Error Recovery of RDMA Packets in Data Center Networks

Yi Wang
School of Electronic Information and Communications
Huazhong University of Science and Technology
Wuhan, China, 430074
Email: ywang@hust.edu.cn
Phone: +86-027-87543236

Kexin Liu
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China, 210023
Email: kxliu@smail.nju.edu.cn
Phone: +86-25-89681372

Chen Tian
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China, 210023
Email: tianchen@nju.edu.cn
Phone: +86-25-89681372

Bo Bai
Future Network Theory Lab
Huawei
Hong Kong, China, 999077
Email: baibo8@huawei.com
Phone: +852-3547-2722

Gong Zhang
Future Network Theory Lab
Huawei
Hong Kong, China, 999077
Email: nicholas.zhang@huawei.com
Phone: +852-3547-2722

***Abstract*—Modern data center applications need high throughput (40Gbps) and ultra-low latency (<10us per hop), along with low CPU overhead. Remote Direct Memory Access (RDMA), which can be deployed in RDMA over commodity Ethernet (RoCEv2) protocol, has the potential to satisfy the requirements. RoCEv2 needs a lossless environment to achieve high performance. RoCEv2 provides Priority-based Flow Control (PFC) to prevent packet loss caused by buffer overflow. But packet loss can still happen in today's data centers due to other reasons such as switch configuration error. There are two retransmission algorithms dealing with the packet loss recovery: Go-Back-0 and Go-Back-N. Unfortunately, by simply applying Go-Back-N algorithm to RoCEv2, the relative throughput will drop to nearly zero when the packet loss rate exceeds 1%. This is mainly caused by the improper triggering mechanism of generating NAK. This paper proposed an Improved Go-Back-N algorithm to solve this problem, which involves two mechanism. The Improved Go-Back-N is easy to be deployed in today's data centers because it makes no changes on switches. It can improve the relative throughput to about 60% when the packet loss rate increases to 1%.**

***Index Terms*—RDMA, Loss recovery, Data Center Networks.**

I. Introduction

Large-scale data centers are built around the world for increasing online services, in which data center networks are critical to data center infrastructure [6], [18], [30], [17], [7]. Modern data center applications such as cloud computing need high bandwidth (40Gbps) and ultra-low latency (<10us per hop) to meet increasing demands. Traditional TCP/IP stacks cannot meet these requirements because it suffers from high CPU overhead [14], [1], [13], [12]. As a promising technology, Remote Direct Memory Access (RDMA) [4], [22] is capable of achieving the stringent performance requirements. RDMA is implemented entirely on the network interface cards (NICs), bypassing the host networking stack, which reduces CPU overhead and latency significantly.

RDMA can be deployed by using InfiniBand Architecture (IBA) [8], [10]. InfiniBand (IB) supports RDMA with single-sided operations. Specifically, a server registers a memory buffer in advance, then clients read (write) from (to) it without further involvement of the server's kernel. Thus CPU overhead is reduced. However, the IB networking stack cannot be deployed in traditional data centers, because traditional data centers are built with IP and Ethernet technologies. To enable RDMA over Ethernet and IP networks, i.e., making use of traditional data center equipments, RDMA over Converged Ethernet (RoCE) [9] and RDMA over commodity Ethernet (RoCEv2) [11] have been proposed. RoCEv2 replaces IB link layer with Ethernet link layer, and replaces IB network layer with UDP and IP. The UDP header is used for ECMP and IP header is used for routing. Similar to IB, RoCEv2 must be deployed over a lossless link layer to enable high performance, so that Priority-based Flow Control (PFC) [5] is used. PFC helps the switch to avoid buffer overflow by forcing the upstream switch and host NIC to pause data transmission, when the occupancy of buffer exceeds the threshold. PFC propagates from the congestion point to the source, while there may be several hops from the source server to the destination server. This can result in unfairness and victim flow [31]. Besides, PFC has many other side effects such as head-of-line blocking and potential for deadlock [21], [23]. In order to reduce the side effects, flow based congestion control mechanisms such as QCN [26], DCQCN [31], [2], [32], and TIMELY [25] have been proposed. DCQCN requires only the standard RED [15] and ECN [28], [27], [29] support from the data center switches. It provides fast convergence to fairness, achieves high link utilization, ensures low queue buildup and low queue oscillations. However, PFC should still be deployed when DCQCN is used.

978-1-7281-1856-7/19/$31.00 ©2019 IEEE

While IB provides PFC to prevent packet loss, the packet loss can still happen in today's data center. There are many other reasons for packet loss besides buffer overflow, such as frame check sequence (FCS) errors, bugs in switch hardware and software, and network configuration errors [20]. When packet loss occurs, retransmission mechanism should be involved. There are two retransmission algorithms dealing with the packet loss in RDMA: Go-Back-0 and Go-Back-N. In Go-Back-0, when packet loss errors, the sender will resend the entire message to the receiver, i.e., resend from the very first packet of the message. It will waste much network bandwidth. In Go-Back-N, retransmission starts from the first dropped packet and the previous received packets are not retransmitted. Go-Back-N is almost as simple as Go-Back-0, it can be implemented on the NIC, and it can save much bandwidth compared to Go-Back-0 and avoid livelock [19]. RoCEv2 does not support selective packet retransmission or the out-of-order reception of packets since the memory of NIC is small, i.e., complex algorithm such as Selective acknowledgement (SACK) [16] cannot be implemented on it.

However, when Go-Back-N is applied, we find that the network throughput will decrease to nearly zero when the packet loss rate increases to 1%, which will hurt the application performance badly [31]. The reason of this problem is that ACK timeout, i.e., timeout retransmission, is triggered. And the main root cause of ACK timeout is that negative acknowledgement (NAK) can only be triggered by the unexpected packets and there is no timer mechanism for NAK.

We propose an Improved Go-Back-N algorithm, which involves sending twice of the last packet and checking expected packet sequence number (PSN) mechanisms, to handle the low throughput problem in lossy RDMA network environment. In our paper, we also discussed about the ACK timeout setting issues. We can also change the ACK timeout value to improve the throughput, however, it's hard and challenging to be general. By using our improved Go-Back-N algorithm, the throughput is increased significantly. When the loss rate is less than 0.1%, the throughput is only about 2.8% lower than the lossless situation. And the relative throughput can recover to about 60% when the loss rate increases to 1%. The Improved Go-Back-N is easy to be deployed in today's data centers because it requires only the settings on the end hosts and makes no changes on switches.

II. Background and Motivation

A. Retransmission Algorithm in RoCEv2

There are two retransmission algorithms dealing with the packet loss in RoCEv2: Go-Back-0 and Go-Back-N. We now introduce these algorithms in details.

Consider a scenario, where two hosts are connected through a switch. After establishing a reliable connection between these hosts, the sender begins to send data to the receiver. The sender will set the Opcode in the Base Transport Header (BTH) of the first packet (PSN=0) to "First", indicating that this is the first packet sending to the receiver. The Opcode of the middle packets is set to"Middle". The Opcode of the last packet is set to "Last". In the last packet, the sender will set the ACK Request bit to true, indicating that it requests the reply from the receiver. As for middle packets, sender will request the reply by a specific interval, such as 64/128. Besides, the sender and receiver will negotiate an ACK timeout value when establishing the connection. When the timeout expires as well as the receiver doesn't receive the desired reply, the retransmission mechanism will be triggered.

The reliable transport service uses a combination of the sequence number and ACK / NAK. These mechanisms are aimed at verifying the packet delivery order, preventing duplicate packets and out-of-sequence packets from being processed, and detecting missing packets. If packet (PSN=n) is lost, the receiver will observe the packet loss when it receives the packet (PSN=n+1). Thus, the receiver generates a NAK (PSN=n) and sends it back to the sender. This NAK not only tells the sender that the receiver hasn't received the packet (PSN=n), but also delivers a signal to the sender that the receiver has already successfully received all the packet with PSN$<$n.

RoCEv2 defines a NAK interval which specifies the duration in which the sender can generate at most one NAK, i.e., the receiver generates a NAK when it receives the wrong packet and won't generate more NAK in a specific NAK interval for the subsequent wrong packets. The NAK interval is necessary. When packet (PSN=n) loss occurs, the receiver sends back NAK (PSN=n) to the sender. At the same time, the sender will continue sending packets whose sequence numbers are not expected before it receives the NAK. If NAK interval is not involved, the receiver will send back NAK (PSN=n) once it receives those subsequent unexpected packets (PSN$>$n). This can result in redundant NAK (PSN=n) packets so that the network bandwidth will be wasted.

In Go-Back-0, when receiving the NAK (PSN=n), the sender will detect the packet loss and retransmit the whole message to the receiver, i.e., retransmit from the first data packet. In Go-Back-0, the sender doesn't need to save the state for retransmission. However, Go-Back-0 will waste much bandwidth, even may cause livelock [19].

Go-Back-0 is replaced by Go-Back-N when deploying RDMA. In Go-Back-N, after receiving the NAK (PSN=n), the sender will detect the packet loss. The packet (PSN=n) and all subsequent packets will be retransmitted by the sender. Go-Back-N is almost as simple as Go-Back-0, but avoids livelock and achieves higher throughput.

It can be seen that in contrast to retransmission algorithm in TCP which uses the duplicate three ACK to trigger fast retransmission, RoCEv2 uses NAK to send back the loss signal to the sender directly. Besides, RoCEv2 defines several transport layer operations such as SEND and WRITE. Data packets are encapsulated in operations to be transmitted. Packets in an operation is called as a message.

B. Low Throughput Issue

PFC prevents packet loss caused by buffer overflow. RoCEv2 needs a lossless network and assumes that the packet

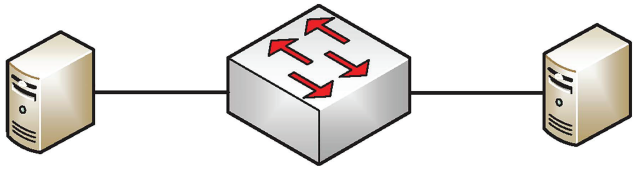


Fig. 1: Network topology

Parameter	Value
Bandwidth	40GB
Propagation Delay Per Hop	1us
Message Size	4MB
Packet Payload	1KB
ACK Timeout	100ms
ACK Interval	256KB
NAK Interval	500us

TABLE I: Parameter values used in our simulation

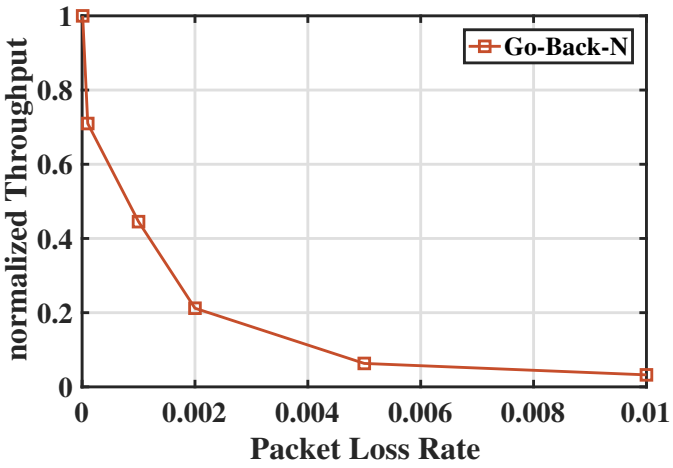


Fig. 2: Impact of packet loss rate on throughput

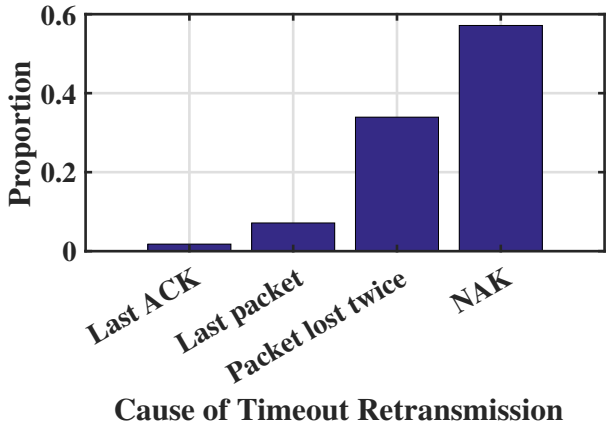


Fig. 3: Proportion for each reasons

loss is a rare event by using PFC. However, there are many other errors that can result in packet loss in data centers such as frame check sequence (FCS) errors, bugs in switch hardware and software, and network configuration errors. Go-Back-N algorithm in RoCEv2 is sufficient when the packet loss rate is low. However, the throughput degrades rapidly when the packet loss rate exceeds 0.1%.

Considering the simple topology as Figure 1, we use ns-3 to simulate the performance of Go-Back-N. The result is shown as Figure 2, which is consistent with the result in DCQCN. When the loss rate is less than 10^{-5}, the throughput is not influenced. When the loss rate increases to 10^{-3}, the throughput decreases to 45%. When the loss rate increases to 10^{-2}, the throughput decreases to 3%.

This problem will hurt the performance of RDMA applications. Thus it cannot meet high throughput and low latency requirements. The performance of application which has short flows to send or doesn't support the pipeline mode, i.e., should wait for the previous flows to complete before it begins to send new flows, will be hurt badly.

Table I shows the parameter setting in this paper. ACK Timeout is retransmission timeout (RTO). ACK Interval is the interval that the sender requests an ACK feedback. Here 256KB implies that the sender will set the ACK Request bit to true per 256 packets (when the payload for each packet is 1KB). NAK interval specifies the duration in which the sender can generate at most one NAK.

III. System Design

A. Problem Analysis

By tracking the flow passing through the node, the key for the low throughput is that ACK TIMEOUT is triggered, i.e., the timeout retransmission. The timeout becomes the critical issues to hurt the throughput of application which has small flows to send and the application which doesn't support the pipeline mode. By analysizing the root cause triggering the ACK TIMEOUT, we find out four reasons that can result in timeout retransmission as shown in Figure 4.

Reason 1. The Last packet of the message is lost.

Reason 2. The ACK (PSN=Last) is lost.

Reason 3. Some NAK is lost.

Reason 4. The same packet is lost twice.

We now analysize these four reasons. When the last packet of the message is lost, as shown in Figure 4(a), the sender is waiting for the ACK and the receiver is waiting for the data packet. The situation will not make any progress before the ACK Timeout running out.

When the ACK (PSN=Last) is lost, as shown in Figure 4(b), the sender is waiting for the ACK and the receiver is waiting for the data packet. Thus resulting in ACK Timeout.

RoCEv2 defines NAK Interval, during which the receiver can only send at most one NAK. Consider such a situation as shown in Figure 4(c). The NAK is lost. The receiver is waiting for the NAK Interval to run out to generate NAK again. The sender has sent the entire message to the receiver before it receives NAK. Thus the sender begins to wait for the ACK from the receiver. Since the whole message is sent, there is no

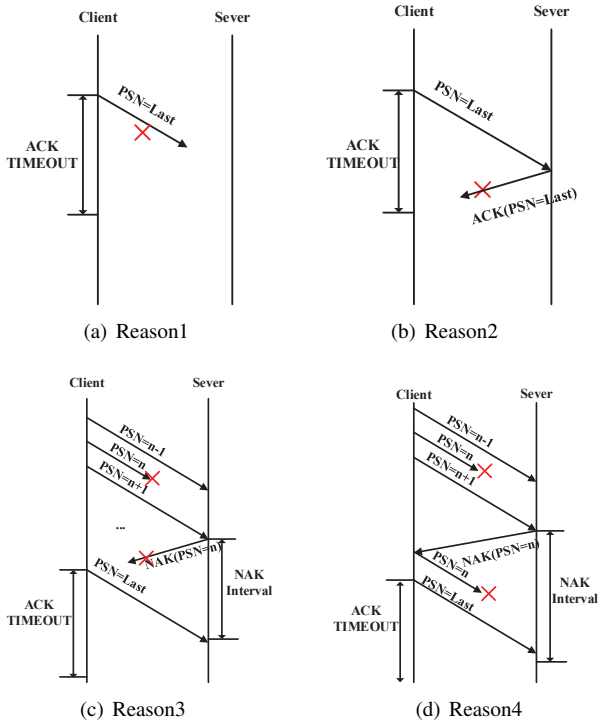


Fig. 4: Reason triggering the timeout retransmission

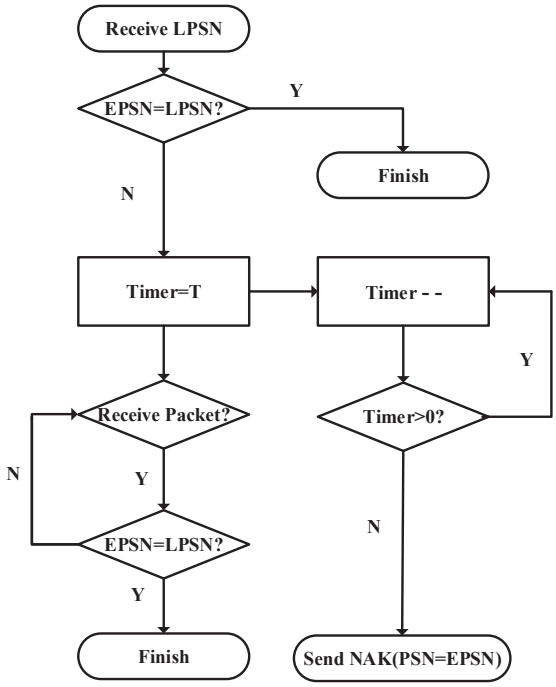


Fig. 5: Check EPSN

more unexpected packets to receive when the NAK Interval runs out. Unfortunately, the NAK can only be triggered by unexpected PSN of packets. The only way to complete this transmission is waiting for the ACK Timeout to run out, thus triggering the timeout retransmission.

When the same packet is lost twice, as shown in Figure 4(d). Because of the same reason as mentioned above: the sender has sent the entire message to the receiver, while the receiver is waiting for NAK Interval to run out to generate a new NAK. Thus ACK Timeout is triggered.

The root cause of reason3 and reason4 is the improper triggering mechanism for NAK, i.e., NAK can only be triggered by data packets, and no timer mechanism is involved. We will discuss the improvement of NAK triggering mechanism later.

We set the packet loss rate to 1% and repeat the simulation 100 times. The ratio of four reasons is shown as Figure 3. The lost of last ACK (reason2) causes 1.8% timeout retransmission. The lost of last packet (reason1) causes 7.1% timeout retransmission. The twice lost of packets (reason4) causes 33.9% timeout. The lost of NAK (reason3) owns the largest propotion, i.e., 57.1%. It shows that reason3 and reason4 owns about 90%. It can be seen that the improper triggering mechanism for NAK is the root cause of the low throughput issue.

B. Design Overview

There are two solutions to improve the performance of Go-Back-N algorithm. One way is to avoid triggering the timeout retransmission, the other one is to reduce the ACK Timeout value to a lower level. However, the ACK Timeout value is related to the network topology, switch queueing delay and so on. Thus, we cannot generally set it to a specific low level. It's also an open problem in TCP/IP, too large timeout value will waste bandwidth, increase application completion time; too small timeout value will trigger unnecessary retransmission, occupy bandwidth, and even cause network congestion. We use the first way to improve the throughput under lossy networks. We also compare its performance under different ACK Timeout values.

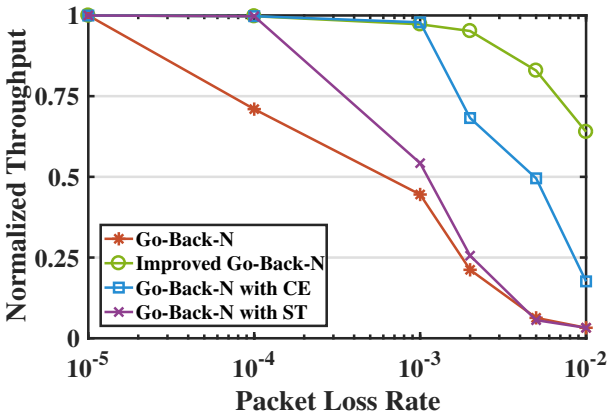
C. Design Details

The Improved Go-Back-N retransmission algorithm is proposed, which contains the next two mechanisms.

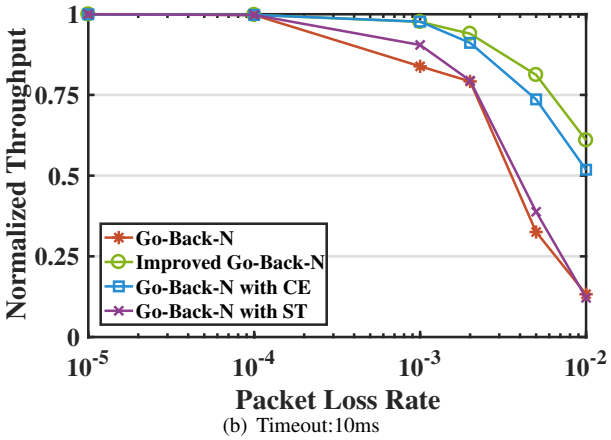
1. ST (Send Twice) : Send the last packet of the message twice.

2.CE (Check EPSN) : Adding the Go-Back-N a timer mechanism to trigger the generation of NAK besides the unexpected packet sequence number. When receiving the last packet of the message, if expected PSN<Last PSN, record the left time of NAK Interval T, the expected PSN (EPSN) and the PSN of the last packet of the message (LPSN). After time T running out, if EPSN<LPSN, send back NAK (EPSN), set the timer to the initial value (NAK interval), and start the timer again. If the packet (EPSN) is received during the time T, only set the timer to the initial value, as shown in Figure 5.

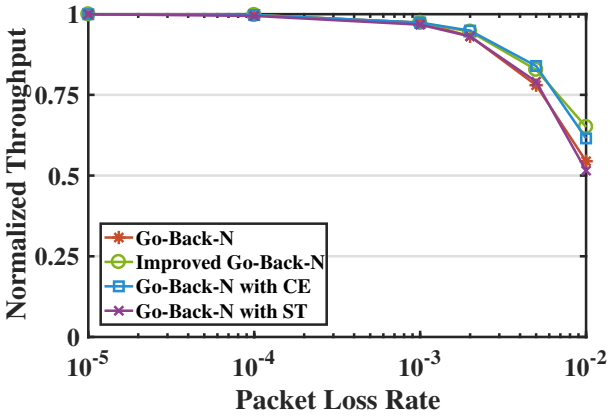
ST reduces the timeout retransmission caused by reason1 and reason2. CE reduces the timeout retransmission caused



(a) Timeout:100ms



(b) Timeout:10ms



(c) Timeout:500us

Fig. 6: Throughput comparison

by reason3 and reason4, i.e., the improper NAK triggering mechanism.

IV. Evaluation

In this section, we evaluate the performance of the Improved Go-Back-N retransmission algorithm by using ns-3[3]. The parameter setting is the same as Table I. We compare the proposed Improved Go-Back-N algorithm, which applies the ST and CE mechanisms at the same time, and the algorithm applying the ST/CE mechanism separately.

A. Performance of Improved Go-Back-N

We compare the algorithm performance under different packet loss rate settings. The results are shown as Figure 6. When setting the ACK Timeout value to 100ms, as shown in Figure 6(a), the Improved Go-Back-N algorithm is best. The second is Go-Back-N with CE, and the third is Go-Back-N with ST. The traditional Go-Back-N plays the worst. When the packet loss rate is less than 0.1%, since the twice loss of packet and the loss of NAK play a main role in timeout retransmission, Go-Back-N with CE recovers the throughput to about 97.2% compared to that in traditional Go-Back-N. Go-Back-N with CE plays even a little better than the Improved Go-Back-N, because the ST mechanism applied to the Improved Go-Back-N caused some unnecessary transmission of the last packet. When the packet loss rate increases to 1%, the effect of the loss of last ACK and the last packet become not negligible. Thus the performance of Go-Back-N with CE decreases to a low level (but still better than the Go-Back-N with ST and the traditional Go-Back-N) . The performance of the proposed Improved Go-Back-N is still good whose throughput is about 64%.

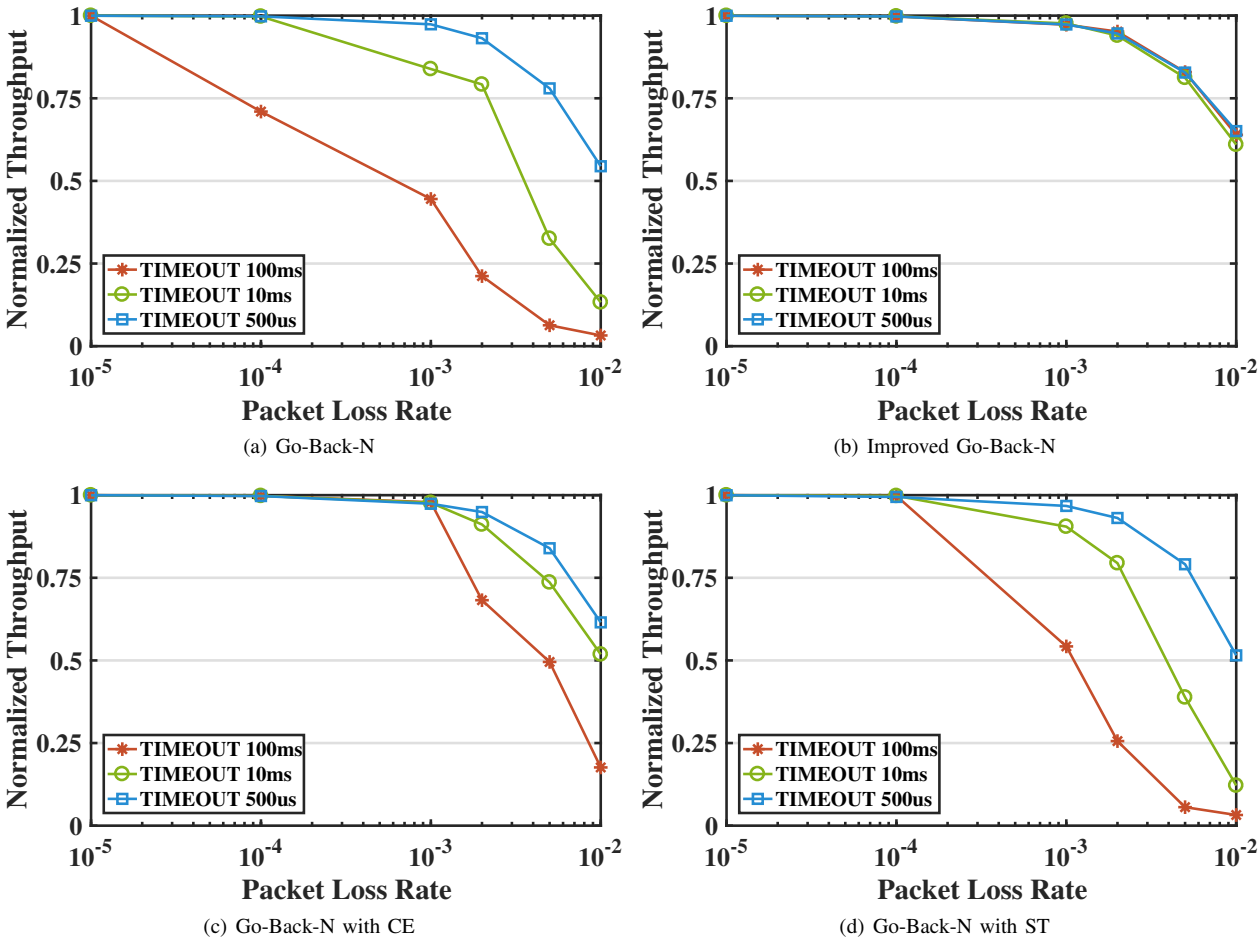
It can be seen that the throughput of original Go-Back-N begins to decrease to 70% when the packet loss rate increases to 0.01%. It decreases more sharply when the packet loss rate continues increasing. When the packet loss rate increases to 0.1%, Go-Back-N with ST plays only a little better than the original Go-Back-N, which is because the twice loss of data packets and the loss of NAK occupy a large proportion.

Figure 6(b) shows that when setting the ACK timeout value to 10ms, the Improved Go-Back-N algorithm plays the best. When the packet loss rate increases to 1%, the throughput maintains around 60%. The Go-Back-N with CE plays a little worse than Improved Go-Back-N when the packet loss rate is greater than 0.1%. Go-Back-N with CE also plays much better than the Go-Back-N with ST and the original Go-Back-N algorithm. In conclusion, the CE mechanism plays an important role even when the timeout value is much smaller.

To verify that the ACK Timeout is the key to the decrease of the throughput, we set the ACK Timeout as 500us (a little larger than the RTT), as shown in Figure 6(c). It can be seen that four algorithms play almost the same which is identical to our analysis that the timeout retransmission is the critical issue to the low throughput. The Improved Go-Back-N and the Go-Back-N with CE play about 10% better than the Go-Back-N with ST and the original Go-Back-N when the packet loss rate increases to 1%. However, this timeout setting is not general and cannot be applied to data center networks.

B. Algorithm Performance Under Different Timeout Setting

We now compare the algorithm under different timeout values, as shown in Figure 7. Figure 7(b) shows that the proposed Improved Go-Back-N performs almost the same



(a) Go-Back-N

(b) Improved Go-Back-N

(c) Go-Back-N with CE

(d) Go-Back-N with ST

Fig. 7: Algorithm performance under different timeout setting

under different ACK timeout values. Thus, the Improved Go-Back-N algorithm is robust for different ACK timeout values. Therefore, We don't need to know the network topology or the congestion extent in advance, when applying the Improved Go-Back-N algorithm. However, it is not the same situation for the other three algorithms.

The performance of original Go-Back-N is largely influenced by the timeout values, as shown in Figure 7(a). When setting the timeout value to 100ms, the throughput decreases to nearly 0% when the packet loss rate reaches 1%. When setting the timeout value to 10ms, the throughput begins to decrease sharply when the packet loss rate exceeds 0.2%. The throughput decrease to 13.2% when the packet loss rate reaches 1%.

Go-Back-N with CE performs well whatever the ACK Timeout value is when the packet loss rate is less than 0.1%, as shown in Figure 7(c). When the packet loss rate continues increasing, however, it performs not so good when setting the timeout value to the default 100ms. The throughput decreases to 68% when the packet loss rate is 0.2% and the throughput is only 17.6% when the packet loss rate is 1%. In this content, the timeout retransmission still plays an important role in affecting the performance of the throughput. When setting the timeout value to 10ms, the throughput is much better and just a little worse than the case where timeout value is set to 500us.

For Go-Back-N with ST, as shown in Figure 7(d). When setting the timeout value to 100ms and 10ms, the throughput begins to decrease when the packet loss rate reaches 0.01%. The throughput is nearly 0% when the packet loss rate is 1%. Thus, the Go-Back-N with ST cannot avoid the most of the timeout retransmission. When setting the timeout value to 500us, the performance is much better.

V. Related Work

Various improved Go-Back-N mechanisms propose to utilize the network throughput. Most of them consider only TCP, and cannot be directly used for RDMA. There are some related work on RDMA mentioned the mechanism of loss recovery, but they do not design it carefully.

IRN [4]: IRN is an improved RoCE NIC design. IRN proposes a simplified form of selective acknowledgement. IRN maintains a bitmap to track which packets have been cumulatively and selectively acknowledged. However, it cannot avoid triggering ACK timeout. It only uses RTO_{min} and RTO_{max} to relieve the low throughput problem caused by ACK timeout.

MP-RDMA [24]: MP-RDMA presents a multi-path transport for RDMA. MP-RDMA simply retransmits unacknowledged packets as new data if there is no new data to transmit and *awnd* allows. However, when the proportion of small flows is large, many duplicate packets will be induced and the network throughput will downgrade.

VI. Conclusion

This paper proposed an Improved Go-Back-N algorithm, which is very easy to deploy on current data center networks. The proposed algorithm is almost as simple as original Go-Back-N, i.e., requiring only the settings on the end hosts and making no changes on switches. Our Improved Go-Back-N makes two changes on traditional Go-Back-N retransmission algorithm, ST (Send Twice) and CE (Check EPSN), so as to avoid most of the timeout retransmission. The simulation shows that it improves the throughput efficiently: when the packet loss rate increases to 1%, the throughput can still reach about 60% compared to lossless condition.

Acknowledgment

The authors would like to thank anonymous reviewers for their valuable comments. This research is supported by the National Key R&D Program of China 2018YFB1003505, the National Natural Science Foundation of China under Grant Numbers 61602194, 61772265, and 61802172, the Collaborative Innovation Center of Novel Software Technology and Industrialization, and the Jiangsu Innovation and Entrepreneurship (Shuangchuang) Program.

Reference

[1] Hadoop. http://hadoop.apache.org/.

[2] ns3-rdma. https://github.com/bobzhuyb/ns3-rdma.

[3] The ns3 simulator. https://www.nsnam.org/.

[4] Revisiting network support for RDMA, author=Mittal, Radhika and Shpiner, Alexander and Panda, Aurojit and Zahavi, Eitan and Krishnamurthy, Arvind and Ratnasamy, Sylvia and Shenker, Scott, booktitle=Proceedings of the ACM SIGCOMM Conference on Data Communication, Budapest, Hungary, Aug. 2018.

[5] IEEE. 802.1Qbb. Priority based flow control, 2011.

[6] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Seattle, WA, USA, Aug. 2008*.

[7] Mohammad Alizadeh, Albert G. Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. Data center TCP (DCTCP). In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, New Delhi, India, Aug. 2010*.

[8] Infiniband Trade Association. Infiniband architecture volume 1, general specifications, release 1.2.1, 2008.

[9] Infiniband Trade Association. Supplement to InfiniBand architecture specification volume 1 release 1.2.2 annex A16: RDMA over converged ethernet (RoCE), 2010.

[10] Infiniband Trade Association. InfiniBand architecture volume 2, physical specifications, release 1.3, 2012.

[11] Infiniband Trade Association. Supplement to InfiniBand architecture specification volume 1 release 1.2.2 annex A17: RoCEv2 (IP routable RoCE), 2014.

[12] Brad Calder, Ju Wang, Aaron Ogus, Niranjan Nilakantan, Arild Skjolsvold, Sam McKelvie, Yikang Xu, Shashwat Srivastav, Jiesheng Wu, Huseyin Simitci, Jaidev Haridas, Chakravarthy Uddaraju, Hemal Khatri, Andrew Edwards, Vaman Bedekar, Shane Mainali, Rafay Abbasi, Arpit Agarwal, Mian Fahim ul Haq, Muhammad Ikram ul Haq, Deepali Bhardwaj, Sowmya Dayanand, Anitha Adusumilli, Marvin McNett, Sriram Sankaran, Kavitha Manivannan, and Leonidas Rigas. Windows azure storage: a highly available cloud storage service with strong consistency. In *Proceedings of the 23rd ACM Symposium on Operating Systems Principles, SOSP, Cascais, Portugal, Oct. 2011*.

[13] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In *6th Symposium on Operating System Design and Implementation (OSDI), San Francisco, California, USA, Dec. 2004*.

[14] Aleksandar Dragojevic, Dushyanth Narayanan, Miguel Castro, and Orion Hodson. Farm: Fast remote memory. In *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation, NSDI, Seattle, WA, USA, Apr. 2014*.

[15] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on networking*, (4):397–413, 1993.

[16] Sally Floyd, Jamshid Mahdavi, Matt Mathis, and Dr. Allyn Romanow. TCP Selective Acknowledgment Options. RFC 2018, October 1996.

[17] Yixiao Gao, Yuchen Yang, Tian Chen, Jiaqi Zheng, Bing Mao, and Guihai Chen. DCQCN+: Taming large-scale incast congestion in rdma over ethernet networks. In *IEEE 26th International Conference on Network Protocols, ICNP, Cambridge, UK, Sept. 2018*.

[18] Albert G. Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. VL2: a scalable and flexible data center network. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Barcelona, Spain, Aug. 2009*.

[19] Chuanxiong Guo, Haitao Wu, Zhong Deng, Gaurav Soni, Jianxi Ye, Jitendra Padhye, and Marina Lipshteyn. RDMA over commodity ethernet at scale. In *Proceedings of the ACM SIGCOMM Conference, Florianopolis, Brazil, Aug. 2016*.

[20] Chuanxiong Guo, Lihua Yuan, Dong Xiang, Yingnong Dang, Ray Huang, David A. Maltz, Zhaoyi Liu, Vin Wang, Bin Pang, Hua Chen, Zhi-Wei Lin, and Varugis Kurien. Pingmesh: A large-scale system for data center network latency measurement and analysis. In *Proceedings of the ACM SIGCOMM Conference on Special Interest Group on Data Communication, London, United Kingdom, Aug. 2015*.

[21] Shuihai Hu, Yibo Zhu, Peng Cheng, Chuanxiong Guo, Kun Tan, Jitendra Padhye, and Kai Chen. Deadlocks in datacenter networks: Why do they form, and how to avoid them. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks, HotNets 2016, Atlanta, GA, USA, Nov. 2016*.

[22] Anuj Kalia, Michael Kaminsky, and David G Andersen. Using RDMA efficiently for key-value services.

[23] Mark J. Karol, S. Jamaloddin Golestani, and David Lee. Prevention of deadlocks and livelocks in lossless, backpressured packet networks. In *IEEE Conference on Computer Communications, INFOCOM, Tel Aviv, Israel, Mar. 2000*.

[24] Yuanwei Lu, Guo Chen, Bojie Li, Kun Tan, Yongqiang Xiong, Peng Cheng, Jiansong Zhang, Enhong Chen, and Thomas Moscibroda. Multi-path transport for RDMA in datacenters. In *15th USENIX Symposium on Networked Systems Design and Implementation, NSDI, Renton, WA, USA, Apr. 2018*.

[25] Radhika Mittal, Vinh The Lam, Nandita Dukkipati, Emily R. Blem, Hassan M. G. Wassel, Monia Ghobadi, Amin Vahdat, Yaogong Wang, David Wetherall, and David Zats. TIMELY: RTT-based congestion control for the datacenter. In *Proceedings of the ACM Conference on Special Interest Group on Data Communication, SIGCOMM, London, United Kingdom, Aug. 2015*.

[26] Rong Pan, Balaji Prabhakar, and Ashvin Laxmikantha. QCN: Quantized congestion notification. *IEEE802*, 1, 2007.

[27] Yawen Pan, Chen Tian, Jiaqi Zheng, Gong Zhang, Hengky Susanto, Bo Bai, and Guihai Chen. Support ecn in multi-queue datacenter networks via per-port marking with selective blindness. In *38th IEEE International Conference on Distributed Computing Systems, ICDCS, Vienna, Austria, July. 2018*.

[28] K. K. Ramakrishnan, Sally Floyd, and David L. Black. The addition of explicit congestion notification (ECN) to ip. *RFC*, 3168:1–63, 2001.

[29] Danfeng Shan and Fengyuan Ren. Improving ecn marking scheme with micro-burst traffic in data center networks. In *IEEE Conference on Computer Communications, INFOCOM, Atlanta, GA, USA, May. 2017*.

[30] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Hong Liu, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Hölzle, Stephen Stuart, and Amin Vahdat. Jupiter rising: a decade of clos topologies and centralized control in google's datacenter network. In *Proceedings of the ACM Conference on Special Interest Group on Data Communication, SIGCOMM, London, United Kingdom, Aug. 2015*.

[31] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. Congestion control for large-scale rdma deployments. In *Proceedings of the ACM Conference on Special Interest Group on Data Communication, SIGCOMM, London, United Kingdom, Aug. 2015*.

[32] Yibo Zhu, Monia Ghobadi, Vishal Misra, and Jitendra Padhye. ECN or delay: Lessons learnt from analysis of DCQCN and TIMELY. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies, CoNEXT, Irvine, California, USA, Dec. 2016*.