Gao YX, Tian C, Chen W *et al.* Analyzing and optimizing packet corruption in RDMA network. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 37(4): 743–762 July 2022. DOI 10.1007/s11390-022-2123-8

Analyzing and Optimizing Packet Corruption in RDMA Network

Yi-Xiao Gao[1] (高翼枭), Chen Tian[1,*] (田　臣), *Senior Member*, *IEEE*, *Member*, *CCF*, *ACM*, Wei Chen[1] (陈　伟)
Duo-Xing Li[1] (李多星), Jian Yan[2] (闫　健), Yuan-Yuan Gong[1] (龚媛媛), Bing-Quan Wang[2] (王炳权)
Tao Wu[2] (吴　涛), Lei Han[2,*] (韩　磊), Fa-Zhi Qi[3] (齐法制), Shan Zeng[3] (曾　珊), Wan-Chun Dou[1] (窦万春)
and Gui-Hai Chen[1] (陈贵海)

[1] *State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210046, China*

[2] *Huawei Technologies Co. Ltd, Nanjing 210012, China*

[3] *Institute of High Energy Physics, Chinese Academy of Sciences, Beijing 100190, China*

E-mail: mrgaoxx@smail.nju.edu.cn; tianchen@nju.edu.cn; {chenw, ldx}@smail.nju.edu.cn
jim.yanjian@huawei.com; mf1933022@smail.nju.edu.cn
{wangbingquan4, wutao26, phoebe.han}@huawei.com; {qfz, zengshan}@ihep.ac.cn
{douwc, gchen}@nju.edu.cn

Received December 31, 2021; accepted July 5, 2022.

Abstract Remote direct memory access (RDMA) has become one of the state-of-the-art high-performance network technologies in datacenters. The reliable transport of RDMA is designed based on a lossless underlying network and cannot endure a high packet loss rate. However, except for switch buffer overflow, there is another kind of packet loss in the RDMA network, i.e., packet corruption, which has not been discussed in depth. The packet corruption incurs long application tail latency by causing timeout retransmissions. The challenges to solving packet corruption in the RDMA network include: 1) packet corruption is inevitable with any remedial mechanisms and 2) RDMA hardware is not programmable. This paper proposes some designs which can guarantee the expected tail latency of applications with the existence of packet corruption. The key idea is controlling the occurring probabilities of timeout events caused by packet corruption through transforming timeout retransmissions into out-of-order retransmissions. We build a probabilistic model to estimate the occurrence probabilities and real effects of the corruption patterns. We implement these two mechanisms with the help of programmable switches and the zero-byte message RDMA feature. We build an ns-3 simulation and implement optimization mechanisms on our testbed. The simulation and testbed experiments show that the optimizations can decrease the flow completion time by several orders of magnitudes with less than 3% bandwidth cost at different packet corruption rates.

Keywords datacenter network, packet corruption, programmable switch, remote direct memory access (RDMA)

1 Introduction

Remote memory direct access (RDMA) has become one of the state-of-the-art high-performance network technologies in datacenters [1,2]. It provides extremely low latency (about 3–5 μs) [3] and high throughput with low CPU consumption through its kernel-bypass and RDMA-capable network interface controller (RNIC) offloaded protocol stack. Multitudinous distributed applications and network middleware, such as key-value store [4–6], distributed transactions [7,8], distributed file systems [9,10], cloud storage [2,11,12] and Remote Procedure Call (RPC) systems [13–15], have exploited the performance of RDMA with novel software designs to obtain extremely low service latency.

The reliable transport of RDMA is designed based

Regular Paper
Special Section of Xia Peisu Young Scholars Forum 2021
This work was supported by the Key-Area Research and Development Program of Guangdong Province of China under Grant No. 2020B0101390001, the National Natural Science Foundation of China under Grant Nos. 61772265 and 62072228, the Fundamental Research Funds for the Central Universities of China, the Collaborative Innovation Center of Novel Software Technology and Industrialization of Jiangsu Province of China, and the Jiangsu Innovation and Entrepreneurship (Shuangchuang) Program of China.

*Corresponding Author (Lei Han is the project launcher and Chen Tian is the organizer of communication.)

on a lossless underlying network (i.e., without packet loss caused by switch buffer overflow) and cannot endure a high packet loss rate, e.g., 0.1% [1, 16]. RDMA adopts a go-back-to-N retransmission mechanism in which the transmission window falls back to the lost packet. In Ethernet/IP-based best-effort-delivery datacenter network architecture, RDMA is adapted to IP Routable RDMA over Converged Ethernet (RoCEv2). RoCEv2 works along with Priority-based Flow Control (PFC), a data link layer flow control protocol that prevents packet drop caused by buffer overflow.

However, except for switch buffer overflow, there is another kind of packet loss in the RDMA network, i.e., packet corruption [2, 17, 18], which has not been discussed in depth. According to Microsoft's research [17] on 350k links across 15 production datacenters, packet corruption shows universality in large datacenters. Compared with congestion loss, packet corruption has a rarer occurrence but a higher loss rate. Thus the occurrence of packet corruption causes a more severe system performance downgrade, such as extreme long-tail latency and request timeout [2].

The packet corruption incurs long application tail latency by causing timeout retransmissions in the RDMA network. The retransmission mechanisms can be classified as the out-of-order retransmission and the timeout retransmission, and there is a significant price gap between them [19]. The latency penalty of out-of-order retransmissions is an RTT, which is often less than 100 µs in the RDMA network. While the timeout value of retransmissions should be set to maximal possible network RTT (e.g., 64 ms by default), which is much bigger than the latency of most small flows and average network RTT. The timeout retransmission incurs high latency to small flows. There are four causes of timeout retransmissions in RDMA: NAK corruption, corruption of packet and its retransmission packet (packet corruption twice), the flow tail packet corruption, and the flow tail ACK corruption [20]. The first two types are attributed to the NAK-once mechanism in the RDMA protocol. The timeout caused by flow tail packet/ACK corruption also happens in other transport protocols such as TCP and QUIC [21].

There are two challenges to optimizing packet corruption in the RDMA network. 1) Packet corruption is inevitable with any remedial mechanisms. Since packet corruption is caused by hardware, it is nearly impossible to eliminate it by any remedial mechanisms such as redundancy by adding additional packets. 2) RNICs are not programmable. Unlike software transport protocol stacks like kernel TCP, RDMA transport protocol is offloaded to fixed-function hardware, i.e., RNICs. For deployment consideration, a handy solution should not modify existing RNICs and should be forward compatible with the standard RDMA protocol supported by existing RNICs.

This paper proposes some designs that can guarantee the expected tail latency of applications with packet corruption. The key idea is controlling the occurring probabilities of timeout events caused by packet corruption by transforming timeout retransmissions into out-of-order retransmissions. This transforming is achieved by inserting "dummy headers" (i.e., packets without payloads) at the end of each flow and repeating significant packets (i.e., flow tail ACKs, NACKs, and retransmitted packets) of which the corruption causes timeout. We implement these two mechanisms with the help of programmable switches and the zero-byte message feature provided by existing RNICs, which are deployable and compatible with existing RDMA hardware. We also build a probabilistic model to quantify the problem by estimating the occurrence probability and the effects of different corruption patterns and mechanisms.

We build an ns-3 simulation and implement optimization mechanisms on our testbed. The simulation and testbed experiments show that the optimizations can decrease the flow completion time by several orders of magnitudes with less than 3% bandwidth cost at different packet corruption rates. Besides, we also discuss some practical problems in deployment such as lossy RDMA compatibility and cooperation with corruption repair systems.

In summary, we make the following contributions.

• We build a probabilistic model that can quantify the occurrence probabilities of packet corruption in the RDMA network (Subsection 3.2) and give the probabilities of the four kinds of timeout retransmissions (Subsections 3.3–3.5) based on it.

• We design "repeat" and "dummy headers" optimizations to improve application tail latency for the packet corruption in the RDMA network and analyze them (Section 4).

• We introduce P4 programmable switches [22] to implement the "repeat" methodology (Subsection 5.1) and exploit zero-byte message RDMA feature to generate dummy headers (Subsection 5.2) without modifying existing RDMA protocol and RNICs.

2 Background

2.1 Reliable Protocol in RDMA

There are three kinds of RDMA transport services implemented on commodity RNICs: reliable connection (RC), unreliable connection (UC), and unreliable datagram (UD). Each type of transport services is provided by corresponding communication channels (i.e., queue pairs, QP). Users initiate data send/receive requests to RNICs by posting work queue elements (WQEs) into QPs. RDMA operations include one-sided (i.e., remote CPU bypassed) operations (e.g., RDMA_READ and RDMA_WRITE) and two-sided (i.e., remote-CPU involved) operations (e.g., RDMA_SEND). RC is the most commonly used transport type in RDMA since it provides reliable service [1, 2, 16]. In this paper, we only discuss RC. We use requesters and responders to refer to the client and the receiver of RDMA respectively, which is consistent with the naming in the RDMA protocol.

Fig.1 shows the receive state machine of the RC responder. The state machine is simplified for addressing packet order preserving. Thus it omits some upper-layer verification such as OpCode verification that does not influence our analysis. QPs maintain packet sequence number (PSN) and expected packet sequence number (ePSN) on the requester and the responder, respectively. The responder only accepts packets with PSN exactly matching ePSN ($PSN = ePSN$) and increases ePSN after successful receiving. If the responder detects a packet sequence error ($PSN > ePSN$), it generates a NAK that carries the current ePSN and sends the NAK to the requester. The responder only sends NAK once for each ePSN (even though more than one packet sequence error is detected). The NACK-once mechanism prohibits generating a series of NAKs by subsequent packets when one packet is lost. Our experiment on Mellanox ConnectX-6 RNICs shows that NAK is sent only once for each ePSN, which is the same as the protocol stipulates. When a requester receives a NAK, it restarts transmission from the packet with the ePSN carried in the NAK. The responder takes duplicated packets ($PSN < ePSN$) as valid packets and resends ACKs and responses.

The requester also uses a timer for timeout retransmission. If the requester does not receive an/a ACK/NAK for more than the retransmission time out (RTO) $= 4.096 \times 2^t$ µs, it restarts transmission from the first unacknowledged packet. Parameter t is set by users according to deployment scale and configuration. The timeout value should always be larger than the maximal queueing delay in the network to avoid spurious retransmission (i.e., retransmission triggered mistakenly when no packet is lost). For example, perftest sets the time timeout value to 267 ms, and Alibaba's large-scale storage RDMA network [2] endures a four-second reconnection timeout (which equals retransmission timeout multiplying retry times).

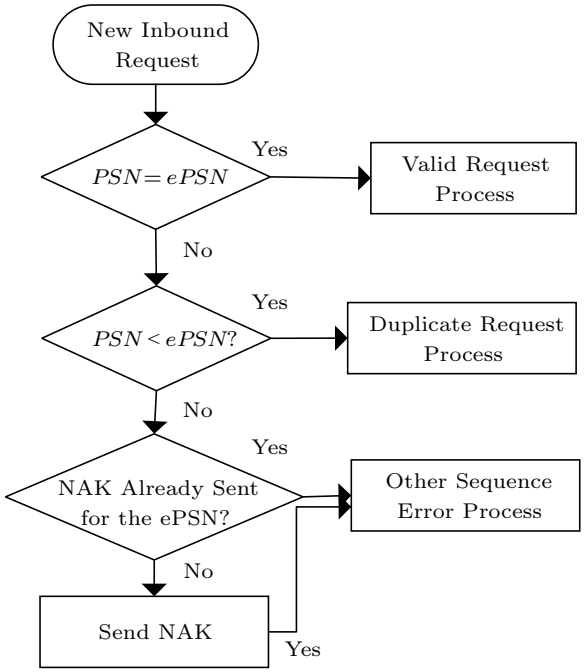

Fig.1. Receive state machine of RDMA RC responder.

2.2 Packet Corruption in Datacenter

Zhuo *et al.* studied packet corruption based on statistics on 350k links across 15 Microsoft's production datacenters [17]. These links are optical fibers between switches. The links between hosts and switches are not considered since their physical medium can be copper lines, which have different physical features from optical fibers. Here we conclude some crucial conclusions for packet corruption from [17].

Firstly, packet corruption affects fewer links than congestion but imposes a higher loss rate. Zhuo *et al.* computed the percentage of links with a congestion/corruption loss rate above 10^{-8}, and found that the total number of links with corruption is less than 2%–4% of those with congestion [17]. Among these links, 52.8%/12.7% of them have more than $10^{-5}/10^{-3}$ packet loss rate, while the ratio for congested links is 7.6%/0.2%.

The packet corruption rate on a link is stable over

time and is not correlated with link utilization. To the contrast, congestion loss has positive relevance to link utilization. Corruption also exhibits weak locality, i.e., the chances of multiple corrupting links being on the same switch or being topologically close are noticeable but low. Finally, corruption is also asymmetric and uncorrelated with link location.

Table 1 [17] shows the root causes of packet corruption and their distributions in Microsoft's datacenter.

Table 1. Root Causes of Packet Corruption and Their Distributions [17]

Root Causes of Packet Corruption	Proportion
Connector contamination	17%–57%
Damaged or bent fibers	14%–48%
Decaying transmitters	Less than 1%
Bad or loose transceivers	6%–45%
Shared-component failure	10%–26%

3 Packet Corruption Model

3.1 Interaction of Packet Corruption and RDMA Protocol

Interacting with the RDMA protocol, the packet corruption of specific packet types (NAK packets), packet positions (the last packets of flows), and packet features (packets with the same PSN) is more detrimental to application performance. The cause is the price gap between out-of-order retransmission and timeout retransmission in the RDMA transport protocol.

The latency penalty of out-of-order retransmission is relatively tiny. When a packet is corrupted and dropped, the responder detects a packet sequence error and sends a NAK to the requester. The requester then retransmits the packets from the NAK PSN. Due to this packet corruption, an additional RTT is added to the flow latency, and the links suffer bandwidth loss since the inflight packets after the corrupted packet are invalidated and dropped by the responder. Some common optimizations in RDMA systems, such as limiting inflight request size [13, 15], can reduce this bandwidth loss.

However, the requester must wait for the timeout to retransmit all inflight packets if the out-of-order retransmission cannot be triggered. Since the timeout value should be set to maximal network RTT (e.g., 64 ms by default), which is much bigger than the latency of most small flows and average network RTT, timeout retransmission incurs extremely high latency to flows. The latencies of today's datacenter RPCs are often less than 10 µs [2, 15, 23, 24], which suffer more than 1 000x tail latency if a timeout retransmission is triggered.

Since timeout retransmissions are only triggered in the absence of ACKs/NAKs, we can analyze the timeout retransmission according to the state of ACKs/NAKs (corrupted or not generated), and can conclude that four kinds of packet corruption can cause timeout retransmission: NAK corruption, packet corruption twice, flow tail packet corruption, and flow tail ACK corruption [20].

• *NAK Corruption.* Each NAK corruption causes a timeout retransmission. Since the responder only sends NAKs once for each ePSN (Subsection 2.1), when a packet and its corresponding NAK are both corrupted, the following packets will be silently dropped by the receiver due to the disorder, and no response will be sent to the requester. As a result, the requester cannot sense the loss and continues transmitting until timeout. Fig.2 demonstrates the case.

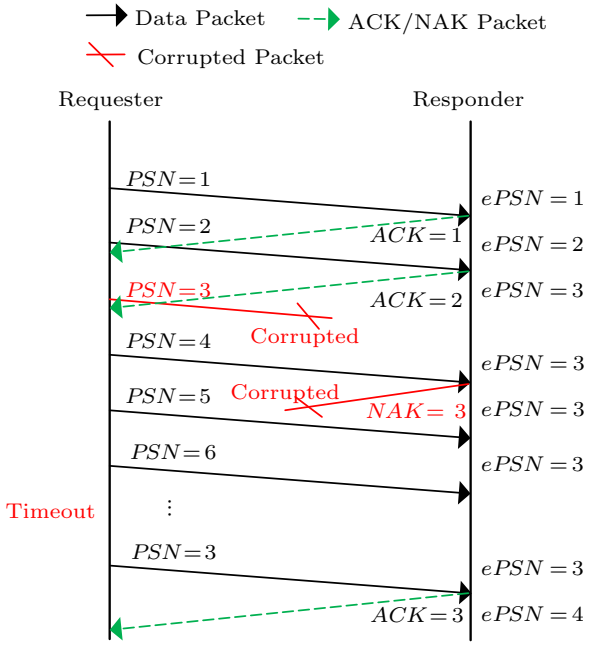

Fig.2. Requester timeout caused by NAK corruption.

• *Packet Corruption Twice.* The case is also related to the only-once NAK mechanism in RDMA. Suppose a packet and its retransmission packet with the same PSN are both corrupted. In that case, the requester also launches a timeout retransmission since the responder will not generate a NAK again for this ePSN. Fig.3 demonstrates the case.

• *Flow-Tail Packet Corruption.* Once the flow tail packet (the last packet of a flow) is corrupted and dropped, the requester must wait for a timeout since

no more packet can trigger out-of-order retransmission. Fig.4 demonstrates the case.

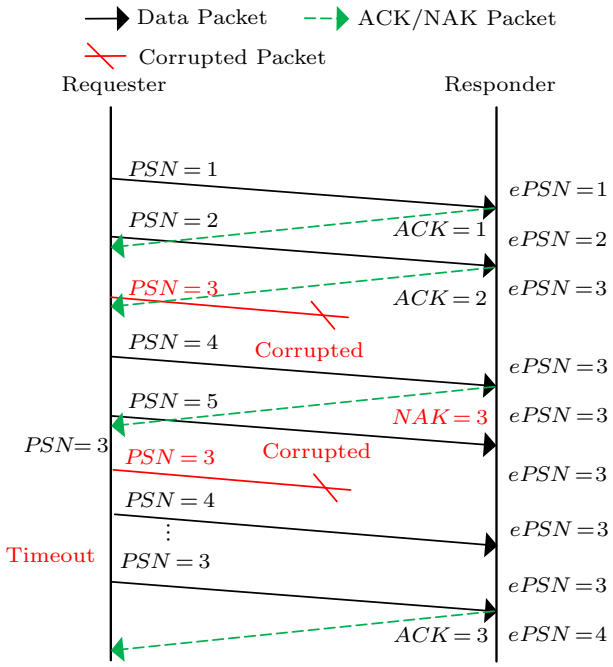

Fig.3. Requester timeout caused by packet corruption twice.

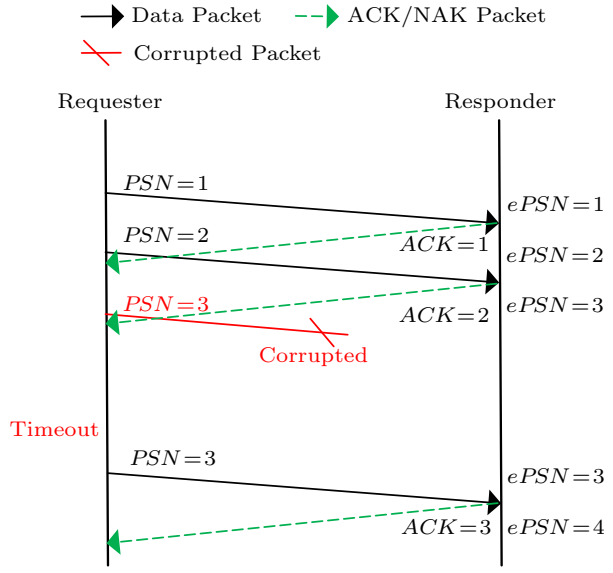

Fig.4. Requester timeout caused by flow tail packet corruption.

Compared with NAK corruption and packet corruption twice, the consequence of flow tail packet corruption is more complex. The RDMA connections (QPs) are usually multiplexed among threads or applications to decrease the number of QPs per RNIC, which avoids cache miss in RNIC RAM [2, 25]. In such scenarios, flows are aggregated in QPs, thus the "flow" in "flow tail packet corruption" should be defined as batches of RDMA requests transmitted on the same QP with a time gap so that there is no more packet on the same QP to be transmitted immediately after each flow tail packet. Once a flow tail packet is dropped, the timeout may not be triggered before the arrival of next flow in this QP. In such cases, the latency penalty of flow tail packet corruption is the inter-flow gap rather than the RTO. The inter-flow gap time is determined by the traffic load on hosts (i.e., host link utilization) and the number of QPs on each host. With a lower host link utilization and a larger number of QPs per RNIC, the latency penalty of flow tail packet corruption is more severe.

• *Flow Tail ACK Corruption.* Similar to flow tail packet corruption, once the ACK for the flow tail packet is corrupted and dropped, the requester must wait for a timeout since no more packet can trigger subsequent ACKs. Fig.5 demonstrates the case.

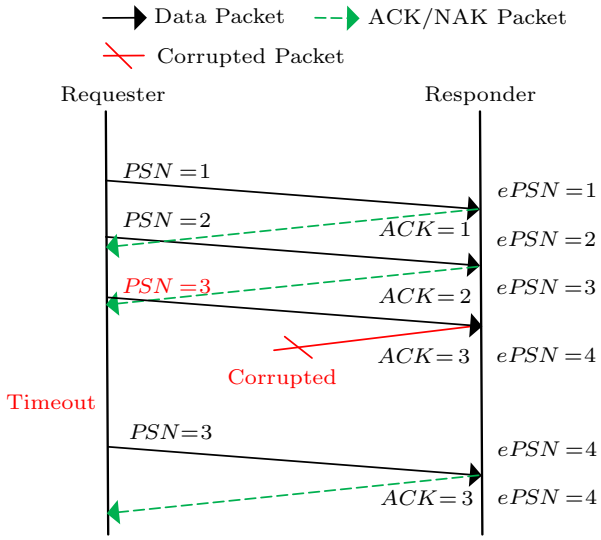

Fig.5. Requester timeout caused by flow tail ACK corruption.

3.2 Packet Corruption Model

Since packet corruption is inevitable but probabilistic, the actual damage of different kinds of packet corruption to system performance should also be evaluated according to their occurrence probability. Thus we build a mathematical model to estimate the probability of different kinds of packet corruption. Table 2 shows the symbols we use in the following theorems and poofs.

The probability of link corruption in large-scale datacenter networks is not revealed in previous work [2, 17] and might vary with different lives of datacenters. How-

ever, the statistical corruption rate of corrupted links can be analyzed. As we discussed in Subsection 2.2, the causes of corruption are mostly physical (e.g., connector contamination and damaged/bent fiber) [17] and are not related with the location of each bit. Thus, as a simple but effective estimation, it is reasonable to assume the corruption events of each single bit (b) on a corrupted link are independent identically distributed (i.i.d.) and obey Bernoulli distribution with probability α:

$$P(C_b) = \alpha.$$

Table 2. Mathematic Symbols Used in This Paper

Symbol	Meaning
b, h, p	Bit, header, packet respectively
l, w	Link and path respectively
X, Y	Aliases of $b/h/p$ and l/w
C_X	Event of X being corrupted
$C_X(Y)$	Event of X ($b/h/p$) being corrupted on Y (l/w)
S_X	Size of X (in bytes)
$R(S)$	Value range of S (in bytes)
T_p	Event of packet p being the tail packet of a flow
ω	Link corruption probability
α	Bit corruption probability on a corrupted link
β	Packet corruption probability on a corrupted link
γ	Header corruption probability on a corrupted link
δ	ACK/NAK corruption probability on a corrupted link
P_A	Probability of timeout with corruption type A
P_A^B	Probability of timeout with corruption type A with optimization B
B_X^Y	Bandwidth of X timeout with optimization Y
$\overline{C}$	Complement of event C
$P(A\|B)$	Probability of event A with the occurrence of B

According to the assumption of single bit corruption probability distribution, we can obtain the probability β of single packet corruption.

Theorem 1. *The bit corruption probability β of single packet of size S_p on a corrupted link is $1-(1-\alpha)^{8S_p}$.*

Proof. Since a packet p contains bit b_i ($i = 1, ..., 8S_p$), we have

$$\begin{aligned}\beta &= P(C_p) = 1 - P(\overline{C_p}) \\ &= 1 - P(\bigcup_{i=1}^{8S_p} \overline{C_{b_i}}) \\ &= 1 - \prod_{i=1}^{8S_p}(1 - P(C_b)) \\ &= 1 - (1-\alpha)^{8S_p}.\end{aligned}$$

□

Since the packet size S_p is upper-bounded with maximum transmission unit (MTU) $(S_p)_{\max}$, the probability of single packet corruption β is also upper-bounded by:

$$\beta \leqslant 1 - (1-\alpha)^{8(S_p)_{\max}} = \beta_{\max}.$$

According to relevant researches [17, 18], the typical value range of β in a datacenter is $[10^{-8}, 10^{-2}]$. Since the majority of the packets on wire are data packets with MTU ($S_p \approx (S_p)_{\max}$), we assume this value range is also approximately applied to $\beta_{\max}$, i.e., $\beta \approx \beta_{\max}$ for all subsequent analyses.

The size of the RoCEv2 packet headers (including Ethernet CRC field) S_h is 62 bytes (18-byte Ethernet header and CRC footer, 20-byte IP header, 8-byte UDP header, 12-byte Base Transport Header, and 4-byte ICRC header). In order to simplify the calculation, we assume that all corruption of the headers can be checked out by validating the checksum. The size of a full data RoCEv2 packet with a 1 024-byte MTU is 1 086 bytes. In some datacenters, jumbo frame of larger MTU (e.g., 4 KB or 8 KB) might be used [15, 26], and all the conclusions should be simply recalculated with the same equations. Given these sizes, we have the approximation.

Theorem 2. *The corruption probability of the packet header (γ) of full-data RoCEv2 packets approximates $0.06\beta_{\max}$.*

Proof. Note that $S^h = 8 \times 62$, $(S_p)_{\max} = 8 \times 1\,086$ and applying the formula $(1-x)^\alpha = 1 - \alpha x + O(x^2)$, we have:

$$\begin{aligned}\gamma &= 1 - (1-\alpha)^{8S_h} \\ &= 1 - (1-\beta_{\max})^{\frac{S_h}{(S_p)_{\max}}} \\ &= 1 - (1-\beta_{\max})^{\frac{62}{1\,086}} \\ &\approx 0.06\beta_{\max} + O(\beta_{\max}^2).\end{aligned}$$

□

Thus in our bit corruption error model, the probability of header corruption (γ) is much smaller than the probability of packet corruption (β).

We can also estimate the corruption probability of ACK/NAK (δ) with size $S_{\text{NAK}} = 68$ on a corrupted link:

$$\begin{aligned}\delta &= P(C_{\text{NAK}}) \\ &= 1 - (1-\alpha)^{8S_{\text{NAK}}} \\ &= 1 - (1-\beta_{\max})^{\frac{S_{\text{NAK}}}{(S_p)_{\max}}} \\ &\approx 0.06\beta \approx \gamma.\end{aligned}$$

For packets going through paths with more than one link, we have to use the following theorem for analysis.

Theorem 3. *For flows going through an L-link path with the corruption rate of each link C_l =*

ω, the probability of packet/header/NAK corruption ($P(C_p)/P(C_h)/P(C_{\text{NAK}})$) approximates $L\beta\omega/L\gamma\omega/L\delta\omega$.

Proof. We only prove the probability of packet corruption since the proofs of the other cases are similar. It is assumed that the paths of packets are $\{l_i|i=1,...,L\}$. For each link, the probability of packet $P(C_p(l_i))$ is $\omega\beta$. The corruption probability of each packet going through the path is

$$\begin{aligned} P(C_p) &= 1 - P(\overline{C_p}) \\ &= 1 - \prod_{i=1}^{L} P(\overline{C_p(l_i)}) \\ &= 1 - (1-\beta\omega)^L \\ &\approx L\beta\omega. \end{aligned}$$

□

3.3 NAK Corruption

As we referred in Subsection 3.1, each NAK corruption can incur a timeout event on the requester. The NAK corruption only occurs when the packet and its NAK are both corrupted. Note that due to the asymmetry of corruption [17], the corruption of the links that the data and NAKs go through should be calculated independently. Thus the probability of NAK corruption timeout can be calculated as follows.

Theorem 4. *For a flow going through an L-link path, the probability of NAK corruption timeout on each link (P_{NAK}) approximates $0.06L^2\beta_{\max}^2\omega^2$.*

Proof. Note that since the packet and its NAK go through different paths, the event NAK corruption (C_{NAK}) and the corruption of packet (C_p) are independent; thus we have:

$$\begin{aligned} P_{\text{NAK}} &= P(C_p C_{\text{NAK}}) \\ &= P(C_p)P(C_{\text{NAK}}) \\ &= L\beta\omega \times L\delta\omega \\ &\approx 0.06L^2\beta_{\max}^2\omega^2. \end{aligned}$$

□

The value range of the NAK corruption timeout probability (P_{NAK}) is much smaller than the probability of packet corruption ($P(C_p)$) since the NAK corruption requires bidirectional link corruption and the size NAK is much smaller than the packet size in some cases. On a 2-link path ($L=2$), the NAK corruption timeout probability is up to $2.4\times10^{-5}\omega^2$ (when $\beta=0.01$).

3.4 Packet Corruption Twice

The concurrent corruption of a packet and its retransmission also incurs a timeout event. According to our model, the corruption of each bit signal on wire is independent; thus obviously the corruption of a packet and its retransmission are also independent. Then the probability of packet corruption timeout twice (P_{twice}) can be calculated.

Theorem 5. *For flows going through an L-link path W with the corruption rate of each link $C_l=\omega$, the probability of packet corruption timeout twice (P_{twice}) approximates $L\beta_{\max}^2\omega$.*

Proof. The packet corruption twice timeout only occurs when a packet p and its retransmission p' are corrupted; thus we have

$$\begin{aligned} P_{\text{twice}} &= P(C_{p'}C_p) \\ &= P(C_{p'}|C_p)P(C_p) \\ &= L\beta\omega P(C_{p'}|C_p). \end{aligned}$$

Note that the corruption of retransmission of p' is only related to the path corruption rate; thus its probability is the same with the corruption of p', and we have

$$\begin{aligned} P(C_{p'}|\ C_p) &= P(C_{p'}|\ C_L) \\ &= P(C_p|\ C_L) \\ &= \frac{P(C_p)}{P(C_L)} \\ &= \frac{L\beta\omega}{1-(1-\omega)^L} \\ &\approx \frac{L\beta\omega}{1-(1-L\omega)} = \beta. \end{aligned}$$

Thus we have

$$\begin{aligned} P_{\text{twice}} &= L\beta\omega P(C_{p'}|C_p) \\ &\approx L\beta^2\omega \approx L\beta_{\max}^2\omega. \end{aligned}$$

□

The value range of probability P_{twice} is $[10^{-16},10^{-4}]L\omega$, i.e., P_{twice} is $[10^{-16},10^{-4}]$ per corrupted link. P_{twice} is much larger than P_{NAK} ($P_{\text{twice}}\approx\frac{16.5}{L\omega}P_{\text{NAK}}$).

3.5 Flow Tail Corruption

The probability of flow tail packet corruption (P_{tail}) is affected not only by the bit error rate but also by the flow size distribution. This corruption event can be explained as "the corrupted packet is one of the flow tail" and calculated.

Theorem 6. *The probability of flow tail packet corruption on each link (P_{tail}) approximates $\mathbb{E}(\frac{1}{S})\beta_{\max}\omega$, where S is the random variable of the flow size (counted in packets).*

Proof. Let T_p denotes the event that packet p is the tail packet of a flow. For any packet in a flow of N packets, $P(T_p)$ is $1/N$. Thus $P(T_p)$ for packet p from any flow with a random flow size S is:

$$\begin{aligned} P(T_p) &= \sum_{S_i \in R(S)} P(T_p|S=S_i)P(S=S_i) \\ &= \mathbb{E}\left(\frac{1}{S}\right). \end{aligned}$$

Here $R(S)$ is the value range of S. Thus P_{tail} can be calculated as:

$$\begin{aligned} P_{\text{tail}} &= P(T_p|C_p)P(C_p) \\ &= \mathbb{E}\left(\frac{1}{S}\right)\beta\omega \approx \mathbb{E}\left(\frac{1}{S}\right)\beta_{\max}\omega. \end{aligned}$$ □

For the flow distribution with a larger average flow size, the mathematical expectation $\mathbb{E}(\frac{1}{S})$ is smaller. For the WebSearch workload, $\mathbb{E}(\frac{1}{S})$ is 0.055. Under such flow distribution, P_{tail} is larger than the probability of the other two kinds of corruption ($P_{\text{tail}} \approx \frac{0.055}{\beta_{\max}} P_{\text{twice}} \approx \frac{0.91}{\beta_{\max} L\omega} P_{\text{NAK}}$). However, as we discussed in Subsection 3.1, although flow tail corruption occurs with a higher probability, the penalty flow latency of this kind of corruption is also affected by the inter-flow gap time.

Similarly, for the flow tail ACK corruption, we have Theorem 7.

Theorem 7. *The probability of the flow tail ACK packet corruption on each link* ($P_{\text{tail\_ACK}}$) *approximates* $0.06\mathbb{E}(\frac{1}{S})\beta_{\max}\omega$*, where* S *is the random variable of flow size* (*counted in packets*).

Proof. The proof is similar to Theorem 6. □

The corruption probability of the flow tail ACK is smaller than the flow tail packet corruption probability ($P_{\text{tail\_ACK}} \approx 0.06 P_{\text{tail}}$).

4 Optimizations for Different Scenarios of Packet Corruption

In this section, we show our designs on optimizing different scenarios of packet corruption. The key idea is converting timeout retransmissions into out-of-sequence retransmissions. For low-probability events such as NAK and duplicated packet corruption, we adopt "repeat" as our basic methodology. This simple but effective technique can decrease the possibility of timeout exponentially. For high-probability flow tail packets transmission, we generate "dummy" headers after each tail packet. The solution is low-overhead and is also handy in software with some RDMA programming techniques. For all the methods, we analyze their protocol interaction, timeout probability reduction, and software/hardware overhead.

4.1 Repeating Corrupted NAK

A simple but effective technique for preventing corruption is adding redundancy by repeating. For each NAK packet, generating a replication can effectively decrease the probability of NAK corruption timeout since the timeout will not be triggered as one replication arrives at the requester. Fig.6 demonstrates an example of repeating NAK on the top-of-rack (ToR) switch to avoid the NAK corruption timeout.

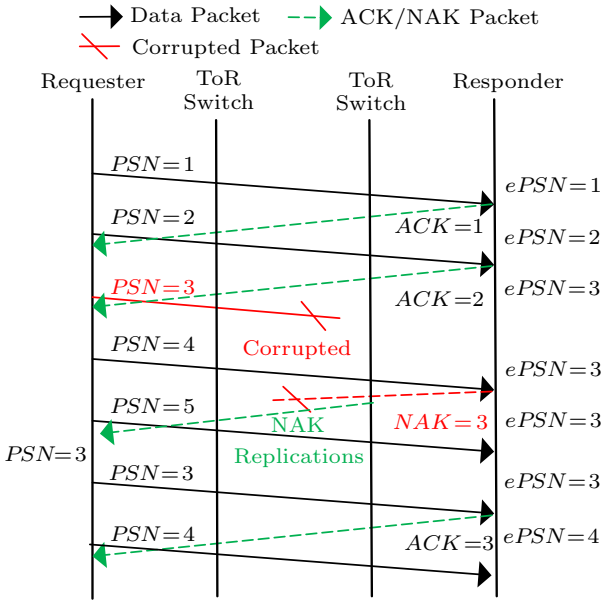

Fig.6. Optimizing NAK corruption timeout with repeating on ToR switches.

The probability of NAK corruption $P_{\text{NAK}}^{\text{repeat}}$ can be further decreased by adding the number of replicated NAK packets, and can be calculated as follows.

Theorem 8. *For flows going through an* L*-link path* W *with the corruption rate of each link* $C_l = \omega$*, the probability of NAK corruption* $P_{\text{NAK}}^{\text{repeat}}$ *with* $(N-1)$ *replicated NAK packets approximates* $0.06^N L^2 \beta_{\max}^{N+1} \omega^2$.

Proof. Since the corruption of the replicated NAK is only related to the link state, we have

$$\begin{aligned} & P(C(NAK')|C(NAK)) \\ =\ & P(C(NAK')|C(L_{\text{NAK}})) \\ =\ & P(C(NAK'))/C(L_{\text{NAK}}) \\ =\ & \frac{L\delta\omega}{1-(1-\omega)^L} \\ \approx\ & \frac{L\delta\omega}{1-(1-L\omega)} = \delta. \end{aligned}$$

Thus

$$\begin{aligned} P_{\text{NAK}}^{\text{repeat}} &= P(C_p C_{NAK_1} ... C_{NAK_N}) \\ &= P(C_p) P(C_{\text{NAK}}) P(C_{\text{NAK}'} | C_{\text{NAK}})^{N-1} \\ &= L\beta\omega \times L\delta\omega \times \delta^{N-1} \\ &\approx 0.06^N L^2 \beta_{\max}^{N+1} \omega^2. \end{aligned}$$ □

On a 2-link path ($L = 2$), using one replication ($N = 2$) can decrease the probability to a very low rate even with a high corruption rate (less than $1.4 \times 10^{-8}\omega^2$ when $\beta_{\max} \leqslant 10^{-2}$).

In this design, the requester does not sense the NAK replication and treats the NAK replications as different NAKs when processing them. The influence of duplicated PSN fall-back is small if all the NAK replicas arrive at the requester closely. Since the RDMA protocol stack, including the function of generating NAK, is offloaded to hardware RNICs, the replication can only be generated by RNICs or inside network hardware, e.g., by programmable switches. The hardware packet replications are low-latency, i.e., without adding large time gap to the packet and its replication. The arrival of the replicas is very close since they pass through the same path after being generated (RDMA does not support packet-level multipath [19]).

The major overhead of the solution is generating additional replications for each NAK. The average additional bandwidth consumption B_{NAK} on a corrupted link can be calculated by counting the expectation of additional bytes transmitted for each packet:

$$\begin{aligned} \mathbb{E}(B_{\text{NAK}}^{\text{repeat}}) &= \frac{S_{\text{NAK}}(N-1)\beta}{S_p} \\ &< 0.000\,6\beta_{\max}(N-1). \end{aligned} \tag{1}$$

The additional bandwidth overhead is small even on a corrupted link. For the normal links, there is no additional bandwidth overhead. The implementation of NAK replication is discussed in Section 5.

4.2 Repeating Retransmission of Corrupted Packets

Repeating can be also applied for retransmitted packets to avoid timeout caused by packet corruption twice. If the requester sends $(N-1)$ replications of the retransmitted packet with ePSN, the timeout retransmission will only be triggered when all the replications are dropped. Fig.7 demonstrates an example of the repeating the retransmission of the corrupted packet on the ToR switch.

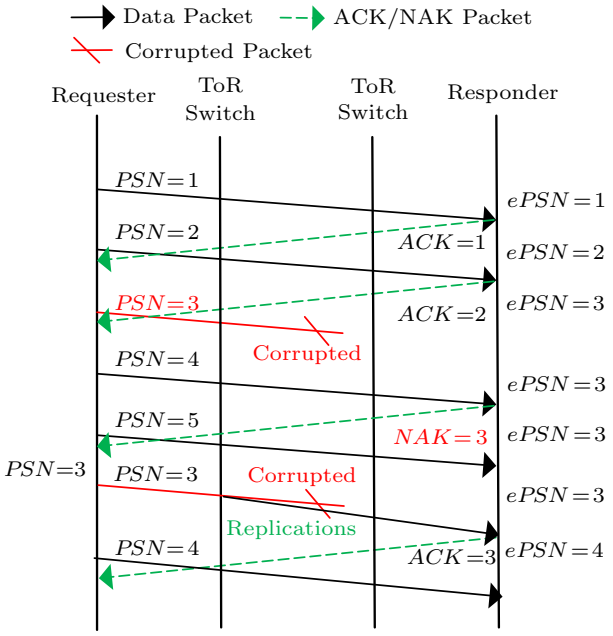

Fig.7. Optimizing packet twice corruption timeout with repeating.

The probability of timeout with repeating the retransmission of the corrupted packet can be calculated as follows.

Theorem 8. *For flows going through an L-link path W with the corruption rate of each link $C_l = \omega$, the probability of packet corruption twice timeout ($P_{\text{twice}}^{\text{repeat}}$) with N repeated retransmissions approximates $L\beta_{\max}^{N+1}\omega$.*

Proof.

$$\begin{aligned} P_{\text{twice}}^{\text{repeat}} &= P(C_p C_{p'_1} ... C_{p'_n}) \\ &= P(C_p) P(C_{p'} | C_p)^N \\ &= L\beta^{N+1}\omega \approx L\beta_{\max}^{N+1}\omega. \end{aligned}$$ □

$P_{\text{twice}}^{\text{repeat}} \approx L\beta_{\max}^{N+1}\omega$ is much larger than $P_{\text{NAK}}^{\text{repeat}} \approx 0.06^N L^2 \beta_{\max}^{N+1}\omega^2$ with the same number of replications. For example, to ensure $P_{\text{twice}}^{*} < 2 \times 10^{-8}\omega$ with $L = 2$, $\beta_{\max} = 10^{-2}$, we should replicate the original retransmission for two more times ($N = 3$).

The overhead of generating additional NAKed packets is larger than that of generating NAKs. Similar to (1), we can calculate additional bandwidth overhead on a corrupted link for replicating NAKed packets $B_{\text{twice}}^{\text{repeat}}$:

$$\mathbb{E}(B_{\text{twice}}^{\text{repeat}}) = (N-1)\beta < 0.01(N-1).$$

The order of magnitudes of additional bandwidth consumption is determined by the packet corruption rate β, while it is commonly less than 5% (e.g., $N = 6$, $\beta = 0.01$) on a corrupted link. Since we only replicate

the packet of which $PSN = ePSN$ (i.e., the first retransmitted packet after NAKed), the bandwidth overhead should be much smaller than the retransmission bandwidth in most cases.

4.3 Attaching "Dummy Headers" at Flow Tail

To handle with the loss of the flow tail packet, the method used in QUIC [21] is repeating the last packet or setting a separate RTO merely for this packet. Fig.8 demonstrates the strategy of repeating the flow tail packet.

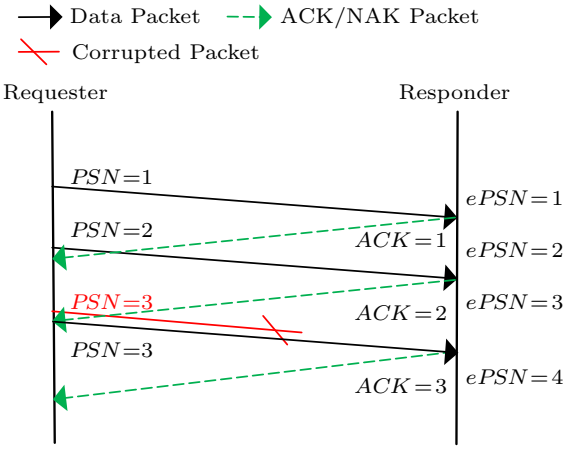

Fig.8. Optimizing flow tail packet corruption timeout with repeating.

However, the overhead of repeating is considerable when flow sizes are mostly small. While, in some popular RDMA applications such as key-value stores, the flow size is usually very small [13, 15]. For example, in Alibaba's RDMA-based Elastic Block Storage service, 50% of the total data is contributed by flows with the size less than 8 KB [27].

Instead of simple packet replication, our methodology is converting the timeout retransmission into out-of-order retransmission. For each flow tail packet, we add some sentinel packets just after each flow tail packet. They are named "dummy headers" since they only contain necessary RDMA headers for transport without data payload. The dummy packets are still allocated with PSNs just like other data packets, but their sizes are small and do not contribute to flow completion. The risk of timeout is transferred from the flow tail packet to the dummy headers. Once the flow tail packet is corrupted, the responder can sense its loss and generate a NAK through receiving the subsequent dummy headers with a larger PSN. Besides, the timeout retransmission of the dummy headers does not affect the completion of the flow. Fig.9 demonstrates how the dummy headers work with tail packet corruption.

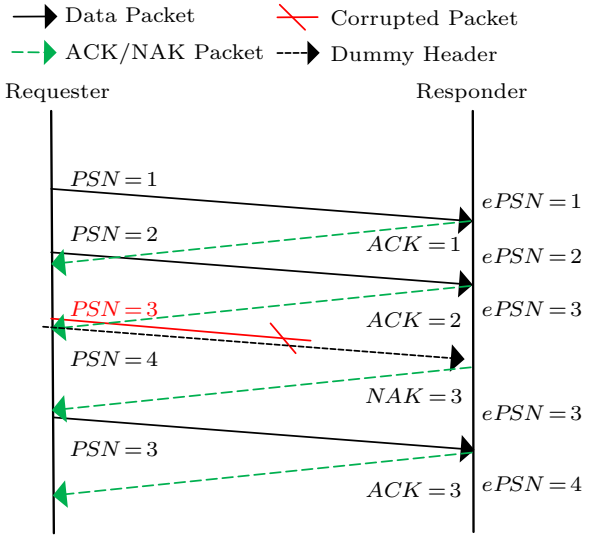

Fig. 9. Optimizing flow tail packet corruption timeout with dummy headers.

For the dummy headers, we have the following theorem.

Theorem 10. *For flows going through an L-link path W with the corruption rate of each link $C_l = \omega$, the probability of flow tail corruption with N dummy headers $P_{\text{tail}}^{\text{dummy}}$ approximates $0.06^N \mathbb{E}(\frac{1}{S}) L \beta_{\max}^{N+1} \omega$.*

Proof. Similar to the proof in Theorem 5, we have the approximation $P(C_h|C_p) \approx \gamma$; thus

$$\begin{aligned} P_{\text{tail}}^{\text{dummy}} &= P(T_p C_p C_{h_1} ... C_{h_N}) \\ &= P(T_p|C_p) P(C_p) P(C_h|C_p)^N \\ &= L\beta\omega \times \mathbb{E}\left(\frac{1}{S}\right) \gamma^N \\ &\approx 0.06^N \mathbb{E}\left(\frac{1}{S}\right) L \beta_{\max}^{N+1} \omega. \end{aligned}$$

□

For flows on a two-link path ($L = 2$) that obey the WebSearch flow distribution ($\mathbb{E}(\frac{1}{S}) = 0.055$), using only one dummy header ($N = 1$) can decrease $P_{\text{tail}}^{\text{dummy}}$ to $6 \times 10^{-7}\omega$.

In contrast, the effectiveness of repeating the flow tail packet is as follows.

Theorem 11. *For flows going through an L-link path W with the corruption rate of each link $C_l = \omega$, the probability of flow tail corruption with N replicated flow tail packets $P_{\text{tail}}^{\text{repeat}}$ approximates $\mathbb{E}(\frac{1}{S}) L \beta_{\max}^{N+1} \omega$.*

Proof. The proof is similar to Theorem 10. □

The flow timeout probability using "repeat" is much higher than that using "dummy headers" with the same

number of replications ($P_{\text{tail}}^{\text{dummy}} = 0.06^{N+1} P_{\text{tail}}^{\text{repeat}}$ with N replications).

Dummy headers and repeating are also effective for optimizing flow tail ACK corruption. The theoretical result is similar.

Theorem 12. *For flows going through an L-link path W with the corruption rate of each link $C_l = \omega$, the probability of flow tail ACK corruption with N dummy headers $P_{\text{tail\_ACK}}^{\text{dummy}}$ and the probability of flow tail ACK corruption with N replicated flow tail packets $P_{\text{tail\_ACK}}^{\text{repeat}}$ both approximate $0.06^{N+1}\mathbb{E}(\frac{1}{S})L\beta_{\max}^{N+1}\omega$.*

Proof. The proof is similar to Theorem 10. □

The bandwidth overhead of dummy headers is also much smaller. The bandwidth overhead of N dummy headers $B_{\text{tail}}^{\text{dummy}}$ is

$$\mathbb{E}(B_{\text{tail}}^{\text{dummy}}) = \mathbb{E}(\frac{1}{S}) \times \frac{62N}{1\,086} \approx 0.06N\mathbb{E}(\frac{1}{S}).$$

While the bandwidth overhead of repeating B_{repeat} is

$$\mathbb{E}(B_{\text{tail}}^{\text{repeat}}) = \mathbb{E}(\tfrac{1}{S}) \times N \approx 16\mathbb{E}(B_{\text{tail}}^{\text{dummy}}).$$

Since the overhead $\mathbb{E}(B_{\text{tail}}^{\text{repeat}})$ or $\mathbb{E}(B_{\text{tail}}^{\text{dummy}})$ is added to each path rather than corrupted paths ($\mathbb{E}(B_{\text{NAK}})$ and $\mathbb{E}(B_{\text{twice}})$ only exist on corrupted paths), the overhead of flow tail timeout optimization is much bigger than that of the other two. In the worst case, when the flows are all consisted of one packet ($\mathbb{E}(\frac{1}{S}) = 1$), $\mathbb{E}(B_{\text{repeat}})$ is 100% of original throughput while $\mathbb{E}(B_{\text{dummy}})$ is only 6% with one dummy header.

5 Implementation

Since the RDMA protocol is commonly offloaded to non-programmable RNICs, how to implement the strategies in practice should also be considered. One option is to implement these strategies on new generations of RNICs, which is a clean-slate solution but cannot benefit existing RDMA clusters. Instead, we involve in-network programmable switches to tune the data plane of RDMA. P4 [22], as one of the most popular programmable switch models, has risen wide interests [28–30], and the products (e.g., Intel Tofino and Tofino2) have already been in volume production and deployed in some datacenters.

5.1 Repeating NAK and Corrupted Packets with P4

We implement a prototype with the optimization of NAK/retransmission repeating referred in Section 4 in P4 and deploy it on the Barefoot Tofino programmable switch. The function of the packet replication can be implemented with the "Mirror" function in P4. NAKs are replicated on the ToR switches of the responders and packet retransmissions on the ToR switches of the requesters. For replication, NAK packets can be easily distinguished by parsing the RDMA AETH headers and verifying the Opcode field.

However, the retransmitted packets cannot be distinguished according to their packet headers. Instead, we store the ePSN (i.e., the PSN of the corrupted packet) carried in NAKs on the ToR switch of the requester and replicate the first packet with this PSN. The ePSN is deleted after the replication. Since the retransmitted packets are bound to following corresponding NAK packets, this design will replicate retransmitted packets of each NAK packet. Since all RDMA packets (including NAKs) only contains their destination QP numbers, we cannot associate the packets and NAKs of the same QP; thus we do not check the QP number during the replication. The replication might be wrong when a packet with the same PSN is transmitted during this time. However, since the value space of the packet sequence number is very large (2^{24}), the probability of the PSN collision is very small. Besides, a spurious replication does not incur any fatal consequences except for the failure of avoiding timeout.

5.2 Inserting Dummy Headers with RDMA Zero-Byte Message

Adding dummy headers is more easy to implement on requesters rather than inside network. The requester RNIC generates and sends several empty packets if there is no more data to send in the connection (QP). On the responder side, the empty packets should be silently processed without generating completion or passing anything to applications. Acknowledgment and retransmission for the packets should still proceed for the compatibility to the current PSN/acknowledgement mechanism on RNICs.

We can implement an effective simplification of dummy headers without modifying the RDMA hardware through the "zero-byte message" feature. A requester is allowed to post a send request with zero data length (called "zero-byte message") without providing the remote address for the RDMA operation. When processing the request, the requester sends a packet without payload (but still with RDMA protocol headers and correct PSNs) to the responder and the responder

will not check the operation address when receiving the packet. This feature is initially designed to keep inactive RDMA connections alive, but is exactly proper to be used for protecting tail drops. The injecting of dummy headers can be done when posting a batch of RDMA requests in the RDMA application (RPC library or RDMA-coupled applications) or detecting the drained send queue in the RDMA driver. The dummy requests can be marked as unsignal to avoid generating corresponding completion signals. Note that there is also an exception: for each tail RDMA read operation, some dummy READs should be injected to pull dummy headers sent by the responder.

Inevitably, implementing dummy headers through zero-byte message consumes additional CPU cycles and PCIe operations. In the worst case where each flow consists of only one RDMA request, for each RDMA request, N additional operations should be posted by CPU and processed by RNIC if using N dummy headers. To decrease the additional RDMA operations, we calculate the time interval Δt between the current post time t_c and the last post time t_l for each QP: $\Delta t = t_c - t_l$, and only inject dummy headers when $\Delta t > T$. The parameter T should be configured smaller than the tolerable tail network latency for applications since the expected transmission completion time with corruption is approximately $T + RTT$, where RTT is the network transmission time of the initial request. For example, if a KV store application hopes to control network latency in 300 µs and its measured or estimated network tail latency without corruption (due to congestion) is 100 µs, then T should be configured up to 200 µs.

6 Evaluation

In this section, we evaluate the impact of packet corruption and our strategies with packet corruption through experiments from different aspects. Subsection 6.1 introduces the simulation and testbed we use in our evaluation. Subsection 6.2 demonstrates the long tail latency caused by timeout with the flow completion time (FCT) distribution under different packet corruption ratios in simulation and the long tail latency caused by timeout in our testbed. Subsection 6.3 shows the effects of the optimizations in both simulation and testbed results and also verifies the theories of probability in Section 3 and Section 4.

6.1 Experimental Setup

We build a simulator in ns-3 to simulate the RDMA protocol. We implement a simplified transport layer of the RDMA protocol, including the support for RDMA_WRITE/RDMA_SEND operations with "First", "Middle", "Last", and "Only" value in BTH headers, PSN/ePSN mechanism and standard TX/RX state machine dictated in IB specification, and ACK and NAK mechanisms. Our understanding of PSN and ACK/NAK mechanisms is also verified through drop tests on our RDMA testbed.

In simulation tests, the topology we use is standard non-blocking CLOS [31] ($k = 4$, 16 hosts). Since the topology is small, a high corruption occurrence rate (ω, referred in Subsection 3.2) is set to obtain observable results. In all simulation tests, we assume the corruption occurs on all links ($\omega = 1$) (except for the links between hosts and ToR switches, as referred in Subsection 2.2). Datacenter-scale network simulation takes a very long time to proceed, which is another topic in research [32]. In simulation tests, we use WebSearch and DataMining as the traffic patterns. The arrival of flows obeys Poission distribution and the parameter of the Poission distribution is set according to the traffic load, i.e., a low link utilization (0.1) and a high link utilization (0.6). There are two kinds of traffic modes used in simulation: point-to-point and full mesh. In the point-to-point traffic mode, one requester initiates RDMA_WRITE requests to one responder. In the full-mesh traffic mode, all nodes in the CLOS topology initiate RDMA_WRITE requests to all the other nodes. The destination node is randomly picked for each request. To show the best effect of our optimizations, we configure the parameter of inserting dummy headers $T = 0$ in all tests.

Our evaluation is conducted on a testbed consisting of one Barefoot Wedge 100BF-65X P4 programmable switch with 6.5 Tbps Barefoot Tofino ASIC and two server machines. Each server machine has a 24-core CPU (Intel Xeon E5-2650 v4) and a Mellanox ConnectX-6 RNIC. The packet corruption is simulated by randomly dropping packets on the switch. The lossy-RoCE acceleration features on Mellanox RNICs are disabled by default since the mechanisms are non-standard enhancements without specification provided by NIC vendors. The influence of the acceleration features is discussed in Subsection 7.1. The benchmark test is proceeded with the Perftest benchmark tool[①].

[①] https://github.com/linux-rdma/perftest, July 2022.

For each test case, we conduct 10 000 iterations 1 024 B RDMA_WRITE ping-pong tests and mark down the average, 99.9th and maximal latency. In each ping-pong test, the client sends an request to the server, and then the server echoes an response to the client.

6.2 Impact of Packet Corruption on RDMA Transport

Simulation Results. To understand the influence of packet corruption on applications, we evaluate the FCT with different packet corruption rates in simulation. Fig.10 shows the FCT Cumulative Distribution Function (CDF) of 10 000 flows in simulation. The distribution of the flows is WebSearch and the average load (link utilization) is 10%/60%. We can observe extremely long latencies and flow CDF distribution distortion even with a small packet corruption rate (0.001). With higher link utilization, the distortion of FCT CDF is more severe.

Testbed Results. We also run 10 000 iterations 1 024 B RDMA_WRITE ping-pong test with different packet corruption rates on our testbed. Fig.11(a) shows the average, 99.9th, and maximal latency in the test. The maximal tail latency is more than 266 ms (the value for retransmission timer) when there is packet corruption. Besides, the same extremely long 99.9th tail latency is also observable when the corruption rate is no less than 1/1 024 (12.7% corrupted links referred in Subsection 2.2).

The average latency in the test can be used to calculate the number of timeout events and then verify the theories of probability in Section 3. Since the transmit depth is 1 ($\mathbb{E}(\frac{1}{S}) = 1$), the flow tail timeout probability is $P_{\text{tail}} = 1 \times \beta_{\max} \times 1 = \beta_{\max}$, the flow tail ACK timeout probability is $P_{\text{tail\_ACK}} = 0.06 \times 1 \times \beta_{\max} \times 1 = 0.06\beta_{\max}$, the NAK corruption probability is $P_{\text{NAK}} \approx 1^2 \times \beta_{\max} \times$

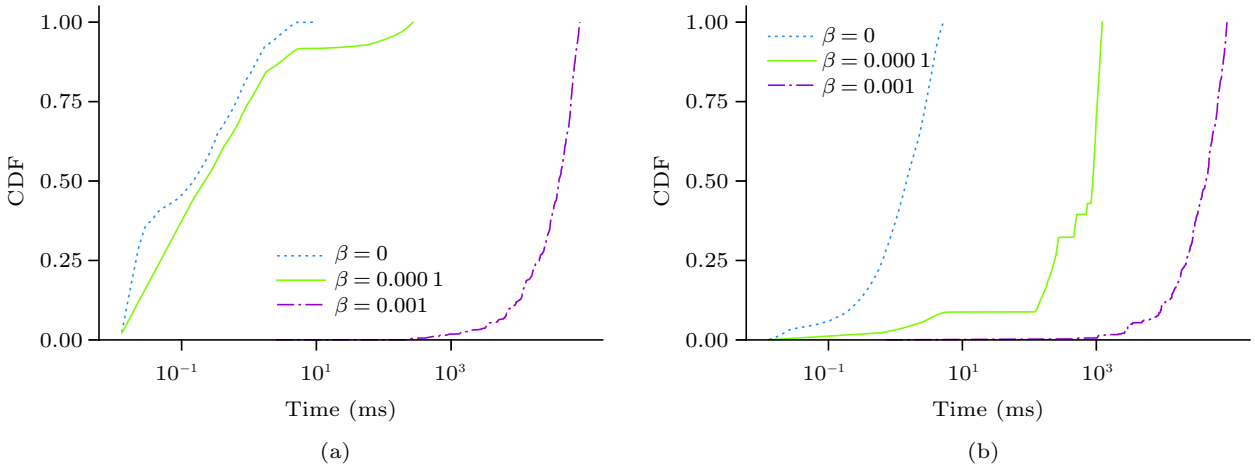

Fig.10. CDF of flow completion time with different packet corruption rates (β) and link loads (U). (a) $U = 0.1$. (b) $U = 0.6$.

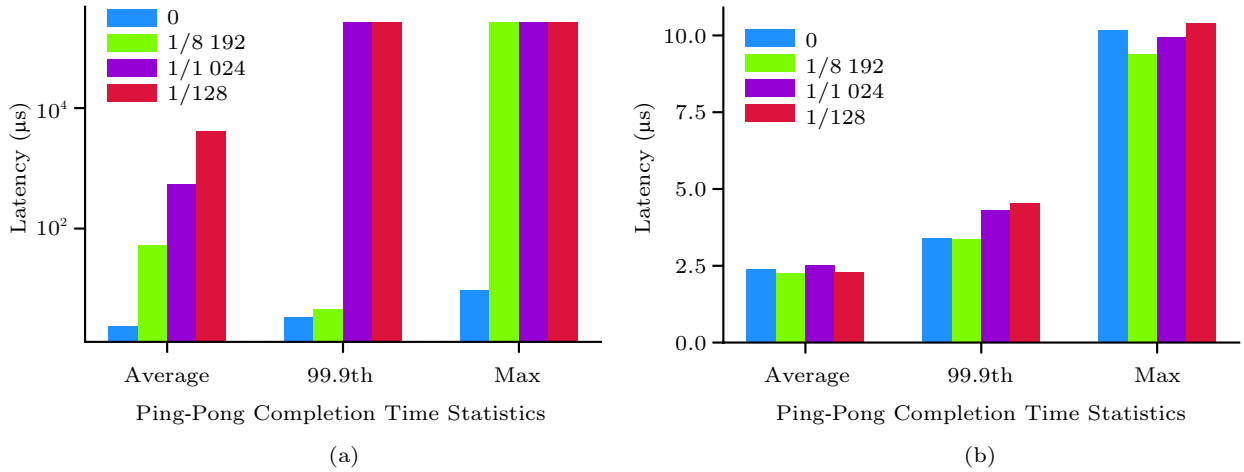

Fig.11. Ping-pong latency with different corruption rates with/without optimizations. (a) Without optimizations. (b) With the optimization of 2 dummy headers, 1 NAK replication, and 1 NAKed packet replication.

$0.06\beta_{\max} \times 1^2 = 0.06\beta_{\max}^2$, and the packet corruption twice probability is $P_{\text{twice}} = 1 \times \beta_{\max}^2 \times 1 = \beta_{\max}^2$. Since $P_{\text{tail}} >> P_{\text{tail\_ACK}} >> P_{\text{twice}} >> P_{\text{NAK}}$, taking the ping-pong communication mode into consideration, the timeout probability $P_{\text{timeout}} = 1 - (1 - P_{\text{tail}})^2 \approx 2\beta_{\max}$. Thus in 100 000 iteration tests, the number of timeout events is approximately $200\,000 \times \beta_{\max}$ with corruption ratio $\beta_{\max}$. For example, the estimated average latency of 1/128 corruption rate with the probabilistic model approximates $\frac{266\,000\ \mu\text{s} \times 200\,000 \times 1/128}{100\,000} \approx 4 \times 10^3\ \mu\text{s}$, while the average latency is 3.8×10^3 in the test.

6.3 Effectiveness of Different Optimizations

Simulation Results. Figs.12 and 13 show the FCT CDF under different optimizations with varied load (U) and corruption rate (β) in the point-to-point simulation test. The workloads are WebSearch (Fig.12) and DataMining (Fig.13). The parameters of default optimization (All) are 2 dummy headers, 1 NAK replication, and 3 NAKed packet replications. The more effective parameters of optimization (All Optimized) are 3 dummy headers, 2 NAK replication, and 4 NAKed packet replications. "NAK", "Twice", "Dummy", and "Repeat" represent enabling NAK replication, NAKed packet replication, dummy headers, and replicating flow tail packet, respectively. "NoLoss" represents that no packet corruption is configured without optimizations. "None" represents that none of the optimizations is enabled. For all the cases, the combination of optimizations (labeled as "All" and "All Optimized" in the figures) can effectively decrease the flow latency by several order of magnitudes. Except the case of high corruption rate and load ($\beta = 10^{-3}, U = 0.6$), the CDF of flows with all optimizations with the packet loss is very close to the CDF without packet loss, which shows that the combination of optimizations is effective for dealing with packet corruption. The more aggressive parameters labeled as "All Optimized" (3 dummy headers, 2 NAK replications and 4 NAKed packet replications) result in smaller FCT compared with the default opti-

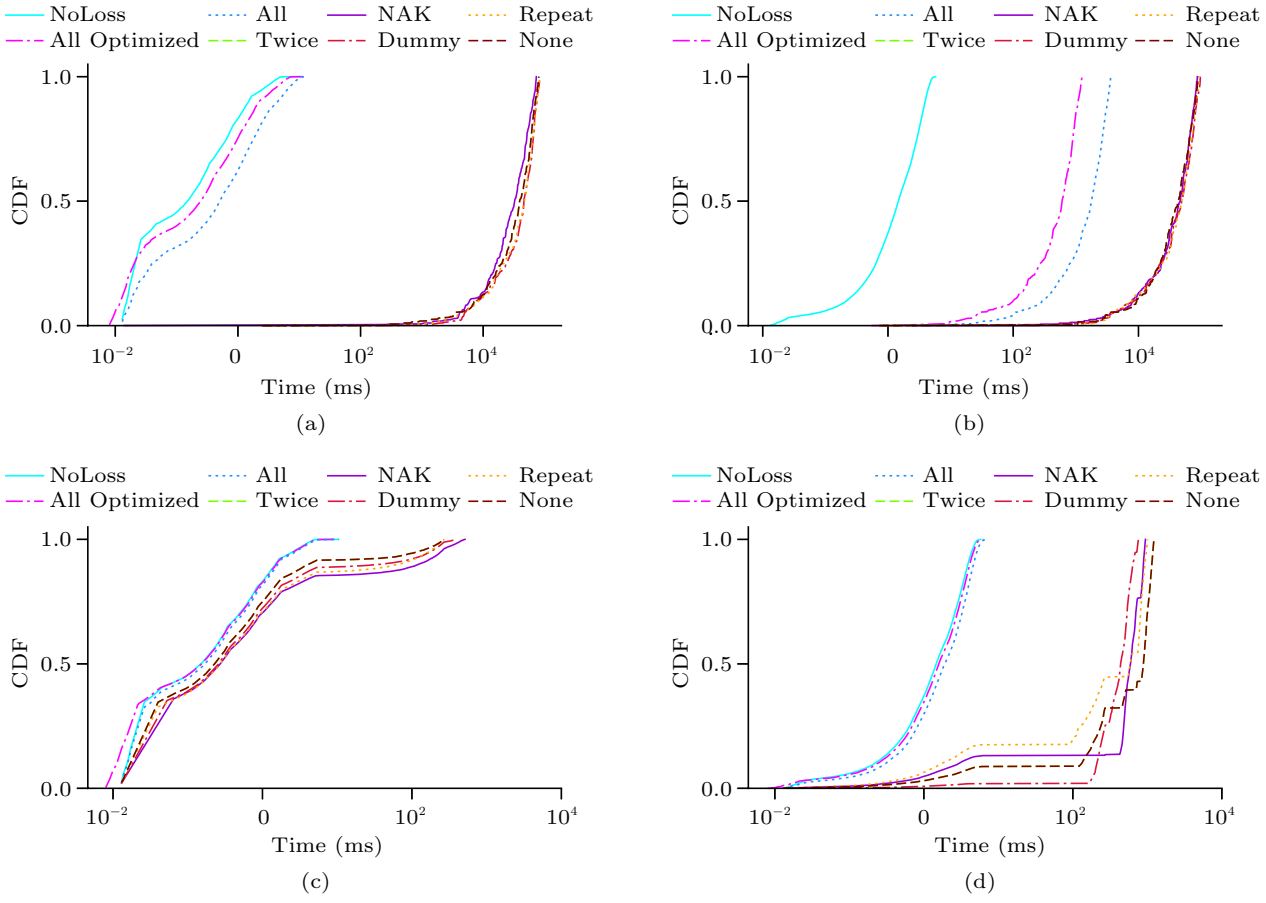

Fig. 12. Flow completion time in point-to-point tests with different packet corruption rates (β) and link loads (U), and with flow distribution WebSearch. (a) $\beta = 10^{-3}$, $U = 0.1$. (b) $\beta = 10^{-3}$, $U = 0.6$. (c) $\beta = 10^{-4}$, $U = 0.1$. (d) $\beta = 10^{-4}$, $U = 0.6$.

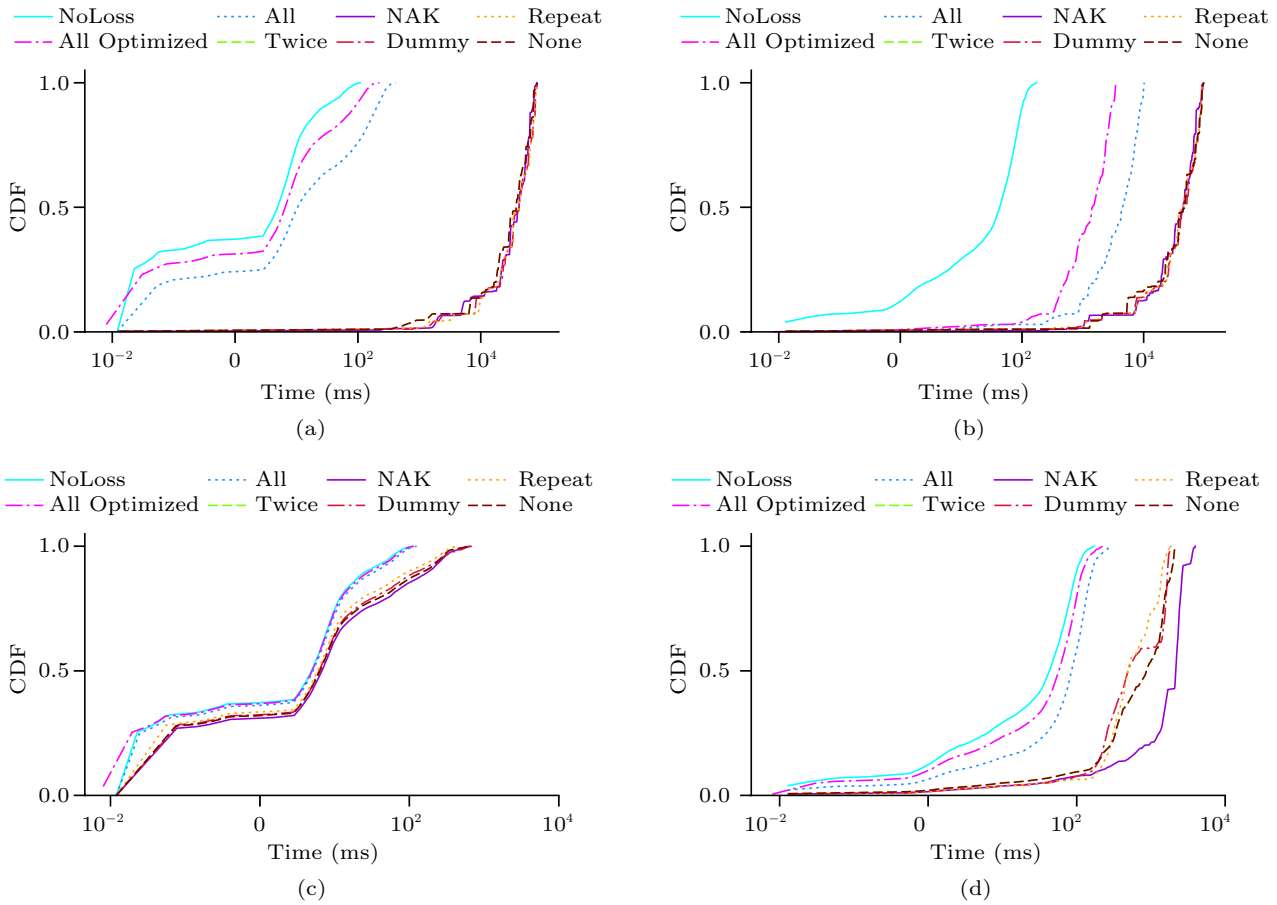

Fig. 13. Flow completion time in point-to-point tests with different packet corruption rates (β) and link loads (U), and with flow distribution DataMining. (a) $\beta = 10^{-3}$, $U = 0.1$. (b) $\beta = 10^{-3}$, $U = 0.6$. (c) $\beta = 10^{-4}$, $U = 0.1$. (d) $\beta = 10^{-4}$, $U = 0.6$.

mization parameters labeled as "All" (2 dummy headers, 1 NAK replication and 3 NAKed packet replications), which verifies that using more replications can achieve better performance. Under the high corruption rate and link load ($\beta = 10^{-3}, U = 0.6$), though the combination of optimizations still can reduce the FCT by several orders of magnitude, the additional latency compared with no packet loss is still high since the bandwidth becomes the bottleneck with such a high corruption rate and link load. Note that the link load U is calculated according to the data payload of injected requests into the link, where retransmissions in transport are not counted. The reduction of FCT is not obvious with any singe kind of optimization, which indicates that all the kinds of corruption occur in the test.

Fig. 14 shows the FCT CDF under different optimizations of WebSearch flow distribution with varied loads (U) and corruption rates (β) in the full-mesh simulation test. In the full-mesh test, as we referred, each client node sends requests to all the other servers nodes. Similar to the point-to-point test results, in the full-mesh test, the FCT CDF with all optimizations (All) is close to the CDF without packet loss (NoLoss).

Testbed Results. Fig. 11(b) shows the average, 99.9th, and maximal latency with optimizations in 10 000 iterations 1 024 B RDMA_WRITE ping-pong tests with different packet corruption rates on our testbed. The parameters are 2 dummy headers, 1 NAK replication and 1 NAKed packet replication. The maximal latency is less than 10 µs in the test, which indicates that no timeout event occurs. Here we choose to use 2 dummy headers rather than 1 since the timeout probability with 1 dummy header with $\beta_{\max} = \frac{1}{128}$ is $P_{\text{timeout}} \approx 2P_{\text{tail}}^{\text{dummy}} \approx 2 \times 0.06^N \mathbb{E}(\frac{1}{S}) L \beta_{\max}^{N+1} \omega = 2 \times 0.06 \times 1 \times 1 \times (\frac{1}{128})^2 \times 1 = 7 \times 10^{-6}$. The timeout event may be observed with 100 000 iterations tests since the mathematical expectation of the number of timeout events is $100\,000 \times 7 \times 10^{-6} = 0.7$, while the timeout probability with 2 dummy headers $P_{\text{timeout}} \approx 2P_{\text{tail}}^{\text{dummy}} \approx 2 \times 0.06^N \mathbb{E}(\frac{1}{S}) L \beta_{\max}^{N+1} \omega = 2 \times 0.06^2 \times 1 \times 1 \times (\frac{1}{128})^3 \times 1 = 3 \times 10^{-9}$. The time-

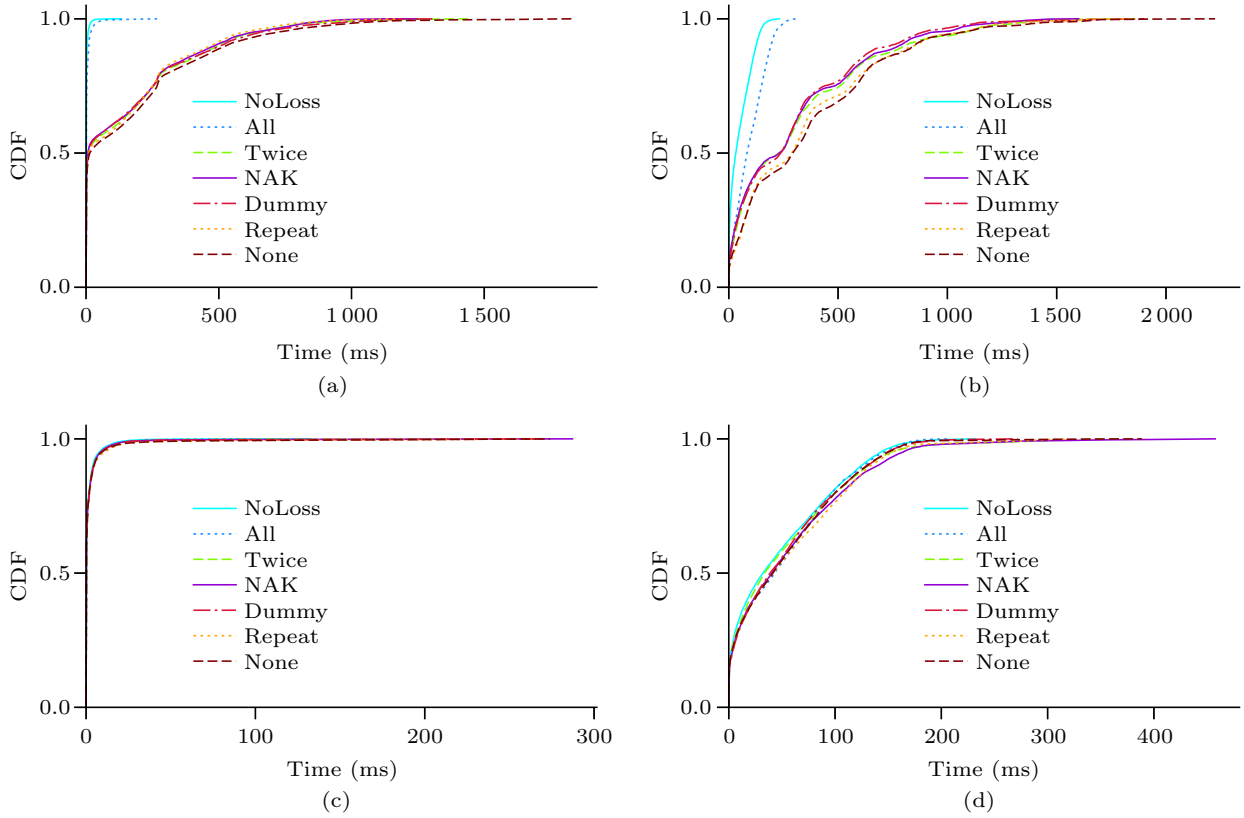

Fig.14. Flow completion time of fullmesh test with different packet corruption rates (β) and link loads (U), and with flow distribution WebSearch. The parameters of default optimization (All) are 2 dummy headers, 1 NAK replication and 3 NAKed packet replications. "NAK", "Twice", "Dummy", and "Repeat" represent enabling NAK replication, NAKed packet replication, dummy headers, and replicating flow tail packet respectively. "NoLoss" represents no packet corruption is configured without optimizations. "None" represents none of the optimizations is enabled. (a) $\beta = 10^{-3}$, $U = 0.1$. (b) $\beta = 10^{-3}$, $U = 0.6$. (c) $\beta = 10^{-4}$, $U = 0.1$. (d) $\beta = 10^{-4}$, $U = 0.6$.

out probability of NAK corruption with 1 NAK replication is $P_{\text{timeout}} \approx 2P_{\text{NAK}}^{\text{repeat}} \approx 2 \times L^2\beta_{\max}\delta^N\omega^2 = 2 \times 1^2 \times \frac{1}{128} \times (0.06 \times \frac{1}{128})^2 \times 1^2 = 3 \times 10^{-9}$. The timeout probability of packet twice corruption timeout with one NAKed packet replication is $P_{\text{timeout}} \approx 2P_{\text{twice}}^{\text{repeat}} \approx 2L\beta_{\max}^{N+1}\omega = 2 \times 1 \times (\frac{1}{128})^3 \times 1 = 9 \times 10^{-7}$. These two timeout possibilities are less than 10^{-6} (0.1 times in 100 000 iterations of tests) and should not be observed in 100 000 iteration tests.

To verify that each optimization works, we run the same test "without" each optimization mechanism with packet corruption ratio $\beta_{\max} = \frac{1}{128}$. Fig. 14 shows the average, 99.9th, and maximal flow latency with 0 or 1 dummy header, without NAK replication or NAKed packet replication. The timeout probability with 0 and 1 dummy header is $P_{\text{timeout}} \approx 2P_{\text{tail}} = 1/64$ and $P_{\text{timeout}} \approx 2P_{\text{tail}}^{\text{dummy}} \approx 7 \times 10^{-6}$ respectively. Without NAK replication, the timeout probability is $P_{\text{timeout}} \approx 2P_{\text{NAK}} \approx 2 \times 0.06 \times (\frac{1}{128})^2 \approx 7 \times 10^{-5}$. Without NAK packet replication, the timeout probability is $P_{\text{timeout}} \approx 2P_{\text{twice}} \approx 2 \times (\frac{1}{128})^2 \approx 1.2 \times 10^{-4}$. Thus in our test of 100 000 iterations, the expectation of times of different kinds of timeout is 1 562 (without dummy headers), 0.7 (with 1 dummy), 7 (without twice repeat) and 12 (without NAK repeat) respectively, and the average latency increment should approximate 4×10^4 µs, 3×10^0 µs, 2×10^1 µs, and 3×10^1 µs respectively, which match the results of our tests in Fig.15.

7 Discussion

7.1 Lossy RDMA Compatibility

In order to avoid the problems incurred by PFC such as victim flow [16] and deadlock [1], some researchers and RNIC vendors [19] try to adapt and deploy RDMA in a lossy environment without PFC. For new generations of RNICs, e.g., the Mellanox ConnectX-6 RNIC, selective repeat is supported to replace the Go-back-to-N retransmission mechanism dictated in IB (Infiniband) specifications. The impact of repeating NAK and NAK packets depends on the specific implementation of retransmission.

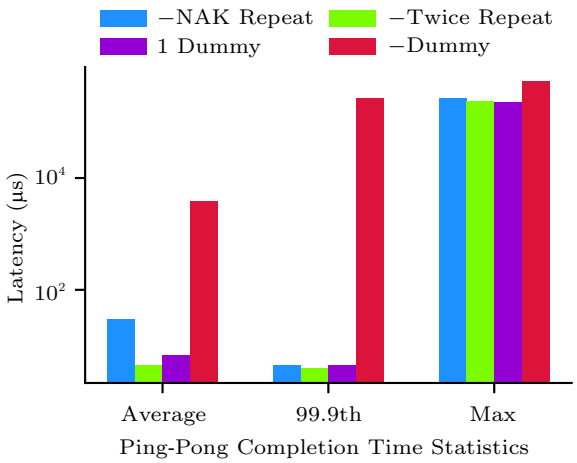

Fig.15. Ping-pong latencies with different corruption rates with different optimizations. The default optimization parameters are 2 dummy headers, and 1 NAK/NAKed packet (twice) replication. "$-X$" means the X mechanism is disabled.

Besides, Mellanox ConnectX RNICs implement a series of lossy features called lossy-RoCE accelerations. The lossy-RoCE accelerations include optimizations on transport such as adaptive retransmission, limited send window, and optimizations on congestion control such as slow restart. The difference between corruption loss in the lossless network and congestion loss in the lossy network is that the latter should experience flow rate reduction while the former should not. Thus the NAKs caused by corruption may incur spurious flow rate reduction. The optimizations in Section 4 may exacerbate this mistake since they repeat NAKs. Thus, we repeat the ping-pong tests in Subsection 6.2 and Subsection 6.3 with enabling and disabling lossy-RoCE accelerations, and the results show that there is no difference between the flow latencies when enabling and disabling lossy-RoCE accelerations.

While, the "dummy headers" mechanism designed for protecting tail packet loss does not have such potential side effects in the lossy RDMA network. However, we suggest that the feature should be finally implemented into the RNIC hardware by NIC vendors to reduce CPU and PCIe overhead of the software implementation.

7.2 Cooperation with Corruption Repair Systems

Some datacenter operators may deploy network monitoring systems [2, 17] based on packet loss (NAK) or timeout events. The monitoring systems are used to fast locate the malfunctioning link/device for the network operator. Using optimizations for packet corruption may interfere these applications since the numbers of NAK and timeout events are changed. Some emergency measures such as breakdown switch isolation and path switching [2] might be interfered. The network monitoring systems should be adapted when coexisting with loss optimizations. However, since we do not eliminate the NAK, the existing monitoring system should always be able to locate the problem. P4-based network monitoring systems can be designed and deployed together with these optimizations.

8 Related Work

Zhu *et al.* discussed the influence of packet loss on goodput in RDMA [16]. Most of RDMA applications are also latency-sensitive applications; thus the long tail latency caused by packet loss deserves attention and discussions. Besides, a low packet loss rate (e.g., less than 10^{-3}) incurs limited goodput loss, but can result in an extremely high tail latency through timeout events. Guo *et al.* discussed the transport livelock problem in RDMA caused by organized packet drop (e.g., one packet drop every 256 packets) [1]. Packet corruption we discuss in the paper is random and irregular, which may not cause livelock commonly.

Similar to Zhu *et al.* [16], Wang *et al.* [20] also discussed the throughput loss caused by packet loss in the RDMA network and brought up remedial mechanisms such as checking ePSN and sending twice (i.e., repeating in Subsection 4.3) while they did not discuss the more important flow latency either. Unfortunately, they misunderstood the mechanism of NAK. The "NAK interval" does not exist on real RNICs. We have verified on Mellanox RNICs that if the NAK is dropped, the retransmission timeout is always triggered on the requester, just as the protocol specification dictates. Besides, without a probabilistic model and consideration for the size difference of packets and ACK/NAKs, the occurrence frequencies of causes of timeout are misestimated. Both checking ePSN and sending twice are not currently deployable since RNICs have to be modified. We have shown in Subsection 3.5 and Subsection 6.3 that our optimizations are more precise with lower overhead compared with sending twice in Wang *et al.*'s work [20].

9 Conclusions

The paper investigates the packet corruption in RDMA network. Unlike previous work, we treated

the corruption as a quantifiable and controllable phenomenon. Based on the conclusions of previous work, we built a probabilistic model on packet corruption and analyzed all kinds of corruption that may cause timeout. With these analyses, datacenter operators may have some clues of the annoying "slow IOs" (application requests with extremely long latency) [2] caused by corrupted links. RDMA NIC vendors can also take such features into consideration when designing new RNICs. Besides, we also provided an example of switch-based solutions with P4 switches, which may be inspiring for related researchers. Finally, our optimizations, especially the "dummy headers" in software, can be directly applied with little software modification in current RDMA systems in industry.

References

[1] Guo C, Wu H, Deng Z, Soni G, Ye J, Padhye J, Lipshteyn M. RDMA over commodity ethernet at scale. In *Proc. the 2016 ACM SIGCOMM Conference*, August 2016, pp.202-215. DOI: 10.1145/2934872.2934908.

[2] Gao Y, Li Q, Tang L *et al.* When cloud storage meets RDMA. In *Proc. the 18th USENIX Symposium on Networked Systems Design and Implementation*, April 2021, pp.519-533.

[3] Kalia A, Kaminsky M, Andersen D G. Design guidelines for high performance RDMA systems. In *Proc. the 2016 USENIX Conference on USENIX Annual Technical Conference*, June 2016, pp.437-450.

[4] Kalia A, Kaminsky M, Andersen D G. Using RDMA efficiently for key-value services. In *Proc. the 2014 ACM Conference on SIGCOMM*, August 2014, pp.295-306. DOI: 10.1145/2619239.2626299.

[5] Mitchell C, Geng Y, Li J. Using one-sided RDMA reads to build a fast, CPU-efficient key-value store. In *Proc. the 2013 USENIX Annual Technical Conference*, June 2013, pp.103-114.

[6] Dragojević A, Narayanan D, Hodson O, Castro M. FaRM: Fast remote memory. In *Proc. the 11th USENIX Conference on Networked Systems Design and Implementation*, April 2014, pp.401-414.

[7] Wei X, Shi J, Chen Y, Chen R, Chen H. Fast in-memory transaction processing using RDMA and HTM. In *Proc. the 25th Symposium on Operating Systems Principles*, October 2015, pp.87-104. DOI: 10.1145/2815400.2815419.

[8] Chen Y, Wei X, Shi J, Chen R, Chen H. Fast and general distributed transactions using RDMA and HTM. In *Proc. the 11th European Conference on Computer Systems*, April 2016, Article No. 26. DOI: 10.1145/2901318.2901349.

[9] Yang J, Izraelevitz J, Swanson S. Orion: A distributed file system for non-volatile main memories and RDMA-capable networks. In *Proc. the 17th USENIX Conference on File and Storage Technologies*, February 2019, pp.221-234.

[10] Kim J, Jang I, Reda W, Im J, Canini M, Kostić D, Kwon Y, Peter S, Witchel E. LineFS: Efficient smartNIC offload of a distributed file system with pipeline parallelism. In *Proc. the 28th ACM SIGOPS Symposium on Operating Systems Principles*, October 2021, pp.756-771. DOI: 10.1145/3477132.3483565.

[11] Weil S A, Brandt S A, Miller E L, Long D D E, Maltzahn C. Ceph: A scalable, high-performance distributed file system. In *Proc. the 7th Symposium on Operating Systems Design and Implementation*, November 2006, pp.307-320.

[12] Aghayev A, Weil S, Kuchnik M, Nelson M, Ganger G R, Amvrosiadis G. File systems unfit as distributed storage backends: Lessons from 10 years of Ceph evolution. In *Proc. the 27th ACM Symposium on Operating Systems Principles*, October 2019, pp.353-369. DOI: 10.1145/3341301.3359656.

[13] Kalia A, Kaminsky M, Andersen D G. FaSST: Fast, scalable and simple distributed transactions with two-sided (RDMA) datagram RPCs. In *Proc. the 12th USENIX Symposium on Operating Systems Design and Implementation*, November 2016, pp.185-201.

[14] Su M, Zhang M, Chen K, Guo Z, Wu Y. RFP: When RPC is faster than server-bypass with RDMA. In *Proc. the 12th European Conference on Computer Systems*, April 2017, pp.1-15. DOI: 10.1145/3064176.3064189.

[15] Kalia A, Kaminsky M, Andersen D G. Datacenter RPCs can be general and fast. In *Proc. the 16th USENIX Conference on Networked Systems Design and Implementation*, February 2019, pp.1-16.

[16] Zhu Y, Eran H, Firestone D, Guo C, Lipshteyn M, Liron Y, Padhye J, Raindel S, Yahia M H, Zhang M. Congestion control for large-scale RDMA deployments. In *Proc. the 2015 ACM Conference on Special Interest Group on Data Communication*, August 2015, pp.523-536. DOI: 10.1145/2785956.2787484.

[17] Zhuo D, Ghobadi M, Mahajan R, Förster K T, Krishnamurthy A, Anderson T. Understanding and mitigating packet corruption in data center networks. In *Proc. the Conference of the ACM Special Interest Group on Data Communication*, August 2017, pp.362-375. DOI: 10.1145/3098822.3098849.

[18] Zhuo D, Ghobadi M, Mahajan R, Phanishayee A, Zou X K, Guan H, Krishnamurthy A, Anderson T. RAIL: A case for redundant arrays of inexpensive links in data center networks. In *Proc. the 14th USENIX Symposium on Networked Systems Design and Implementation*, March 2017, pp.561-576.

[19] Mittal R, Shpiner A, Panda A, Zahavi E, Krishnamurthy A, Ratnasamy S, Shenker S. Revisiting network support for RDMA. In *Proc. the 2018 Conference of the ACM Special Interest Group on Data Communication*, August 2018, pp.313-326. DOI: 10.1145/3230543.3230557.

[20] Wang Y, Liu K, Tian C, Bai B, Zhang G. Error recovery of RDMA packets in data center networks. In *Proc. the 28th International Conference on Computer Communication and Networks*, July 29-August 1, 2019. DOI: 10.1109/ICCCN.2019.8846946.

[21] Langley A, Riddoch A, Wilk A *et al.* The QUIC transport protocol: Design and Internet-scale deployment. In *Proc. the Conference of the ACM Special Interest Group on Data Communication*, August 2017, pp.183-196. DOI: 10.1145/3098822.3098842.

[22] Bosshart P, Daly D, Gibb G *et al.* P4: Programming protocol-independent packet processors. *ACM SIGCOMM Comput. Commun. Rev.*, 2014, 44(3): 87-95. DOI: 10.1145/2656877.2656890.

[23] Marty M, Kruijf M, Adriaens J *et al.* Snap: A microkernel approach to host networking. In *Proc. the 27th ACM Symposium on Operating Systems Principles*, October 2019, pp.399-413. DOI: 10.1145/3341301.3359657.

[24] Singhvi A, Akella A, Gibson D *et al.* 1RMA: Re-envisioning remote memory access for multi-tenant datacenters. In *Proc. the 2020 Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, August 2020, pp.708-721. DOI: 10.1145/3387514.3405897.

[25] Monga S K, Kashyap S, Min C. Birds of a feather flock together: Scaling RDMA RPCs with Flock. In *Proc. the 28th ACM SIGOPS Symposium on Operating Systems Principles*, October 2021, pp.212-227. DOI: 10.1145/3477132.3483576.

[26] Handley M, Raiciu C, Agache A, Voinescu A, Moore A W, Antichi G, Wójcik M. Re-architecting datacenter networks and stacks for low latency and high performance. In *Proc. the Conference of the ACM Special Interest Group on Data Communication*, August 2017, pp.29-42. DOI: 10.1145/3098822.3098825.

[27] Li J, Wang Q, Lee P P C, Shi C. An in-depth analysis of cloud block storage workloads in large-scale production. In *Proc. the 2020 IEEE International Symposium on Workload Characterization*, October 2020, pp.37-47. DOI: 10.1109/IISWC50251.2020.00013.

[28] Geng J, Yan J, Ren Y, Zhang Y. Design and implementation of network monitoring and scheduling architecture based on P4. In *Proc. the 2nd International Conference on Computer Science and Application Engineering*, October 2018, Article No. 182. DOI: 10.1145/3207677.3278059.

[29] Ye J L, Chen C, Huang Chu Y. A weighted ECMP load balancing scheme for data centers using P4 switches. In *Proc. the 7th IEEE International Conference on Cloud Networking*, October 2018. DOI: 10.1109/CloudNet.2018.8549549.

[30] Sultana N, Sonchack J, Giesen H, Pedisich I, Han Z, Shyamkumar N, Burad S, DeHon A, Loo B T. Flightplan: Dataplane disaggregation and placement for P4 programs. In *Proc. the 18th USENIX Symposium on Networked Systems Design and Implementation*, April 2021, pp.571-592.

[31] Singh A, Ong J, Agarwal A *et al.* Jupiter rising: A decade of Clos topologies and centralized control in Google's datacenter network. In *Proc. the 2015 ACM Conference on Special Interest Group on Data Communication*, August 2015, pp.183-197. DOI: 10.1145/2829988.2787508.

[32] Zhang Q, Ng K K W, Kazer C, Yan S, Sedoc J, Liu V. MimicNet: Fast performance estimates for data center networks with machine learning. In *Proc. the 2021 ACM SIGCOMM Conference*, August 2021, pp.287-304. DOI: 10.1145/3452296.3472926.

Yi-Xiao Gao received his B.S. degree in information management and information system from Nanjing University, Nanjing, in 2017. He is currently a Ph.D. candidate in the Department of Computer Science and Technology and State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing. His research interests include datacenter networking, congestion control, distributed network and storage systems.

Chen Tian received his B.S., M.S., and Ph.D. degrees in communication engineering from Huazhong University of Science and Technology, Wuhan, in 2000, 2003, and 2008, respectively. He is currently a professor with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing. From 2012 to 2013, he was a postdoctoral researcher with the Department of Computer Science, Yale University, New Haven. His current research interests include data center networks, network function virtualization, distributed systems, and Internet streaming.

Wei Chen received his B.S. degree in computer science and technology from Nanjing University of Aeronautics and Astronautics, Nanjing, in 2021. He is working towards his M.S. degree in the Department of Computer Science and Technology and State Key Laboratory for Novel Software Technology at Nanjing University, Nanjing. His research interests include datacenter networks and distributed systems.

Duo-Xing Li received his B.S. degree in computer science and technology from Southeast University, Nanjing, in 2020. He is currently a Master student in the Department of Computer Science and Technology and State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing. His main research interest includes datacenter networks.

Jian Yan received his B.S. degree in computer science and technology from the School of Communication and Information Engineering, Shanghai University, Shanghai, in 2011, and his Ph.D. degree from the School of Microelectronics, Fudan University, Shanghai, in 2017. He is currently a principle engineer with Huawei Technologies Co. Ltd, Nanjing. His research interests include datacenter networks, computer architecture and FPGA-based reconfigurable computing.

Yuan-Yuan Gong received her B.S. degree in software engineering from Hunan University, Changsha, in 2019, and her M.S. degree in computer science and technology from Nanjing University, Nanjing, in 2022. Her research interest includes the task scheduling of big data systems.

Bing-Quan Wang received his B.S. degree in computer science and technology from the School of Computer Science and Technology, Southeast University, Nanjing, in 2016, and his M.S. degree in computer science and technology from the School of Computer Science and Technology, Nanjing University, Nanjing, in 2019. He is currently working for Huawei. His research interest focuses on data center networks.

Tao Wu received his M.S. degree in automation engineering from Nanjing University of Aeronautics and Astronautics, Nanjing, in 2009. He is currently a principle engineer with Huawei Technologies Co. Ltd, Nanjing. His research interest focuses on data center networks.

Lei Han is currently the director of Huawei Technologies Co. Ltd, Nanjing, and also the deputy director with Boole Laboratory, Data Communication, Huawei, Nanjing. His research interests include datacenter networks and networked systems.

Fa-Zhi Qi is currently the director of the Computing Center, Institute of High Energy Physics, Chinese Academy of Sciences, Beijing. He is also currently in charge of the computer network systems of China Spallation Neutron Source and the Computing and Network Communication High Energy Photon Source, Dongguan. His research interests include network management and network security, high-performance network, future network technology, and application of new network technology.

Shan Zeng received her M.S. degree in computer science and technology from Beihang University, Beijing, in 2010. She received her Ph.D. degree in computer science and technology from the University of Chinese Academy of Sciences, Beijing, in 2017. Her research interests include the application of network technologies and data transmission systems in high-energy physics.

Wan-Chun Dou received his Ph.D. degree in mechanical and electronic engineering from the Nanjing University of Science and Technology, Nanjing, in 2001. He is currently a full professor of the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing. He visited the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, as a visiting scholar respectively in 2005 and 2008. His research interests include workflow, cloud computing, and service computing.

Gui-Hai Chen received his B.S. degree in computer software from Nanjing University, Nanjing, in 1984, his M.E. degree in computer applications from Southeast University, Nanjing, in 1987, and his Ph.D. degree in computer science from The University of Hong Kong, Hong Kong, in 1997. He is a distinguished professor of Nanjing University, Nanjing. He had been invited as a visiting professor by the Kyushu Institute of Technology, Kitakyushu, University of Queensland, St Lucia, and Wayne State University, Detroit. He has a wide range of research interests with focus on parallel computing, wireless networks, data centers, peer-to-peer computing, high-performance computer architecture, and data engineering.