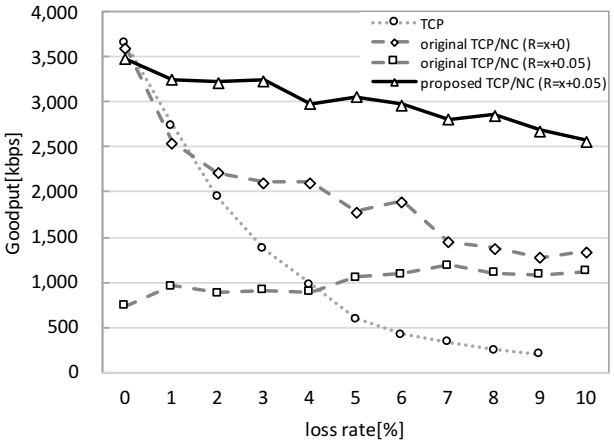


Fig. 12. Goodput versus loss rate.

As future work, we will implement the proposed method in Linux and evaluate its effectiveness and limitations in practice.

REFERENCES

[1] J. K. Sundararajan, D. Shah, M. Medard, S. Jakubczak, M. Mitzenmacher, and J. Barros, "Network coding meets tcp: Theory and implementation," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 490–512, March 2011.

[2] J. K. Sundararajan, D. Shah, and M. Medard, "Feedback-based online network coding," *arXiv.org*, Apr. 2009.

[3] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, Oct 2006.

[4] J. K. Sundararajan, D. Shah, and M. Medard, "Arq for network coding," in *2008 IEEE International Symposium on Information Theory*, July 2008, pp. 1651–1655.

[5] QualNet, http://web.scalable-networks.com/qualnet-network-simulator.

[6] J. Chen, W. Tan, L. Liu, X. Hu, and F. Xu, "Towards zero loss for tcp in wireless networks," in *Performance Computing and Communications Conference (IPCCC), 2009 IEEE 28th International*, Dec 2009, pp. 65–70.

ISBN: 978-988-14047-7-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)