Suppressing Redundant TCP Retransmissions in Wireless Mesh Networks

Shuhei Aketa, Eiji Takimoto, Yuto Otsuki, Shoichi Saito, Eric W. Cooper, Koichi Mouri

***Abstract*—The link quality of wireless mesh networks is variable, hence dynamic route modifications based on changes in link quality can improve communication performance. A drawback of the route modification is the occurrence of order errors, which triggers retransmission control in TCP communication. However, retransmission is unnecessary for order errors in route modification as retransmission is redundant. In this study, we propose a method that discriminates the causes of order errors on the basis of their TCP sequence numbers and reception intervals. We also propose a method that suppresses the redundant retransmissions caused by order errors. When the determination method infers that an order error is due to a route modification, the suppression method retains the acknowledgement (ACK) packet so as not to activate the retransmission control of the TCP sender. Even if the determination method makes a misjudgment, its impact is minimized by limiting the waiting time. The results evaluated in Linux show a 50% reduction of retransmitted packets in Reno and CUBIC and a 90% reduction in selective acknowledgement (SACK) Reno and SACK CUBIC.**

***Index Terms*—Wireless Mesh Networks, TCP, Order Errors, Retransmission Control**

I. INTRODUCTION

WIRELESS mesh networks are built such that each node is connected by direct radio, consequently making such networks susceptible to interference. Distance vector routing protocols (e.g., AODV [1], DSR [2]) construct routes depending on the network topology without considering the effect of interference, and do not change a route if there is no change in the topology. Therefore, these protocols experience problems when communication performance is degraded because of network traffic congestion. In contrast, link state routing (LSR) protocols (e.g., OLSR [3], FSR [4]) construct routes depending on the link quality between each node and take into consideration the effect of interference. OLSRs can follow a change in link quality during communication and perform route modification, but this link quality is characterized by coarse granularity. Using a metric with finer granularity improves communication performance. Therefore, dynamic metrics for wireless multihop networks have been proposed. The expected transmission count (ETX) [5] dynamic metric is based on the number of transmitted data frames at the MAC layer. The expected transmission time (ETT) [6] is a metric in which the communication rate is considered in ETX. These metrics reflect dynamic communication environments. They can be used to select a path with good link quality and are expected to improve communication performance [7]. However, an increased occurrence of order errors has been reported when using dynamic metrics because the route modifications can cause preceding packets to be overtaken [8]–[10].

LSR protocols with dynamic metrics can also cause order errors. TCP corrects out-of-order packets, but in TCP with fast transmit, order errors are sometimes misidentified as packet loss when there is a delayed arrival of packets, as required for sequence error detection. When order errors by route modification occur with a change to a vacant path from a congested path (Fig. 1), the packets overtaken tend to increase in width. When the width of the overtaken packets becomes large, the overtaken packets are retransmitted. When out-of-order packets due to route change are retransmitted, this transmission is redundant since there was no packet loss. However, redundant retransmissions that reduce communication performance are considered to be a necessary evil in order to reduce the TCP congestion window and thus reduce waste in radio resources.

In this study, we propose a method to suppress redundant TCP retransmissions that are caused by route modifications in wireless mesh networks. Two TCP features are responsible for generating redundant retransmissions:

- TCP does not determine the cause of the order error.
- TCP that performs fast retransmit when receiving three consecutive duplicate ACKs does not relate these to the cause of the error sequence.

The proposed method determines the route change or packet loss caused by order errors on the basis of the difference between the reception interval and the TCP sequence number during TCP data packet reception. When the order error is caused by packet loss, the proposed method does nothing. When it is caused by a route change, the proposed method delays the transmission of the TCP ACK packet that may duplicate the ACK. Thus, the proposed method controls redundant retransmissions. We proposed a method for determining the cause of order errors depending on only the difference of the sequence number [11]. However, this method can not handle burst-type packet loss. We improved the determination method for depending on the difference between the reception interval and the sequence number. In this study, we do not take node mobility into account.

The rest of the paper is organized as follows: In section 2, we explain why redundant retransmission occurs. In section 3, we describe the proposed method. In section 4, we evaluate the performance of our protocol. In section 5, we explain related work on suppressing the impact of order errors. In section 6, we present our conclusions.

II. REDUNDANT RETRANSMISSION SEGMENT

Fig. 1 is a typical example of order errors caused by route change. A route is selected for the destination node from the

This study was supported by the Grant-in-Aid for Challenging Exploratory Research of the Japan Society for Promotion of Science (JSPS) under the Contract No. 25730065.

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

S. Saito is with Nagoya Institute of Technology.

ISBN: 978-988-19253-9-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

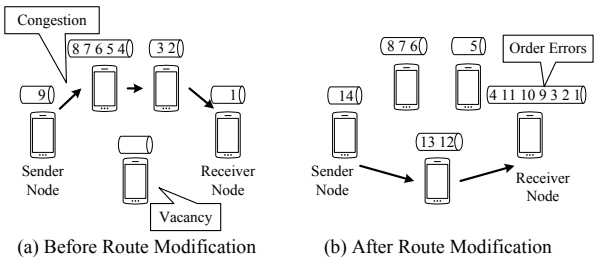

Fig. 1. Typical example of order error.

source node in Fig. 1 (a). The queue of the relay node begins accumulating packets with the start of communication. After a while, the routing changes to the vacant route below, because of the reduced link quality of the route in use. In this case, the arrival time to the destination node is short because there is no packet waiting in the queue of the relay node of the new route. The route is changed when a sender node sends eight packets, as shown in Fig. 1. Packets 9, 10, and 11 arrive at the destination node before packet 4. As in this example, packets may be overtaken because of the difference in the arrival times of packets before and after path switching.

Fig. 2 illustrates the transition of the TCP sequence number where the out-of-order packets occurred because of a route change. We observed these results at the destination node in a preliminary experiment using the network simulator QualNet [12]. The blue line represents the sequence number of the TCP data packet received at the destination node. The green line represents the sequence number of the TCP ACK packet sent to the destination node. As shown in Fig. 2 (1), order errors occur for five consecutive packets after receiving the packet sequence number with a large difference because of a route change at 344.06 s. The ACK packet for the arriving packet which has been overtaken, is a duplicate ACK. The receiver TCP sends duplicate ACKs because of the arrival of the overtaking packets at 344.1 s (Fig. 2 (2)). The source node that received the duplicate ACKs performs a fast retransmit. Therefore, the retransmitted packets arrive at the destination node at 344.16 s (Fig. 2 (4)). Packets with a fast retransmit are received with a delay but without loss (Fig. 2 (3)). In other words, this retransmission control is redundant. In addition, three packets are retransmitted because of the characteristics of TCP NewReno [13], which assumes that the loss of multiple packets is due to a single transmission window size, and then retransmits multiple data packets. This is also a redundant retransmission of the previously received packets.

III. PROPOSED METHOD

TCP performs a fast retransmit when it receives three duplicate ACKs regardless of the cause of the order error. In this study, we propose a method for determining the cause of order errors depending on the difference between the reception interval and the TCP sequence number at the TCP data packet reception. Moreover, we propose a method for suppressing retransmission by delaying the transmission of the duplicate ACK packet when the cause is determined to be a route change.

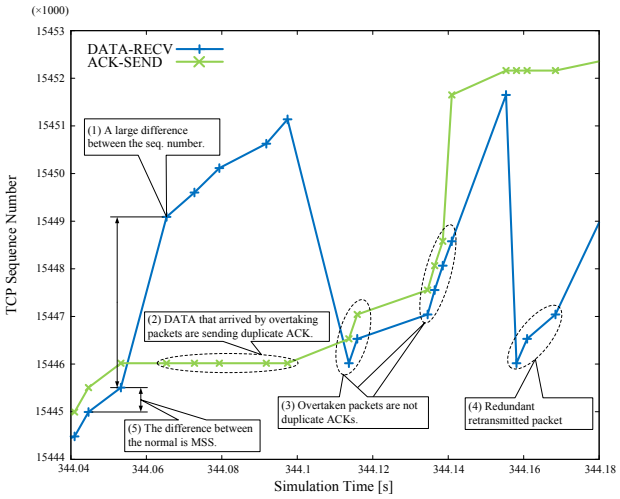

Fig. 2. Transition of TCP sequence number with order errors.

A. Method for Determining Order Errors

The size differences in the TCP sequence number of the order error tend to differ depending on the cause. Upon receiving the data packet in the correct arrival order, the difference between the sequence numbers is equal to the maximum segment size (MSS). The reason is the Nagle algorithm [14]. Out-of-order packets due to rerouting occur by subsequent data packets passing through the new path and overtaking the preceding data packets. That is, when rerouting occurs, it increases the difference in the sequence number. On the other hand, sporadic packet loss has a higher frequency of occurrence than burst-type packet loss. In other words, when sporadic packet loss occurs, the difference in the sequence number decreases. The packet arrival interval tends to vary depending on the cause. When packet loss occurs, the arrival interval becomes longer because the original packet to be received did not arrive, so there will be a time interval associated with the lost packets. In contrast, route changes do not tend to cause significant changes in the arrival interval, and they do not directly affect the arrival interval of the overtaken packets. We confirmed both these patterns using QualNet. The proposed method determines the cause of the out-of-order packet depending on the difference between the reception interval and the TCP sequence number at the time of data packet reception. We calculate the difference in the sequence number of the data packet received immediately before the sequence number of the TCP header. We then calculate the difference in the reception interval from the difference between the arrival time of the previous and current data packets.

The threshold of the sequence numbers is three times the MSS value. Three times this size means three packets because the trigger for fast retransmit is receiving three duplicate ACKs, which corresponds to the three or more overtaken packets. The threshold of the reception interval is the product of the overtaking packet width and the average arrival interval of the most recent packet of the eight. This threshold is the expected arrival time of the next packet. This method can also handle burst-type packet loss. The proposed method confirms these thresholds at the occurrence of the order error.

ISBN: 978-988-19253-9-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

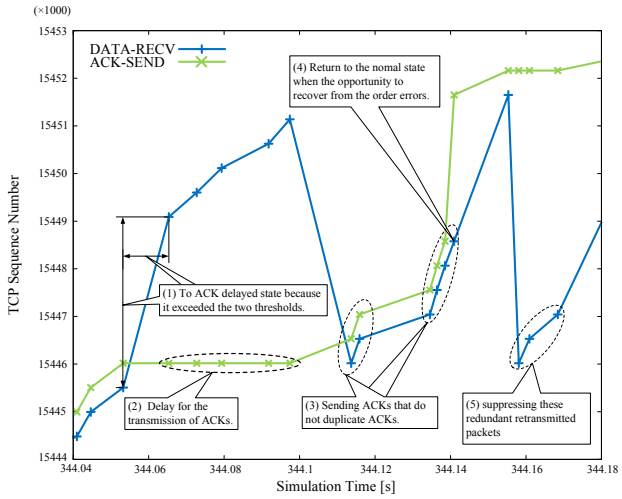

Fig. 3. Behavior of the proposed method.

B. Method for Suppressing Redundant Retransmissions

This suppression method, which delays the transmission of the ACK packet, determines the occurrence of order errors by rerouting, as explained in Section III-A. It is possible to suppress fast retransmission when the source node does not receive duplicate ACKs after a delayed ACK transmission. The proposed method suppresses fast retransmit using two states, i.e., the normal state and the ACK delayed state. In the normal state, the receiver performs the normal TCP operation. The ACK delayed state is a state of recovery, waiting for in-order packets and any out-of-order packets delayed because of route changes, and suppressing retransmission of duplicate ACKs in the ACK delayed state.

The procedure of the proposed method is described below:

Step1 When a detected value exceeds the thresholds of the reception interval and sequence number, the destination node shifts into the ACK delayed state.

Step2 The ACK packet for the data packet that arrived by overtaking delayed packets sends a duplicate ACK.

Step3 The ACK packet for an arriving data packet that was overtaken transmits without waiting for order errors.

Step4 When all the overtaken data packets have been received, the node returns to its normal state because it has recovered from order errors.

Fig. 3 shows an example of the proposed method. This figure is the same as Fig. 2 but includes the steps of the proposed method. Each number below corresponds to those in Fig. 3.

1) When the difference in the sequence number and reception interval exceeds the threshold of 344.06 s, the destination node shifts into the ACK delayed state.
2) The destination node delays the transmission of the ACK packet for the five arriving data packets by going into the ACK delayed state (at 344.1 – 344.06 s).
3) Since the ACK received for the overtaken data packet is not a duplicate ACK, it transmits normally.
4) Recovery from an out-of-order event occurs when the destination node returns to its normal state at 344.14 s, and the ACK packets that have been delayed are discarded.

Following this procedure, the ACK packets in Fig. 3 (2) are delayed to reduce the number of redundant retransmission packets in Fig. 3 (5).

If the TCP selective acknowledgement (SACK) option [15] is used, the source node retransmits the ACKs it receives with SACK options. When the ACKs with SACK options are transmitted in the ACK delayed state, the source node performs a redundant retransmission. Therefore, the proposed method removes the SACK option of these ACKs.

C. Measures of Misjudgment

The proposed method determines the cause of the out-of-order packets depending on the difference between the reception interval and the TCP sequence number at data packet reception. This involves a transition to the ACK delayed state, which may be invoked erroneously by attributing delays to out-of-order packets, and may thus inhibit necessary retransmissions. This may cause the worst case retransmission timeout (RTO). When the proposed method makes a wrong decision to transition to the ACK delayed state and waits for the required ACK packet to be sent, the source node experiences zero window conditions and cannot receive ACK packets, thus falling into a non-transmission state in which it cannot send packets or fast retransmit. This state continues until RTO occurs at the source node. When RTO occurs, the window size decreases to the one assigned by the congestion control, and the throughput is reduced. Therefore, it is necessary to suppress RTOs due to misevaluation. To address this problem, the proposed method sets a timer, called the misevaluation check timer, fora data packet received in the ACK delayed state.

The determination method errs if it does not recover from the order error by the expiration of the misevaluation check time and return to the normal state. When the misevaluation check time expires, the last three duplicate delayed ACK packets are transmitted. The source node uses fast retransmit on receiving the three duplicate ACKs. As a result, the receiver recovers from order errors by fast retransmit, even when the cause of the delay was a misevaluation. The misevaluation check timer sets the product value of the overtaking width and the average packet arrival interval according to the eight most recent packets.

IV. EVALUATION

We conducted an experiment to evaluate the effectiveness of the proposed method using machines operating on a Debian 7.5 OS with kernel 3.2.0. We implemented the proposed method as a kernel module in the Linux OS. We compared eight patterns, TCP Reno and TCP CUBIC, TCP Reno with SACK, and TCP CUBIC with SACK, applying the proposed method to each pattern. The experiment topology was similar to that in Fig. 4, with six nodes in 9 m × 14 m, indoors. The source and destination nodes could not communicate directly, and we minimized the transmission power using a fabric wall between them. Each node used IEEE 802.11b, and the channel capacity was 11 Mbps. Each node runs the OLSRd [16] Linux daemon of the OLSR routing protocol with the dynamic metrics of the ETX. We conducted an FTP file transfer from the source to the destination node over this connection for 51.2 MB (100,000 packets), and the

ISBN: 978-988-19253-9-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

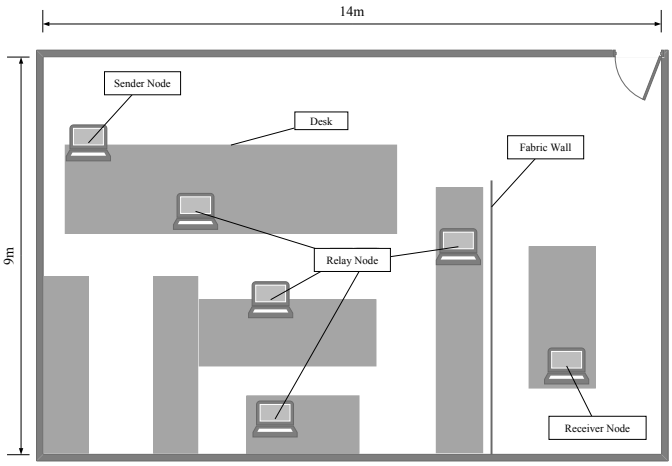

Fig. 4. Evaluation topology.

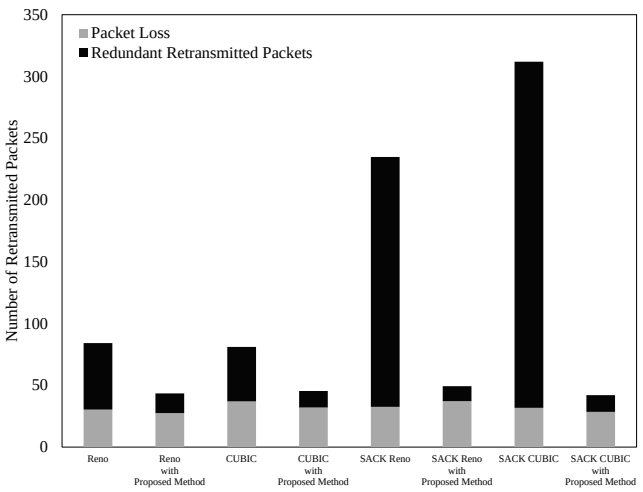

Fig. 5. Effect of reducing retransmission control.

packet size and MSS were 512 bytes. In each scenario, we performed 90 experiments, and the results shown are average values.

A. Results

Fig. 5 shows the total number of retransmitted packets for each congestion control algorithm. The number of packets lost, which is calculated from the difference between the total number of transmitted and received data packets, indicate the number of packets that must be retransmitted. Thus, the number of redundant retransmitted packets is the difference between the total number of retransmitted packets and the number of packets lost. When the proposed method was not applied, of the 85 retransmission packets, 60% of the packets retransmitted by Reno and CUBIC were redundantly retransmitted. In contrast, when the proposed method was applied, the number of retransmitted packets was reduced by 55% in Reno and CUBIC. Of the SACK Reno and SACK CUBIC retransmitted packets, 90% were redundantly transmitted. The SACK option erroneously recognizes packet loss as not all overtaken packets result from order errors by rerouting, so redundant retransmission occurs. When the proposed method was applied to the SACK options, the total number of redundant retransmission packets was reduced by 80% in SACK Reno and 87% in SACK CUBIC.

Fig. 6 shows the goodput and round trip time (RTT) for each congestion control algorithm. CUBIC and SACK CUBIC improved their goodput by about 1%, and reduced their RTTs by 4%. On the other hand, Reno and SACK Reno improved their goodput by only 0.4%, and showed no significant change in the RTT. From the above, using CUBIC, we confirmed the effectiveness of the proposed method. However, Reno showed reduced effectiveness. The ineffective results for Reno are because its window size is smaller than that in CUBIC.

Fig. 7 shows the instantaneous throughput for SACK CUBIC. SACK CUBIC with the proposed method is shown with a solid line, and the regular SACK CUBIC is shown with a broken line. SACK CUBIC performs redundant retransmissions and congestion control by making route changes. In this experiment, the rerouting occurred in cycles of roughly 10 s. SACK CUBIC regularly showed a decreased throughput. Conversely, the proposed method can be used to maintain a

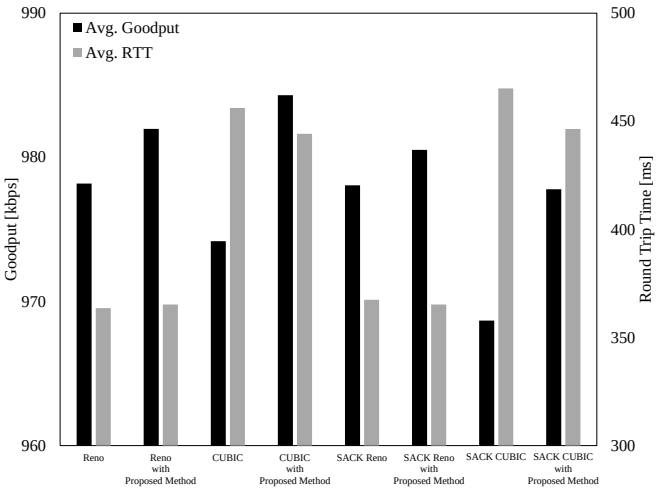

Fig. 6. Throughput and RTT for each congestion control algorithm.

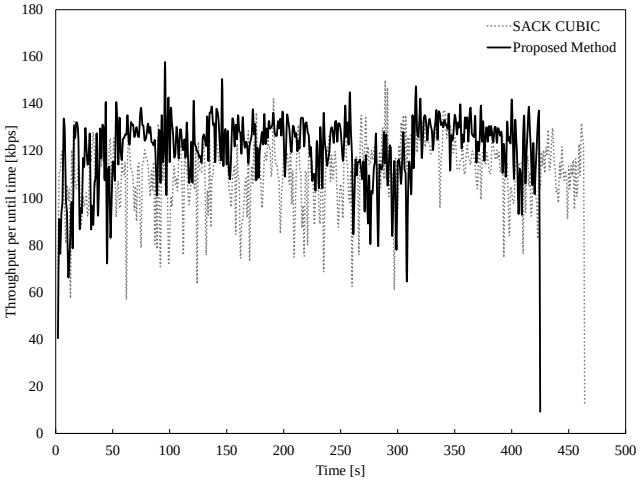

Fig. 7. Instantaneous throughput with SACK CUBIC and with the proposed method.

high overall throughput and is less affected by route changes. It completed the transfer of more early files compared with SACK CUBIC. The method reduced throughput time by 250–300 s, because the link quality of the originally selected path had deteriorated. The method can also determine out-of-order packets due to burst-type packet loss and route changes. Therefore, the proposed method is unaffected even if the rate

ISBN: 978-988-19253-9-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

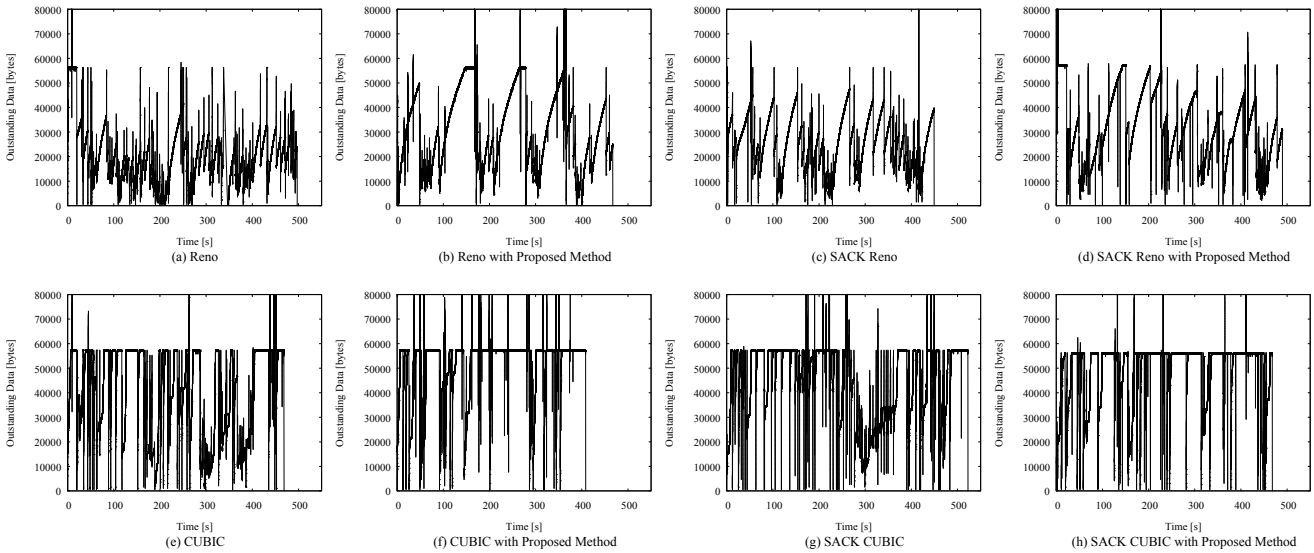

Fig. 8. Instantaneous window size for each congestion control algorithm.

of packet loss has increased. If delays in the ACK packet are erroneously determined as the cause for order errors, communication temporarily stops and RTO occurs.

B. Discussion

From the experiment results, we confirmed that the proposed method is effective in reducing the number of redundantly retransmitted packets and improving network throughput. The proposed method reduced retransmitted packets by 55% in Reno and CUBIC and by 80–87% in SACK Reno and SACK CUBIC. The reason the SACK option experiences an increased number of redundant retransmission packets is because it erroneously recognizes packet loss; not all overtaken packets are detected as out-of-order by rerouting. In this evaluation, the average width of the overtaken packets is 53 in Reno and 64 in CUBIC. The SACK options mistook overtaken packets as packets lost, increasing its number of retransmission packets in an attempt to retransmit them. The SACK CUBIC result is due to the overtaking width of the redundant retransmission packet being greater than that in SACK Reno. Furthermore, the throughputs of Reno and CUBIC were temporarily reduced when they were receiving duplicate ACKs because of their loss-based congestion control algorithms. The proposed method can also suppress redundant congestion control by discarding the duplicate ACKs at the destination node. The wireless channel is characterized by the transmission of the frames affecting neighboring nodes. Thus, the proposed method can also be used to mitigate the effects of the neighbor nodes and the intra-flow interference by reducing redundantly transmitted packets.

Reno can be applied with the proposed method, and its goodput and RTT do not change as compared with those of CUBIC, because the window size remains lower than that of CUBIC. Fig. 8 shows the instantaneous window size for each congestion control algorithm. We calculated this graph of outstanding data using the tcptrace [17] from PCAP. Reno (Fig. 8(a)) and SACK Reno (Fig. 8(c)) maintain a lower window size, which influences the route change. On the other hand, with the proposed method (Fig. 8(b), (d)), it is possible to maintain a higher window size than that in Reno. A similar trend in shown in CUBIC (Fig. 8(e), (f), (g), (h)). In other words, in Reno and CUBIC, it is possible to suppress redundant congestion control. However, Reno's window size remained lower than that of CUBIC. Reno reduces its window size by half when it detects packet loss. A larger window size cannot be maintained in environments affected by a high packet loss ratio for wireless link and route changes. Reno tends to have TCP zero window conditions because of its small window size. When the delayed ACK packets are sent by the proposed method, the sender node temporarily stops sending data packets. As a result, Reno's stops in transmission can also temporarily suppress redundant congestion control, and this does not improve communication performance. CUBIC always attempts to maintain its window size in areas close to congestion. CUBIC can also delay ACK packets using the proposed method and continues to transmit data packets because there is sufficient window space. From the above results, we conclude that to improve the communication performance in the proposed method, it is necessary to increase window size.

In the experiments, it was difficult to observe packet loss because fewer nodes were involved. The experiments were also less able to determine the cause of erroneous determination of out-of-order packets by the proposed method when bursts in packet loss occurred. When we evaluated the QualNet experiments for 50 nodes of 1000 m $\times$ 1000 m, we obtained an 89% correct determination rate for order errors. For the erroneously determined 11%, RTO occurrence can be minimized using the misevaluation check timer. This experiment was small, with two hops and four selectable routes. Accordingly, the influence of the route change was also small. We can expect further advantages in using the proposed method by increasing the network size.

V. RELATED WORK

TCP-DOOR [18] focuses on arrival order errors due to route changes. and is a method for suppressing redundant

ISBN: 978-988-19253-9-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

congestion control that occurs during path switching in ad-hoc communication. The routing protocol is assumed to be one of reactive routing protocol (e.g., DSR, AODV). TCP-DOOR detects out-of-order packets on the basis of the original sequence number, which is the transmission order (including retransmission packets) of all data packets and ACK packets. When out-of-order events are detected, a source node is temporarily disabled either by congestion control or by recovering the out-of-order packet during congestion avoidance. TCP-DOOR can be expected to improve performances in environments without packet loss [19]. However, TCP-DOOR does not assume packet loss, and, consequently, goodput decreases as the packet loss rate increases [20]. The wireless environment is influenced by interference from neighboring nodes with a high packet loss ratio. The proposed method uses the arrival interval and TCP sequence number to determine packet loss and route change. The proposed method does not require additional information to be present in each packet.

Approaches to delay the processing of the ACK packet include the Paxson algorithm [21] and TCP-DCR [22]. The Paxson algorithm uses an additional delay time before a destination node sends duplicate ACKs. It suppresses redundant retransmission control and congestion control to 20 ms between the first and second duplicate ACKs. However, the Paxson algorithm does not provide a mechanism for determining how to set an optimal delay time, and the 20 ms delay time is based on observation. TCP-DCR is a method for suppressing redundant congestion control and retransmission, to delay congestion control by one RTT upon receiving duplicate ACKs. TCP-DCR assumes an environment in which wireless and wired connections are mixed, divides these mixed and wireless connections, and then performs congestion control and retransmission control in each. The Paxson algorithm and TCP-DCR do not distinguish between rerouting and packet loss. Since packet loss delays the processing of ACK packets, this effect cannot be expected in a wireless mesh environment. The proposed method can minimize RTO using the misevaluation check timer, even if it erroneously determines the cause for an out-of-order packet. The time of the misevaluation check timer is adjusted according to the communication environment and the width of the overtaking packets.

VI. CONCLUSION

In this study, we proposed a method for suppressing redundant retransmissions due to route changes. When a combination of the LSR protocol and dynamic metrics is used, route modification occurs frequently. In this manner, out-of-order events occur, followed by retransmission. TCP mistakes the out-of-order event for packet loss and invokes retransmission control. However, retransmission control for cases involving no packet loss is redundant. The proposed method determines the cause of the out-of-order event depending on the difference between the arrival interval and the sequence number at the destination node. It then delays ACK packets as duplicate ACKs, as determined by route modification. When the source node does not receive duplicate ACKs, it suppresses redundant retransmissions and congestion control. The results evaluated in Linux show a 50% reduction in the number of redundantly retransmitted packets in Reno and CUBIC and 90% reduction in SACK Reno and SACK CUBIC.

REFERENCES

[1] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," RFC3561, 2003.
[2] D. Johnson, Y. Hu, and D. Maltz, "The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4," RFC4728, 2007.
[3] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," RFC3626, 2003.
[4] P. Guangyu, G. Mario, and C. Tsu-Wei, "Fisheye state routing in mobile ad hoc networks," in *Proc. ICDCS*, 2000, pp. 71–78.
[5] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A High-Throughput Path Metric for Multi-Hop Wireless Routing," in *Proc. ACM MobiCom '03*, 2003, pp. 134–146.
[6] R. Draves, J. Padhye, and B. Zill, "Routing in Multi-radio, Multi-hop Wireless Mesh Networks," in *Proc. ACM MobiCom '04*, 2004, pp. 114–128.
[7] X. Ni, K.-c. Lan, and R. Malaney, "On the Performance of Expected Transmission Count (ETX) for Wireless Mesh Networks," in *Proc. ICST ValueTools '08*, 2008, pp. 1–10.
[8] E. Gelenbe and M. Gellman, "Can Routing Oscillations Be Good? The Benefits of Route-switching in Self-aware Networks," in *Proc. IEEE MASCOTS '07*, 2007, pp. 343–352.
[9] Z. Zaidit, T. Tan, and Y. Cheng, "ETX Could Result in Lower Throughput," in *Proc. IEEE ICCCN '09*, 2009, pp. 1–6.
[10] K.-C. Leung, V.-K. Li, and D. Yang, "An Overview of Packet Reordering in Transmission Control Protocol (TCP): Problems, Solutions, and Challenges," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 4, pp. 522–535, 2007.
[11] S. Aketa, Y. Otsuki, K. Mouri, and E. Takimoto, "Improving the Suppression Method of Redundant TCP Retransmission Caused by Route Modification," in *IEICE Technical Report*, ser. MoNA2013-46, vol. 113, no. 304, 2013, pp. 29–34.
[12] QualNet, http://www.scalable-networks.com/content/qualnet.
[13] S. Floyd and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm," RFC2582, 1999.
[14] J. Nagle, "Congestion Control in IP/TCP Internetworks," RFC896, 1984.
[15] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgment Options," RFC2018, 1996.
[16] "olsrd — an adhoc wireless mesh routing daemon," http://www.olsr.org/.
[17] tcptrace OfficialH̃omepage, http://www.tcptrace.org.
[18] F. Wang and Y. Zhang, "Improving TCP Performance over Mobile Ad-Hoc Networks with Out-of-Order Detection and Response," in *Proc. ACM MOBIHOC '02*, 2002, pp. 217–225.
[19] A. Alheid, D. Kaleshi, and A. Doufexi, "An Analysis of the Impact of Out-of-Order Recovery Algorithms on MPTCP Throughput," in *Proc. IEEE AINA '14*, 2014, pp. 156–163.
[20] D. Yang, K.-C. Leung, and V. Li, "Simulation-Based Comparisons of Solutions for TCP Packet Reordering in Wireless Networks," in *Proc. IEEE WCNC '07*, 2007, pp. 3238–3243.
[21] V. Paxson, "End-to-end Internet Packet Dynamics," in *Proc. ACM SIGCOMM '97*, 1997, pp. 139–152.
[22] S. Bhandarkar, N. Sadry, A. Reddy, and N. Vaidya, "TCP-DCR: A Novel Protocol for Tolerating Wireless Channel Errors," *IEEE Transactions on Mobile Computing*, vol. 4, no. 5, pp. 517–529, 2005.

ISBN: 978-988-19253-9-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)