An Optimized RDMA QP Communication Mechanism for Hyperscale AI Infrastructure

Junliang Wang

wangjl23@chinatelecom.cn

China Telecom Co.,Ltd.

Baohong Lin

China Telecom Co.,Ltd.

Jiao Zhang

BUPT

Mengyu Sun

China Telecom Co.,Ltd.

Yongchen Pan

BUPT

Research Article

Keywords: Queue Pair (QP), Hyperscale Communication, Remote Direct Memory Access (RDMA), Multipath Transmission

Posted Date: April 1st, 2024

DOI: https://doi.org/10.21203/rs.3.rs-4174332/v1

Additional Declarations: No competing interests reported.

An Optimized RDMA QP Communication Mechanism for Hyperscale AI Infrastructure

Junliang Wang[1], Baohong Lin[1], Jiao Zhang[2], Mengyu Sun[1], Yongchen Pan[2]

[1]China Telecom Research Institute, China Telecom Co.,Ltd., Guangzhou, 510660, China.
[2]State Key Laboratory of Networking and Switching Technology, BUPT, Beijing, 100876, China.

Contributing authors: wangjl23@chinatelecom.cn; linbh@chinatelecom.cn; jiaozhangg@bupt.edu.cn; sunmyl@chinatelecom.cn; panyongcchen@bupt.edu.cn;

Abstract

The current artificial intelligence (AI) infrastructure widely employs remote direct memory access (RDMA) protocol for high-performance communication in networks, utilizing Reliable Connection (RC)-based Queue Pairs (QP) to ensure end-to-end correct and ordered data transmission. However, as the scale of AI infrastructure continues to expand, this RC-based QP communication mechanism faces deficiencies in scalability and is prone to congestion, resulting in degraded network transfer performance. In this paper, we propose an optimized RDMA QP communication mechanism to address scalability and congestion issues in hyper-scale AI infrastructure networks. Firstly, we replace RC-based QPs with Reliable Datagram (RD)-based QPs and propose a new reliable mechanism to address scalability problems, eliminating the need for repetitive QP establishment by AI processes during external communication. Additionally, to mitigate congestion caused by a single path, we implement multipath data transmission by introducing a new unordered reception method in the network software stack. Through experiments and simulation tests, the optimized RDMA QP communication in large-scale AI infrastructure exhibits exceptional scalability and significantly reduces the occurrence of congestion, resulting in an overall network performance improvement of over 15%.

Keywords: Queue Pair (QP), Hyperscale Communication, Remote Direct Memory Access (RDMA), Multipath Transmission

1 Introduction

The current AI infrastructure utilizes RDMA to fulfill its low-latency, high-throughput network communication requirements. To fully leverage computational resources, AI training tasks distribute model parameters and datasets across various network nodes for distributed training[21, 29, 30]. Subsequently, RDMA transfers are invoked through collective communication operations such as All-to-All and All-Reduce to synchronize training results across all nodes. Consequently, network traffic in AI infrastructure exhibits two key characteristics[12]: 1) rich connectivity, where each network node needs to establish communication with all other nodes.2) burstiness in traffic, with sporadic bursts of point-to-point traffic occurring within short time frames.

In the context of RDMA transmission, the establishment of QPs is essential for providing data transfer channels and maintaining connection states. Presently, RDMA transfers primarily use QPs based on the RC mechanism. When the network cluster is relatively small, RC-type communication mechanisms can meet the specific characteristics of network traffic in AI infrastructure[13]. However, as the scale of AI infrastructure grows, RDMA QP communication based on RC inevitably exhibits performance deficiencies[1, 19]. Firstly, the number of QPs established by RC will increase rapidly as the scale expands, which will lead to a decrease in the performance of the network interface card (NIC). Secondly, the rise in bursty point-to-point traffic, coupled with the single-path transmission nature of RC, increases the probability of congestion at specific nodes significantly.

Recent efforts have attempted to optimize RDMA QP communication mechanisms for larger-scale scenarios. However, scalability issues persist, and congestion control performance remains suboptimal at hyper scales. For instance, approaches like enabling extended reliable connection (XRC) in ConnectX series NICs (Series 5 and above) and introducing dynamically connected transport (DCT) in Series 6 aim to reduce QP numbers proportionally[15]. Yet, these methods still face QP bottleneck challenges as the cluster scales up. Some research has even proposed significant modifications to NIC architectures, such as IRN[12], SRNIC[28], and csRNA[10]. While these modifications enhance QP scalability, the high cost and potential impact on NIC reliability make them impractical for deployment in hyper scale AI infrastructure. Additionally, congestion control algorithms like DCQCN and TIMELY, used in these studies, cannot completely prevent collisions of multiple large flows at specific nodes (usually refers to servers or switches) in extremely large-scale AI training networks[6, 9, 11, 18], leading to congestion in network traffic at those points.

In order to avoid the above problems, we adopt an multipath RD-based mechanism, allowing each AI training process to reuse a set of QPs for all external communication. This approach offers several advantages in hyper-scale AI infrastructure: firstly, eliminating the need to establish a separate QP connection for each external process significantly reduces QP numbers, enhancing NIC processing performance. Secondly, by implementing multipath transmission on top of RD, we avoid the congestion vulnerabilities inherent in the single-path transmission of RC.

While the use of multipath RD instead of RC for RDMA QP communication brings advantages such as avoiding QP number explosion and reducing the frequency of congestion caused by collisions of large flows, it also poses some challenges. 1)

Current RDMA NICs only support unreliable datagram transmission[3], requiring retransmission processing on the host software stack to achieve reliability, increasing packet processing latency. 2) Under RD-based transmission, the receiving process handles all external communication tasks through a single QP, leading to potential interference between different communication flows and even head-of-line blocking issues[14, 20]. 3) The use of RD-based multipath transmission results in a large number of packets arriving out of order, and reorder processing negatively impacts network performance[17].

To address these challenges, we introduce ERD, an RDMA QP communication mechanism capable of efficient reliable datagram transmission. ERD establishes an independent reliability processing queue and employs a NACK-based retransmission in network software stack to minimize the impact of host-side reliability processing on lower-level packet transmission. Furthermore, the ERD employs a dual-path processing mechanism, allowing the NIC to pass received packets directly to the upper layer while handling retransmissions through a separate path. This approach avoids mutual interference and header congestion issues. Lastly, we add data fields to the payload of the original RDMA data frame and efficiently process packet reordering based on these fields. Experimental results demonstrate that in hyper-scale AI clusters, ERD consistently maintains a low value for the number of established QPs compared to RC-based mechanisms. Additionally, the frequency of congestion in the network is reduced by over 50%, leading to an overall network performance improvement of approximately 15% when using ERD.

In summary, our work contributes in the following key aspects:

1. We propose ERD[25], an RDMA QP communication mechanism based on RD transmission, significantly reducing the number of QPs needed for inter-node communication and enhancing scalability. Additionally, ERD adopts multipath transmission to alleviate congestion issues arising from single-path transmission.

2. We describe a NACK method and a dual-path processing mechanism that ensure reliable transmission while alleviating the latency introduced on the host side and avoiding head-of-line blocking issues caused by QP multiplexing.

3. We deploy a packet unordered reception module, introducing additional data packet fields for efficient reordering to support multipath transmission.

4. We deploy and evaluate the overall performance of ERD in RDMA QP communication in hyper scale AI clusters. Through NS-3 simulations, ERD demonstrates superior overall performance compared to RC, with an improvement of around 15%.

2 Background

2.1 RDMA Basic

Queue Pair (QP). QP is the fundamental unit of RDMA communication, defining the transmission channel between two endpoints[4, 22]. Every QP for both the sender and receiver includes a Send Queue (SQ), a Receive Queue (RQ), and a Completion Queue (CQ). Before transferring data through RDMA, the sender and receiver need to create their respective QPs. Each AI training process creates at least one QP when engaging in external communication through RDMA.

Work Queue Element (WQE). WQE is the queue element in SQ and RQ. After the creation of QP, the application layer on the sender side generates a work request, typically involving the transmission indicators (including address and data length) of data from a specific memory region (MR) on the sender to the corresponding MR on the receiver[32]. This work request is segmented into multiple smaller WQEs and pushed into the SQ for the RDMA NIC to process. Upon detecting new WQEs in the SQ from the host side, the control module of the NIC utilizes Direct Memory Access (DMA) engine to extract them into the WQE Cache. The NIC parses the WQEs in the cache, processes the data to be transmitted through the DMA engine, and accurately sends the packets to the receiver[26]. On the receiver side, the RQ generates WQEs before receiving the data, placing the corresponding data packet payload at the correct MR.

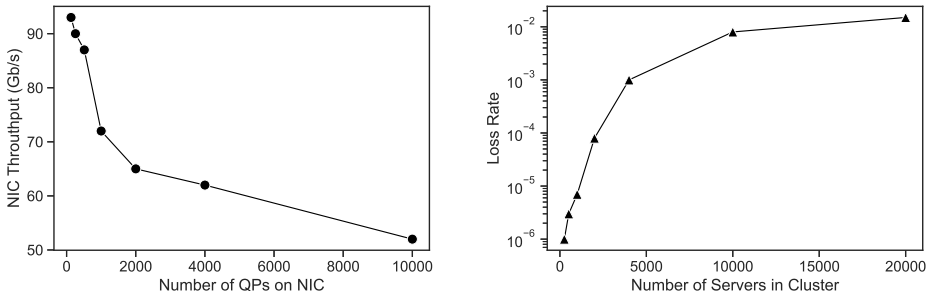
QP transmission types. According to the IBTA specifications[8], the QP transmission types of RDMA can be divided into four categories based on two dimensions: whether they are reliable and whether they establish a connection. These four categories are RD, RC, Unreliable Datagram (UD), and Unreliable Connection (UC)[31]. Currently, there are mature implementation solutions for RC, UD, and UC, while the research on RD type is only at the definition level[2]. Reliable data transmission is essential for AI large model inference and training. Given that RD research is still in its early stage and requires complex design in the network software stack, RC-type QP is the most widely used in current AI infrastructure. The current RDMA NIC can support fully connected RC communication in clusters with less than 2000 nodes. This capability relies on congestion control algorithms such as DCQCN, TIMELY, HPCC, etc., to mitigate congestion occurrences within the cluster.

2.2 Network Performance Issues of RC-type QP

In hyper-scale AI clusters, the connection dependencies and single-path transmission characteristics of RC-type QP pose challenges in terms of connection scalability and issues related to bursty traffic. These issues significantly impact network performance at both the NIC and node levels.

QP scalability issue. This issue occurs at the NIC level. During the AI training process, two communication primitives, namely All-to-All and All-Reduce, are employed to synchronize datasets or training results (model parameters) among all computing nodes. Under the RC-type QP, both communication primitives result in establishing full connections between processes, with each connection requiring the creation of a dedicated QP for inter-process data transfer[5]. Assuming the AI training cluster has a scale of N and each node has p processes, the total number of QPs required per node is $(N-1)\cdot p^2$. Due to the limited capacity of NIC buffers, maintaining such a large number of QPs significantly degrades the data processing performance of the NIC, thereby increasing network transmission latency. Figure 1(a) demonstrates the impact of the quantity of QPs on NIC throughput performance. As the number of QPs created rises to over 1000, the NIC's throughput performance rapidly declines, approaching nearly half of its original capacity.

Bursty traffic issue. This issue occurs at the node level. When transferring data packets in RC-type QP, to ensure the ordered arrival of data, only one path is utilized



(a) (b)

Fig. 1: Transmission performance degradation as network scale increases. (a) Throughput results under general RDMA NIC (Intel E810 Adapter). (b) Loss rate results under different cluster sizes (NS-3 simulator with RC-type flows).

at any given moment to send packets to the counterpart. For AI training processes, the network traffic along this RC path can reach several hundred Gbps[16], and simultaneously, there are multiple end-to-end RC transfers within the cluster. Despite various congestion control algorithms reducing the frequency of intersections and congestion between these RC transfers, the latency of congestion control is severe in ultra-large-scale clusters. Consequently, certain nodes are inevitably prone to significant flow collisions and congestion[24], adversely affecting the transmission performance of wide-ranging end-to-end transfers and even the overall network. Figure 1(b) illustrates how the continuous expansion of the network scale leads to an increasing packet loss rate, consequently causing a rapid decline in network performance.

3 Problems of Using RD-type QP with multipath transmission

Current AI training data centers commonly adopts RC-type QPs. This type of QP can directly leverage the features stored in RDMA NICs, such as connection status and reliability processing, to achieve end-to-end RDMA high-performance transmission. However, at an extremely large scale, the establishment of QPs and the single-path characteristics of RC impose a significant burden on network performance. Therefore, employing another reliable transmission mechanism, RD transmission, becomes a preferable choice.

In the RD transmission mode, each process establishes only one QP for external communication. At this point, QPs are no longer mutually associated with a specific connection. The WQEs stored in SQ may be directed to different processes of various endpoints, and the WQEs stored in RQ may handle data reception tasks for multiple external processes. While this QP reuse method significantly reduces the number of QPs, it lacks the association with fixed connections, making it unable to maintain reliability through connection context information.

To address the reliability of RD, the IBTA standard[8] introduces the EEC (End-to-End Context) mechanism. It establishes a pair of EEC queues between each group of nodes for communication similar to RC, dedicated to maintaining the reliability and ordered processing of data packets. Multiple processes with RD-type QPs can achieve reliable communication with the corresponding node through a single EEC, as illustrated in Figure 2.

Although RD-type QPs with EEC can achieve basic reliable transmission, it has some disadvantages compared to RC, making its application in AI infrastructure relatively limited. Firstly, due to the limited resources of current NICs, EEC can only be deployed in the host's kernel layer. When NIC receives an ACK packet, it needs to pass through the PCIe channel to reach the kernel EEC for reliability processing, introducing additional latency overhead to normal transmission. Secondly, EEC sends data streams based on a sliding window mechanism, allowing only one outstanding message, as illustrated in Figure 2. If a data packet is lost in the data stream, it halts the transmission of subsequent data until the lost packet is retransmitted and confirmed, leading to head-of-line blocking. Since EEC is reused by QPs of different processes, this head-of-line blocking can cause interruptions in several communication processes.

Another advantage of RD-type QPs is the ease of combining with multi-path transmission, avoiding congestion nodes in the network caused by the convergence of different high-traffic paths. However, using multi-path transmission can lead to unordered arrival of packets at the receiving end, requiring the NIC to buffer a large number of packets and reorder them. This imposes a significant burden on the NIC in terms of both caching and performance pressure.

Through the analysis of RD-type QPs in reliable and multi-path transmission, we have identified three challenges in applying multi-path RD:

C1. How to minimize the PCIe latency caused by the transfer of reliability processing from the NIC side to the host side? Due to the relatively fixed design framework of current RDMA NICs, achieving reliability for RD on the NIC side is challenging and needs to be transferred to the host side for processing. The existing EEC solution implements reliability in the host kernel, but the frequent ACK packets contribute significantly to the PCIe transfer latency between the host side and the NIC side, occupying a substantial portion of the normal transmission latency.

C2. How to avoid head-of-line blocking when sending packets while ensuring reliability? To ensure reliable transmission, the common practice (including EEC) is to use a sliding window mechanism, allowing the sender to proceed with subsequent data transmission only after receiving ACK confirmation packets. However, such practices are prone to severe head-of-line blocking in the context of RD-type QPs. An RD-type QP often carries out external communication tasks for multiple processes, and the loss of a data packet from one process can force the interruption of all these processes. Therefore, compared to RC, the consequences of head-of-line blocking in RD are more severe. To address this issue, a reliability assurance mechanism that does not introduce head-of-line blocking is needed.

C3. How to efficiently handle out-of-order packets caused by multi-path transmission? RD with multi-path transmission has a lower probability of congestion at the network side compared to RC with single-path transmission. However, multi-path

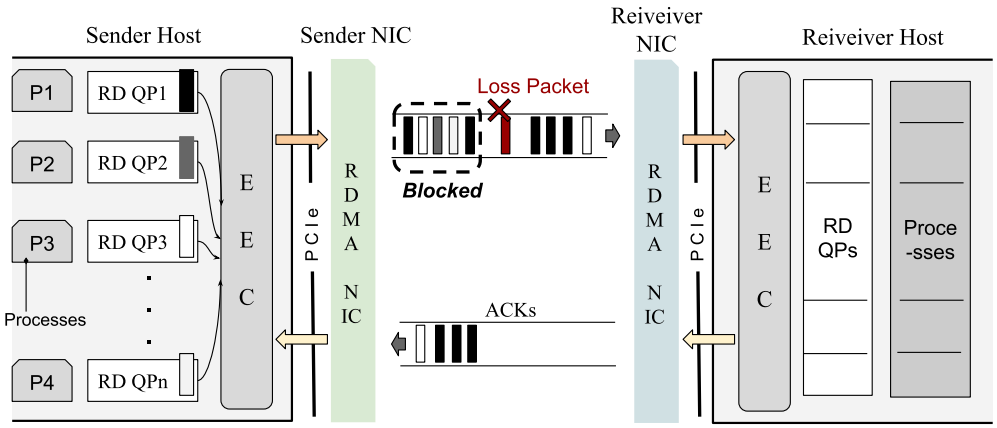


Fig. 2: RD-type QPs achieve reliable transmission with the assistance of EEC. Process P enforces RD QP to generate WQEs. WQEs destined for the same endpoint are collected into the same EEC. The EEC reads these WQEs, generates a series of data packets, and sends them according to the sliding window method.

transmission can result in the NIC receiving a large number of out-of-order packets in a short period. If the NIC is tasked with ordering all these packets, it would require a massive allocation of buffer space and firmware resources, inevitably reducing the throughput capacity of the NIC. Therefore, efficiently handling out-of-order packets in multi-path scenarios is one of the key challenges in ensuring high-performance transmission for RD.

4 Efficient RD Communication Architecture

In this section, we first explain the overall framework design of ERD, and then introduce each module in detail. The design of ERD mainly focuses on the network software stack on the host side. The RDMA NIC side needs to support normal RDMA UD communication.

4.1 High-level Overview

Our ERD architecture is divided into two main parts: the Host side and the NIC side, as illustrated in Figure 3. In order to minimize hardware modifications on the NIC side and alleviate the cache pressure on RDMA NIC, ERD places the primary design modules on the Host side, while the NIC side is responsible for ensuring basic UD transmission and information exchange between Host and NIC. Following the network software stack of the RDMA protocol, the Host side is divided into three layers: the Application layer, the Fabric layer, and the RDMA-core layer. They are responsible for publishing, maintaining, and transmitting work requests to the NIC.

The Application layer mainly includes point-to-point transmission processes triggered by communication primitives such as All-to-All and All-Reduce for AI training tasks. These processes generate work requests, typically requesting the transfer of data from one cache area to another cache on a specific host or receiving data from another

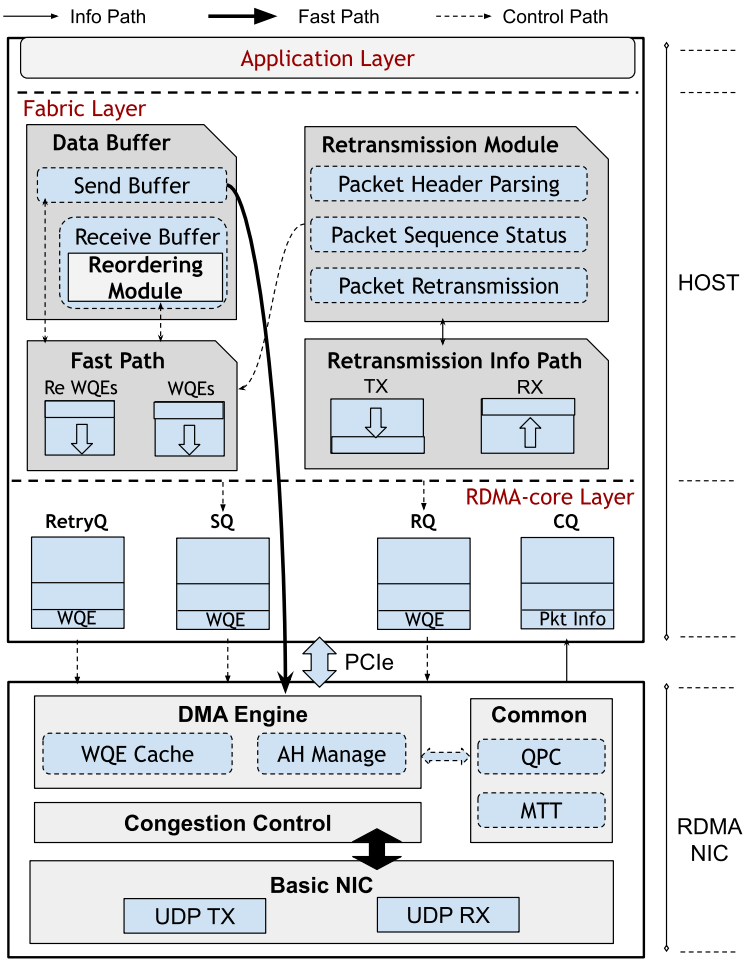


Fig. 3: The overall architecture of ERD. The host side is responsible for work requests and reliability maintenance, and the NIC side is responsible for the generation, sending and receiving of RDMA data packets.

host. The requests are then published to the Fabric layer. The Fabric layer locks the requested cache area, breaks down the work requests into WQEs, and pushes them into the SQ(sender side) or RQ(receiver side) of the RDMA-core layer through the Fast Path module.

Subsequent processing differs for sender and receiver nodes:

On the sender node, the SQ notifies the NIC DMA Engine via doorbell to read the WQE through the PCIe channel. The NIC utilizes modules such as AH management, QPC, and MTT to interpret WQE information, including the memory address of the cache to be sent, data block size, and destination for transmission. The NIC then reads the cache data through the DMA Engine and attaches RDMA information to its

header. Finally, leveraging the Basic NIC, it encapsulates the data into a UDP packet and sends it into the AI data center network.

On the receiver node, the RQ similarly notifies the NIC to read the WQE via doorbell. After parsing the WQE, the NIC obtains information such as the receiving area's cache address and data block size. Upon receiving the data packet, the NIC, using the Basic NIC, decapsulates it and, based on the information provided by the WQE, delivers the packet to the Host side via PCIe, placing it in the correct memory address.

In the event of packet loss, the receiving node detects the lost packet information in the Retransmission Module and notifies the sender through the Retransmission Info Path, prompting it to retransmit the lost data. The sender, based on this information, regenerates the corresponding WQE and pushes it into the RetryQ. Ultimately, the NIC processes and executes the packet retransmission. The functionality of the RetryQ is similar to that of the Send Queue (SQ), but its priority is set to the highest to expedite the reception of retransmitted data by the receiving end.

4.2 Key Design Ideas

In this section, we will address the challenges raised in Section 3. We propose a high-performance RD transmission mechanism, named Efficient Reliable Datagram (ERD). To address these challenges, the following optimization designs are introduced:

1. NACK-Based Retransmission Module: ERD proposes a retransmission module relying on Negative Acknowledgment (NACK). By reducing the frequency of Acknowledgment (ACK) packets, ERD minimizes the overall PCIe latency between the host and NIC caused by reliable processing in RD (**C1**).

2. Dual-Path Processing Mechanism: Regular send/receive operations are directed through the Fast Path of the RDMA protocol stack, while retransmission information exchange is managed via the Retransmission Info Path of the TCP protocol stack. The Fast Path adopts a "best-effort" approach for send/receive operations, leaving loss detection and retransmission triggers to the Retransmission Info Path. This segregation mitigates head-of-line blocking issues in the normal data transmission (**C2**).

3. Efficient Out-of-Order Reception with Reordering Module: a reordering module designed for efficient out-of-order reception is proposed. Leveraging additional packet fields, this module classifies and reorders out-of-order data packets arriving through multiple paths at the Fabric layer. This approach can effectively reduce NIC cache pressure (**C3**).

4.3 NACK-based Retransmission Module

As mentioned in Section 4, the retransmission module of the receiver's ERD needs to detect and record incoming data packets, analyzing information about missing packets. Simultaneously, it is essential to notify the sender about the lost data packet, triggering retransmission, while minimizing notification frequency to reduce PCIe latency.

We have incorporated a NACK-based retransmission module in the Fabric layer on the Host side to meet the aforementioned requirements. The structure of this module

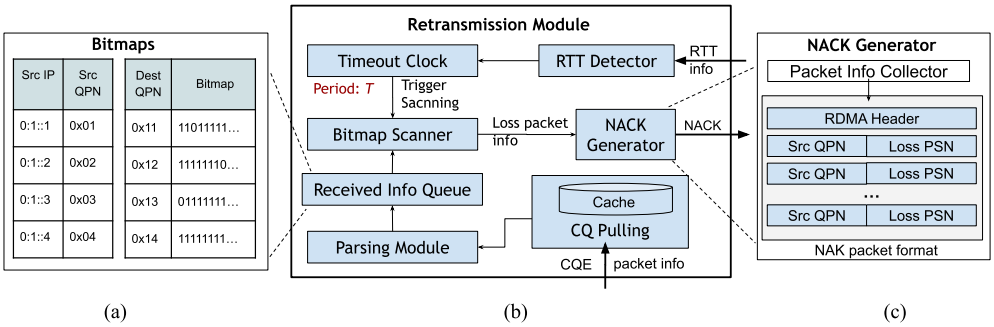


Fig. 4: NACK-based retransmission module. (a) The structure of received info queue. (b) The overall framework of ERD retransmission module. (c) The structure of NACK generator.

is illustrated in Figure 4(b) and primarily operates at the receiver end. The CQ pulling unit is responsible for retrieving packet information from the CQ in the RDMA-Core layer, where each packet's information is integrated into a Completion Queue Element (CQE). Given the continuous arrival of data packets at the receiver, the CQ pulling includes a cache to prevent potential data packet information loss due to insufficient processing speed of CQEs. The Parsing Module decodes the CQE information and records it in the Received Info Queue unit. The Received Info Queue unit uses a Bitmap to record packet sequence information from different nodes (Src IP), different source QPs' number (Src QPN), and different destination QPs' number (Dest QPN), as shown in Figure 4(a). A '1' is set in the corresponding position for each received packet. The Timeout Clock is responsible for periodically triggering a detection signal. Every interval T, this unit sends a trigger scanning signal to the Bitmap Scanner, prompting it to scan all Bitmaps in the Received Info Queue. Upon detecting a missing bit (i.e., a '0' bit) in the Bitmap, the Bitmap Scanner forwards the lost packet information to the NACK Generator. During the scanning process, the Packet Info Collector in the NACK Generator continuously gathers information about lost packets. After the scanning concludes, this information is transformed into NACK packets, as depicted in Figure 4(c). In addition to carrying RDMA header information, the NACK packet also needs to include the source queue pair number(QPN) information and the packet sequence number(PSN) of the lost packet. The converted NACK packets are then sent to the Retransmission Info Path and ultimately received by the sender.

In the above process, a critical issue is determining the triggering period T for the Timeout Clock. If T is too large, the frequency of NACK generation will be low, causing the sender to remain unaware of the receiver's packet reception status for an extended period. On the other hand, if T is too small, the NACK frequency increases, leading to the sender's NIC needing to transmit data to the Host via PCIe at a high rate. This results in a significant amount of additional PCIe latency, negating the advantages of NACK over ACK response packets. Therefore, choosing an appropriate T is essential for the Retransmission Module. Since the setting of T is related to the end-to-end Round-Trip Time (RTT), the RTT Detector unit obtains the maximum end-to-end RTT value from the NIC and transmits it to the Timeout Clock.

Assuming the packet reception rate is P_{speed}, the packet size is set to the Maximum Transmission Unit (MTU) size S_{MTU}, the PCIe processing latency for one NACK packet is T_{pcie}, and the ratio of processing latency for NACK packets to the total packet processing latency is η, the following relationship holds:

$$\eta = \frac{T_{pcie}}{(\frac{P_{speed} \cdot T}{S_{MTU}} - 1) \cdot RTT} \tag{1}$$

In most cases, η is typically required to be no less than 0.996, and T should not be smaller than one Round-Trip Time (RTT). For existing RDMA hardware systems, where P_{speed} is generally in the range of 200 to 400 Gbps, an optimal choice for T is around 2 to 3 RTTs. This range ensures that the sender can promptly address lost data packets while avoiding excessive additional PCIe latency.

4.4 Dual-path Processing Mechanism

In this section, we introduce the dual-path processing mechanism of the ERD, and we will explain the transmission process of normal data packets and retransmitted data packets in end-to-end communication. Figure 5 illustrates the basic framework of the dual-path processing mechanism, including the sender node and the receiver node.

At the sender node, the send buffer module, while transmitting the application layer's WQE to SQ, also caches the WQE. The cached WQE carries two header indices for reference, QPN and PSN. Upon receiving a retransmission command, the send buffer retrieves the corresponding WQE (named Re WQE) based on the command and reissues it to the RetryQ. The Retransmission module is primarily responsible for fetching and parsing NACKs from the retransmission info path, issuing commands (carrying the QPN and PSN of the lost packets) to inform the send buffer about the lost info. To avoid head-of-line block during packet loss, the send buffer module and the Retransmission module depend on the fast path and retransmission info path, respectively, to handle WQE, data packets, or NACK processing.

Fast path: This channel is primarily responsible for handling normal and retransmitted WQEs, as well as facilitating direct memory access (DMA) communication between memory data blocks and the NIC. When no packet loss occurs, WQEs for normal transmission are sent to the SQ based on the original UD-based RDMA protocol. In the event of packet loss, WQEs for retransmission are sent to the RetryQ. Therefore, whether or not packet loss occurs, the fast path always adheres to a 'best-effort' approach and does not wait for acknowledgment from the receiver. Both SQ and RetryQ follow the same steps, where their WQEs are read and identified by the RDMA NIC. Subsequently, the NIC extracts the specified data from memory through DMA and encapsulates it for transmission. However, the RetryQ has a higher priority compared to the SQ, ensuring fast retransmission.

Retransmission info path: This channel is primarily responsible for transmitting NACK packets containing information about lost packets. As NACKs only include metadata for certain packets in the fast path, they have low bandwidth requirements but demand higher reliability. Therefore, the retransmission info path is implemented using TCP/IP for end-to-end connectivity. NACK packets generated by the Retransmission module on the receiver node are transmitted through this channel to the

retransmission module on the sender node, using a distinct set of network protocols from RDMA for end-to-end communication. This segregation ensures that packet loss does not impact normal transmission.

At the receiver node, the application layer initially submits WQEs to the RQ, creating space in memory to await the arrival of external data packets. When a packet is received by the NIC, its payload is transferred to the receive buffer via the fast path, while its header information is extracted to the CQ. The retransmission module reads and analyzes this information further, as described in Section 4.3. In the event of packet loss, the retransmission module generates a NACK, which is transmitted to the sender node via the retransmission info path. It is evident that the receiver node, similar to the sender, employs a dual-path processing mechanism for normal data processing (fast path) and reliability checks (retransmission info path). This ensures end-to-end unobstructed transmission.

The complete data packet transmission and processing workflow can be divided into 5 stages in chronological order, including communication setup, sender transmission, receiver reception and processing, sender retransmission, and resource release.

- Stage 1: Dual-path Communication Setup.
  The sender and receiver establish RDMA and TCP connections. Simultaneously, both ends initiate work requests and split them into WQEs. The sender's WQE is inserted into SQ, providing information such as the address of sending data. The receiver's WQE is placed into RQ, supplying information on the address for storing received data, awaiting the arrival of data packets.
- Stage 2: Sender Transmits Data.
  The WQE in the SQ is initially copied and cached in the send buffer. Subsequently, it is read and analyzed by the NIC through the fast path. The NIC, utilizing the fast path, extracts the data to be transmitted from a specific memory region on the host, encapsulates it into a data packet, and ultimately transmits it into the network.
- Stage 3: Receiver Receives and Processes Packets.
  Upon receiving the data packet, the NIC at the receiver end, on one hand, submits the payload to the receive buffer via the fast path, and on the other hand, extracts the header information of the data packet into the CQ. Subsequently, the retransmission module analyzes, records, and scans these pieces of information, detects which data packets are lost, and encapsulates the lost packet information into a NACK packet. This NACK packet is then sent back to the sender through the retransmission info path.
- Stage 4: Sender Initiates Retransmission.
  Upon receiving the NACK, the sender parses the lost packet information contained within, retrieves the corresponding WQE from the send buffer based on the QPN and PSN. This WQE (Re WQE) is then issued to the RetryQ, undergoing processing with high priority in a manner similar to Stage 2.
- Stage 5: Resource Reclamation.
  Upon receiving all data packets, the receiver notifies the sender. Both ends release their respective cache and communication resources.

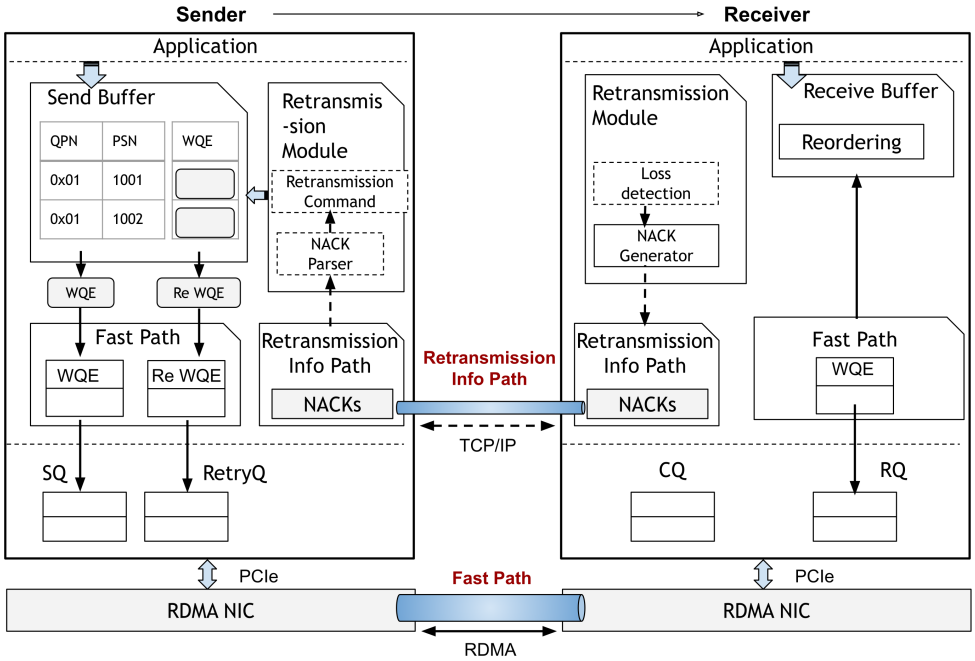


Fig. 5: Dual-path processing mechanism of ERD. Both the sender and the receiver process data and reliability information through dual paths, with each path utilizing RDMA or TCP protocol for data transmission in the network.

The dual-path processing mechanism ensures not only non-blocking and efficient transmission but also the complete transfer of data packets from the sender to the receiver. It is worth noting that if using multi-path RD transmission, the received data at the receiver end may be out of order and cannot be used by the application layer, requiring subsequent reordering processing.

4.5 Packet Format and Reordering Module Design

Multi-path transmission of RD is the most effective way to alleviate hot spots in the network, but it can lead to out-of-order packets at the receiver. To reorder these packets, ERD adds header fields to the original RDMA packets and uses these new fields to achieve high-efficiency reordering in fabric layer.

4.5.1 Packet Format Design

ERD works on top of the RDMA protocol stack's datagram transport, which is defined by the IBTA standard. Therefore, the newly added packet header needs to define data fields in the payload of the RD transport packet as specified by IBTA.

The complete header of an ERD data packet is illustrated in Figure 6. The MAC, UDP, and IP headers form the network layer transport header, responsible for packet forwarding in the Ethernet. The BTH (Base Transport Header), RDETH (RD Extended Header), and DETH (Datagram Extended Header) constitute the RDMA

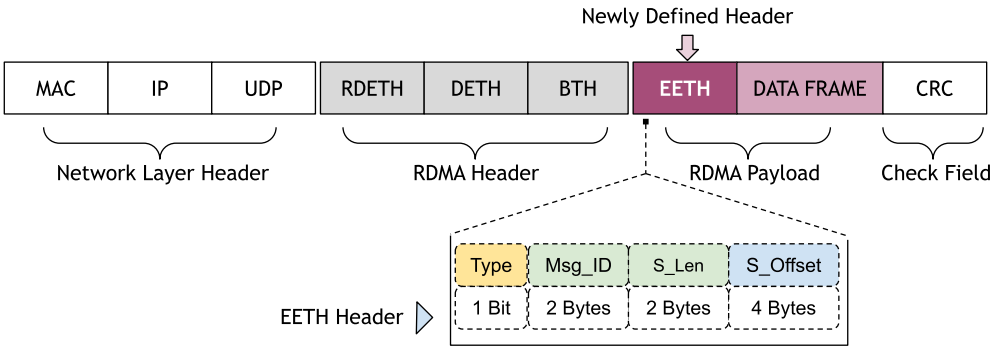


Fig. 6: The header format of the ERD packet. In the payload after the RDMA header, a new EETH field is added for reordering.

transport header, carrying relevant information for RD (Remote Direct) transfers in the RDMA protocol stack. Within the RDMA Payload, ERD introduces the EETH (ERD Extended Header) for packet reordering. This header defines four new fields:

Type: Used to determine the type of the data packet, with a size of one bit. When the packet belongs to a specific long message, this field is set to 1; otherwise, it is set to 0.

Msg_ID: Records the message ID number, occupying 2 bytes. Data packets with the same Msg_ID will be grouped together for sorting.

S_Len: Records the amount of valid payload carried by the DATA FRAME field of the data packet, occupying 2 bytes.

S_Offset: Records the address offset of the payload within the entire message for the data packet, occupying 4 bytes.

The final CRC field serves as a checksum for integrity and correctness verification at the network layer, RDMA layer, and other levels of the data packet.

4.5.2 Reordering Design

The characteristics of RD determine that the receiving node can handle out-of-order packets from different sending nodes and different QPs with the same QP. In order to minimize the computational overhead of sorting, the receiving node of ERD first classifies the incoming packets to identify their sources, and then reorders the packets from the same source. The overall design architecture of the reordering module is illustrated in Figure 7, which includes a packet classification unit and a group classification unit responsible for classification, and receiving buffer and application data units responsible for intra-group reordering.

When a data packet destined for a specific QP reaches the reordering module, the packet classification unit reads the BTH and DETH fields in its RDMA header. Based on these fields, the packet is initially classified, taking into account information such as source IP and source QP, which corresponds to different AI application flows. Subsequently, data packets from the same AI application flow undergo a secondary classification in the group classification unit. This unit reads the Msg_ID field in the

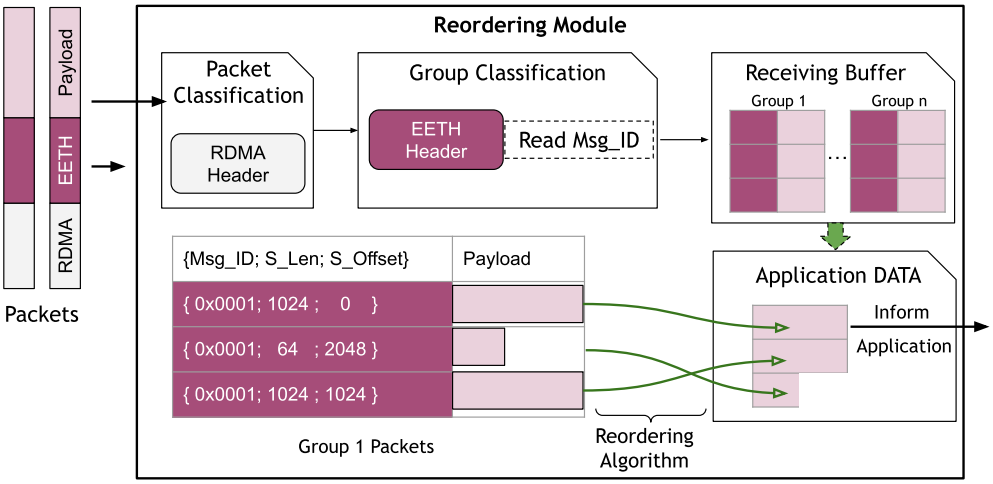


Fig. 7: Reordering module of ERD. Complete, unordered data packets are classified and sorted in the reordering module, and finally form application-identifiable data blocks.

EETH and places packets with the same identifier into the same group within the receiving buffer unit.

After the aforementioned data packet classification process, each packet within a group in the receiving buffer unit originates from the same AI application flow and shares the same message identifier. These packets represent various parts of a specific data block that the AI application intends to transmit before being sent by the sender. However, due to the unordered nature of their arrival in a multipath transmission, these packets require sequential processing upon reaching the receiver. To facilitate ordered handling for upper-layer AI applications, each packet within a group undergoes transformation through a reordering algorithm. The resulting transformation output is then directed to the application data unit, following the specific steps outlined in Algorithm 1.

Within each group, reordering is achieved based on the information in the EETH header of the data packet, which includes the address offset (S_Offset) and the data payload length (S_Len). The receiving buffer unit reads the out-of-order data packets following the first-in-first-out (FIFO) principle. These data packets share the same Msg_ID in the EETH header but differ in S_Offset and S_Len. The receiving buffer unit updates the cache pointer to the relative address in the application data unit indicated by S_Offset and subsequently writes the data payload to the subsequent cache region of that address. Upon completion of the reordering process for all data packets within a group, the application data unit obtains a complete and ordered message data block. In certain transmission scenarios, different messages also require orderly arrangement. In such cases, the application data unit sorts all messages based on the Msg_ID of different groups, thereby accomplishing the ordering of data blocks containing multiple messages.

From the sorting processes of the classification unit and reordering unit within the reordering module, it is evident that the algorithmic complexity of reordering

Algorithm 1: Reordering Algorithm in Receiving Buffer Unit

```
Data: Complete, out-of-order and containing EETH header packets BUFFER
Result: Complete and ordered application data DATA
for EETH_packet in BUFFER do
    read EETH_packet.header;
    group as Msg_id;
end
for group in groups do
    LEN = LEN + S_len;
    pointer = 0x00;
    for EETH_packet in group do
        Move pointer;
        if S_offset == pointer then
            DATA[n] = DATA[n] + EETH_packet.payload;
        else
            Continue;
        end
    end
    pointer = pointer + MTU;
end
n = n + 1;
end
DATA = combine(DATA[0], DATA[1], ... );
if len(DATA) = LEN then
    Return DATA;
else
    Return error;
end
end
```

for N unordered data packets is $O(N)$. The Fabric Layer can efficiently process the unordered packets in the buffer, allowing ERD's reordering module to swiftly handle the unordered data packets. Therefore, compared to the previous reliance on RDMA NIC for reordering, ERD's reordering module avoids the limited cache issues of the NIC and does not introduce significant processing delays.

After completing the reordering process for the data packets, the application data unit checks the integrity of the total payload. If the size of the data block aligns with the sum of the S_Len values of all data packets, it notifies the application layer and uploads the starting address and total length of the data block to the application layer. At this point, the transfer of a data block from the sender's application memory to the receiver's application memory is complete.

5 Evaluation

To thoroughly validate the ERD design solution, we try to address the following questions:

- Does the newly added NACK-based retransmission module and the dual-path processing mechanism enhance the network performance for end-to-end reliable communication? (in Section 5.2)
- What is the optimal number of paths to configure for multipath RD transmission? This parameter setting simultaneously affects the frequency of hot-spot occurrences in the network and the efficiency of the reordering modules in nodes, ultimately impacting the goodput performance and robustness. (in Section 5.3)

5.1 Settings

Environment Setup: We deployed and evaluated ERD on the NS-3 simulation platform, encompassing approximately 7500 lines of code in the fabric and RDMA-core layers. The entire simulation platform ran on four high-performance servers, each equipped with 16 CPUs running at a frequency of 2.2GHz. Additionally, each server was equipped with a 200Gbps Ethernet card, utilizing the Soft-RoCE implementation in the Linux kernel to achieve basic RDMA transport functions. In PCIe settings, all servers configured the bandwidth to 256Gbps, corresponding to PCIe 3.0 $\times$ 32 lanes, with a transmission delay set to 2us. The large-scale data center topology in the simulation platform consisted of five layers, forming a fat-tree structure. The three layers, including Gateway (2 nodes), EoR (16 nodes), and ToR (64 nodes), served as the switch layers, while the Host layer (256 nodes) functioned as the computing service layer. Each Host was configured with at least 4 virtual machines (VMs). Hardware parameters for each layer are provided in Table 1, configured based on commercial data center switching and service equipment. RDMA virtual communication mechanisms such as RC, XRC, ERD were deployed on each VM to measure their performance. To align with the network infrastructure of AI data centers, we used DPDK technology to allow VMs direct access to the underlying NIC interface. Simultaneously, SR-IOV technology was introduced to divide the NIC of each high-performance server into multiple virtual NICs, preventing interference when different VMs forwarded data. Through practical testing, the forwarding latency from a VM to the physical NIC in the simulation platform differed by no more than 500ns from the forwarding latency of a bare-metal server equipped with a Mellanox Connect-X 5 NIC, falling within an acceptable margin of error.

Application Traffic Settings: AI infrastructure involves various types of communication traffic. We simulated All-to-All traffic, which is prone to causing network congestion, by initiating concurrent full-connection communication among all AI applications in all nodes at a specific moment. The traffic from all nodes did not exceed 100Gbps, and the MTU size for each packet was set to 4096 bytes.

Table 1: Hardware Parameters of Network Topology Nodes

Nodes	Throughput (Gbps)	Radix	Delay (us)	Quantity
Gateway	800	16	1.12	2
EoR	400	16	1.01	16
ToR	400	12	0.28	64
Host	100	12	0.06	256
VM	100	8	0.45	1024

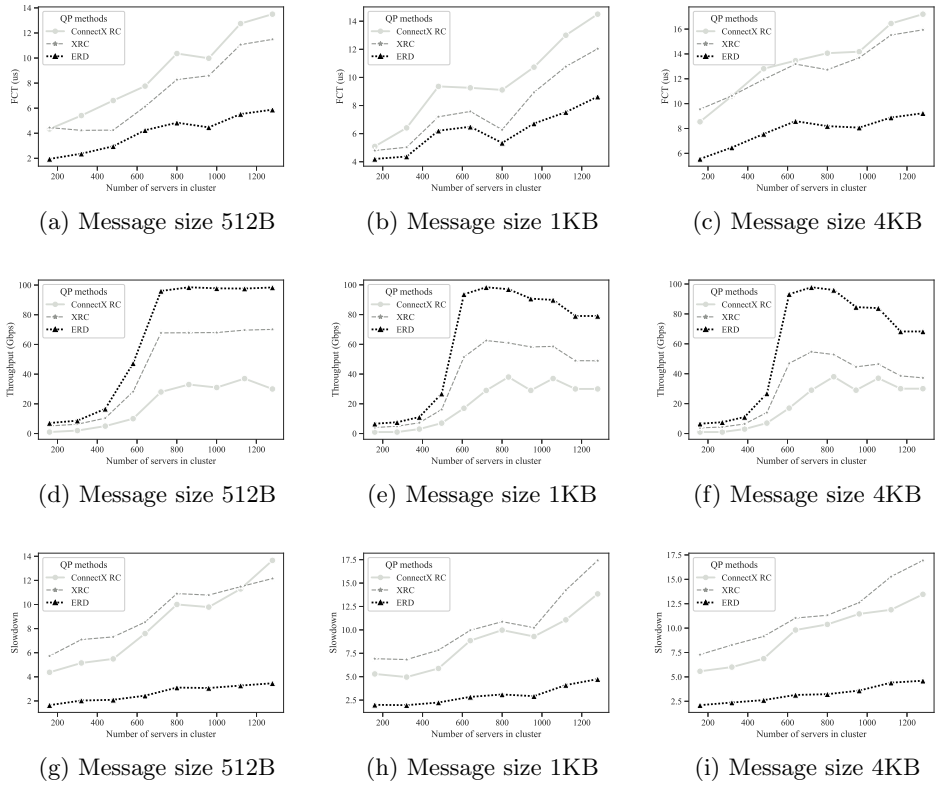
5.2 Communication performance test

We conducted a performance evaluation of ERD compared to existing solutions, benchmarking against current solutions in AI infrastructure, including ConnectX-5 RC and XRC[15]. Network performance was assessed from three perspectives: latency, throughput, and stability. The measured network parameters for the tests included end-to-end average flow completion time (FCT), average throughput, and time slowdown. To avoid randomness, we evaluated the performance separately for different message sizes, specifically 512 bytes, 1 kilobyte, and 4 kilobytes. The results are illustrated in Figure 8.

From the results in Figures 8 (a), (b), and (c), it is evident that in different scenarios with varying message sizes, ERD consistently outperforms the other two QP communication mechanisms in terms of end-to-end flow completion time. As the network scale gradually increases, ERD's latency performance becomes increasingly superior, with the latency reduced by more than 15%. This is attributed to ERD's use of a multipath datagram transmission method, which minimizes congestion in the network. In contrast, the single-path transmission method based on RC is prone to congestion as the network and communication scale expand, leading to a rapid increase in the probability of congestion in the network.

In the case of congestion formation or 'hotspots' at specific nodes in the network, all flows passing through these hotspots experience packet loss. This, in turn, requires the sender to retransmit data packets, further exacerbating the network congestion and ultimately resulting in an increasing flow completion time. While the XRC solution demonstrates slightly better latency performance compared to the original RC scheme, ConnectX RC, its optimization capabilities prove insufficient to overcome the inherent flaws of RC, especially when dealing with message sizes reaching the 4KB level. Consequently, the latency advantage diminishes for XRC at larger message sizes.

At the same time, we also evaluated throughput performance. Figures 8 (d), (e), and (f) illustrate the throughput capabilities of different QP communication mechanisms for various message sizes. It is observed that, with a small number of nodes, the network throughput increases as the network node scale grows for different communication mechanisms. However, after surpassing a threshold of 750 to 800 nodes, the throughput reaches its peak, eventually decreasing with further expansion of the network scale. Nevertheless, the maximum throughput performance of ConnectX RC and XRC is significantly lower than that of ERD by more than 19%. The cause of this phenomenon is also attributed to congestion in the network. On one hand, the use of



(a) Message size 512B (b) Message size 1KB (c) Message size 4KB

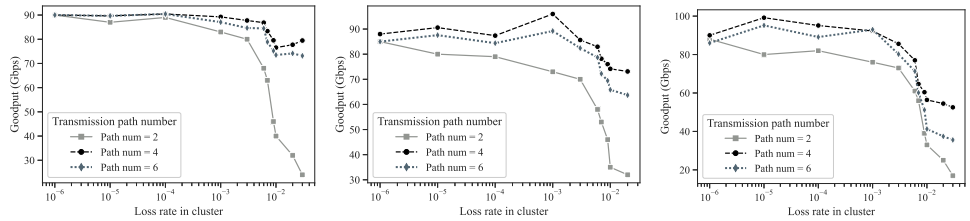
(d) Message size 512B (e) Message size 1KB (f) Message size 4KB

(g) Message size 512B (h) Message size 1KB (i) Message size 4KB

Fig. 8: Performance Test Results of ERD and Two RC Methods for Different Message Sizes. (a), (b), and (c) depict FCT test results, indicating the duration of end-to-end data flow completion, including retransmissions. (d), (e), and (f) represent throughput tests, while (g), (h), and (i) show time slowdown, representing the ratio of worst-case delay to average delay.

RC-based transmission contributes to increased network congestion, leading to a passive reduction in the data packet transmission rate of the network interface cards. On the other hand, the higher number of retransmitted data packets in the connection also occupies effective bandwidth.

Additionally, to verify whether ERD exhibits better stability, we conducted tests and calculations on time slowdown, with results shown in Figures 8 (g), (h), and (i). The results illustrate that as the network scale increases, the slowdown for different QP communication mechanisms consistently grows. This indicates an increasing number of long-tail latency data packets in the network, resulting in gradually unstable network performance and a higher likelihood of packet loss. However, ERD's slowdown is lower compared to the other two RC-based solutions, and its increase is relatively gradual. Even when the network scale exceeds 1000 nodes, ERD does not exhibit a slowdown value exceeding 5 times. The advantage of ERD in performance stability is mainly attributed to the low packet loss rate from multipath transmission and the head-of-line blocking avoidance feature from the dual-channel design. This ensures that normal



(a) 320 nodes (b) 640 nodes (c) 1280 nodes

Fig. 9: Relationship Between Packet Loss Rate and Goodput. The loss rate is manipulated by intermittently disabling ports in the switching equipment, while goodput refers to the traffic consisting of the payload without retransmissions.

communication is not interrupted due to anomalies such as packet loss or timeouts in the network, both at the sender and receiver ends.

Taking into consideration the test results, the performance of ERD excels in terms of transmission delay, throughput, and stability, surpassing the ConnectX-5 RC solution based on RC, as well as the optimized XRC solution, by more than 15%.

5.3 Transmission path configuration

In the RD-type QP communication of ERD with multiple paths, a critical parameter to consider is the configuration of the number of transmission paths. On one hand, it influences the robustness of network transmission; the more paths there are, the more evenly the traffic is distributed throughout the network, enhancing the communication's tolerance to packet loss. On the other hand, it imposes pressure on the reordering module of ERD. Excessive path configurations lead to the reordering module having to handle an excessive amount of out-of-order packets, ultimately resulting in a decrease in goodput. To determine the optimal number of transmission paths, we conducted performance tests on ERD with different numbers of transmission paths in a fat-tree architecture cluster, and the results are illustrated in Figure 9.

The (a), (b), and (c) plots in Figure 9 depict the goodput values measured at different packet loss rates for network cluster sizes of 320, 640, and 1280, respectively. Each curve represents a different configuration of transmission paths. The results indicate that as the packet loss rate increases, the goodput consistently decreases, and the rate of decline varies with different numbers of transmission paths. When the path count is only 2, goodput is significantly affected by the packet loss rate. However, when the path count exceeds 4, goodput begins to decrease. This is attributed to an excessive number of paths exerting undue pressure on the reordering module, ultimately resulting in a phenomenon where goodput declines instead of improving. Based on the comprehensive test results, the optimal number of transmission paths in a fat-tree network architecture is determined to be 3-4.

6 Related Works

In recent years, both the academic and industrial sectors have made numerous attempts to address QP scalability issues and sudden traffic surges in AI infrastructure networks. These efforts encompass both hardware and software solutions.

Hardware Solutions: Various studies have explored hardware solutions, such as StaR[27], which deploys a novel receiving module on RDMA NICs. StaR employs an asymmetric state maintenance strategy to separate connection and retransmission states, thereby enhancing QP scalability. SRNIC[28], utilizing FPGA, introduces a cacheless scheduling strategy on RDMA network cards, shifting the retransmission process to the host side to alleviate card cache pressure. It also leverages FPGA's customizable congestion control module to prevent performance issues arising from sudden network traffic bursts. Additionally, SRD[5] deploys a Datagram-based communication library on Nitro DPU cards, reducing QP numbers under equivalent communication requirements and enhancing transmission stability. These solutions, however, require significant hardware modifications to RDMA NICs, increasing network equipment costs and extending the solution iteration cycle. In contrast, our ERD solution does not entail modifications to RDMA NIC hardware modules, resulting in lower deployment costs within AI infrastructure, facilitating rapid iterations and optimization.

Software Solutions: RoUD[7] adopts a unreliable datagram-based transmission method to reduce the number of QPs created for external communication by applications. XRC[23], an RC-based transmission method supported by IBTA standards[8], reduces QP numbers from $N \cdot P^2$ to $N \cdot P$. It also supports congestion control algorithms on RDMA network cards to mitigate sudden traffic surges. While these solutions mitigate the rate at which QP numbers grow with cluster size to a certain extent, they face challenges as hot spots in the network rapidly increase with a super-large network scale. In contrast, our ERD design significantly reduces the probability of network traffic bursts by supporting multipath transmission for QPs.

7 Conclusion

Scalability issues with QP and congestion problems act as bottlenecks for the hyper-scale deployment of AI infrastructure. We introduce an optimized RDMA QP communication mechanism called ERD, which utilizes RD to maintain a lower number of QPs for equivalent communication requirements. Additionally, ERD employs NACK and a dual-path mechanism as novel reliable safeguards to reduce the extra performance overhead introduced by RD communication. Furthermore, ERD supports multipath transmission through a reordering module, lowering the probability of traffic collisions in the network, mitigating congestion occurrences, and further enhancing end-to-end transmission performance. Experimental results demonstrate the superiority of ERD over other RC-based QP solutions in large-scale AI infrastructure, with a performance improvement of over 15%.

8 Declarations

8.1 Ethical Approval

Not applicable.

8.2 Competing interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

8.3 Authors' contributions

Junliang Wang, Baohong Lin and Mengyu Sun wrote the main part of the manuscript. Junliang Wang, Jiao Zhang and Yongchen Pan are responsible for the design and evaluation. All authors reviewed the manuscript.

8.4 Funding

This work was supported in part by National Key Research and Development Program of China: new data stream heterogeneous processor architecture and computing system (2022YFB4501405), this funding mainly provides data collection and financial support for the research of this paper.

8.5 Availability of data and materials

All data generated or analysed during this study are included in this published article. And the relevant program can be found in [25].

References

[1] Chen G, Lu Y, Li B, et al (2019) Mp-rdma: enabling rdma with multi-path transport in datacenters. IEEE/ACM Transactions on Networking 27(6):2308–2323

[2] Chen Y, Tian C, Dong J, et al (2023) Swing: Providing long-range lossless rdma via pfc-relay. IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS 34(1):63–75. https://doi.org/10.1109/TPDS.2022.3215517

[3] Choi M, Lee S, Kim Y (2022) UD-assisted multi-path transport in RDMA. In: 2022 13th International Conference on Information and Communication Technology Convergence (ICTC), IEEE, pp 127–129

[4] Fent P, van Renen A, Kipf A, et al (2020) Low-latency communication for fast DBMS using RDMA and shared memory. In: 2020 IEEE 36th International Conference on Data Engineering (ICDE), IEEE, pp 1477–1488

[5] Gao YX, Tian C, Chen W, et al (2022) Analyzing and optimizing packet corruption in rdma network. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 37(4):743–762. https://doi.org/10.1007/s11390-022-2123-8

[6] Guo Z, Liu S, Zhang ZL (2019) Traffic control for rdma-enabled data center networks: A survey. IEEE Systems Journal 14(1):677–688

[7] He Z, Chen Y, Hua B (2023) RoUD: Scalable RDMA over UD in lossy data center networks. In: 2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid), IEEE, pp 36–46

[8] Infiniband (2022) Infiniband architecture specification, volume 1. URL https://cw.infinibandta.org/document/dl/8567

[9] Jia C, Liu J, Jin X, et al (2018) Improving the performance of distributed tensorflow with RDMA. International Journal of Parallel Programming 46:674–685

[10] Kang N, Wang Z, Yang F, et al (2022) csRNA: Connection-scalable RDMA NIC architecture in datacenter environment. In: 2022 IEEE 40th International Conference on Computer Design (ICCD), IEEE, pp 398–406

[11] Lee S, Kim Y, Woo H, et al (2021) Efficient user-level multi-path utilization in rdma networks. IEEE Access 9:127619–127629

[12] Li Z, Huang J, Wang S, et al (2024) Achieving low latency for multipath transmission in rdma based data center network. IEEE TRANSACTIONS ON CLOUD COMPUTING 12(1):337–346. https://doi.org/10.1109/TCC.2024.3365075

[13] Ma S, Ma T, Chen K, et al (2022) A survey of storage systems in the RDMA era. IEEE Transactions on Parallel and Distributed Systems

[14] Ma T, Chen K, Ma S, et al (2021) Thinking more about RDMA memory semantics. In: 2021 IEEE International Conference on Cluster Computing (CLUSTER), IEEE, pp 456–467

[15] NVIDIA (2022) Mellanox adapters programmer's reference manual (PRM). URL https://network.nvidia.com/related-docs/usermanuals/Ethernet_Adapters_Programming_Manual.pdf

[16] Park J, Son Y, Yeom HY, et al (2020) Softdc: software-based dynamically connecte transport. CLUSTER COMPUTING-THE JOURNAL OF NETWORKS SOFTWARE TOOLS AND APPLICATIONS 23(1, SI):347–357. https://doi.org/10.1007/s10586-019-02926-0

[17] Park J, Son Y, Yeom HY, et al (2020) SoftDC: software-based dynamically connected transport. Cluster Computing 23:347–357

[18] Pathak AR, Pandey M, Rautaray SS (2020) Approaches of enhancing interoperations among high performance computing and big data analytics via augmentation. CLUSTER COMPUTING-THE JOURNAL OF NETWORKS SOFTWARE TOOLS AND APPLICATIONS 23(2):953–988. https://doi.org/10.1007/s10586-019-02960-y

[19] Shen D, Luo J, Dong F, et al (2023) Enabling distributed and optimal RDMA resource sharing in large-scale data center networks: Modeling, analysis, and implementation. IEEE/ACM Transactions on Networking

[20] Shu J, Chen Y, Wang Q, et al (2020) Th-dpms: Design and implementation of an rdma-enabled distributed persistent memory storage system. ACM Transactions on Storage (TOS) 16(4):1–31

[21] Stewart IA, Erickson A (2018) The influence of datacenter usage on symmetry in datacenter network design. JOURNAL OF SUPERCOMPUTING 74(6):2276–2313. https://doi.org/10.1007/s11227-017-2217-1

[22] Sur S, Jin HW, Chai L, et al (2006) RDMA read based rendezvous protocol for mpi over infiniband: design alternatives and benefits. In: Proceedings of the eleventh ACM SIGPLAN symposium on Principles and practice of parallel programming, pp 32–39

[23] Tang J, Wang X, Dai H (2023) Scalable RDMA transport with efficient connection sharing. In: IEEE INFOCOM 2023-IEEE Conference on Computer Communications, IEEE, pp 1–10

[24] Taranov K, Di Girolamo S, Hoefler T (2021) CoRM: Compactable remote memory over RDMA. In: Proceedings of the 2021 International Conference on Management of Data, pp 1811–1824

[25] Wang J, Huang J (2023) ERD. https://github.com/WangJunliang1/ERD

[26] Wang J, Lin B (2023) RDMA reliability evaluation model for large-scale data center networks. In: 2023 IEEE 3rd International Conference on Computer Communication and Artificial Intelligence (CCAI), IEEE, pp 342–347

[27] Wang X, Chen G, Yin X, et al (2021) StaR: Breaking the scalability limit for RDMA. In: 2021 IEEE 29th International Conference on Network Protocols (ICNP), IEEE, pp 1–11

[28] Wang Z, Luo L, Ning Q, et al (2023) SRNIC: A scalable architecture for RDMA NICs. In: 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23), pp 1–14

[29] Yang H, Xu C, Han Y, et al (2024) An efficient cloud-based elastic rdma protocol for hpc applications. CCF TRANSACTIONS ON HIGH PERFORMANCE

COMPUTING 6(1):45–53. https://doi.org/10.1007/s42514-023-00170-y

[30] Zhang J, Wang Y, Zhong X, et al (2024) Pacc: A proactive cnp generation scheme for datacenter networks. IEEE-ACM TRANSACTIONS ON NETWORKING https://doi.org/10.1109/TNET.2024.3361771

[31] Zhang Z, Liu Z, Jiang Q, et al (2021) Rdma-based apache storm for high-performance stream data processing. INTERNATIONAL JOURNAL OF PARALLEL PROGRAMMING 49(5, SI):671–684. https://doi.org/10.1007/s10766-021-00696-0

[32] Zhuge Q, Zhang H, Sha EHM, et al (2021) Exploring efficient architectures on remote in-memory nvm over rdma. ACM TRANSACTIONS ON EMBEDDED COMPUTING SYSTEMS 20(5, S). https://doi.org/10.1145/3477004