SRC: A Scalable Reliable Connection for RDMA with Decoupled QPs and Connections

Yiren Zhao
Microsoft Research

Ran Shu
Microsoft Research

Yongqiang Xiong
Microsoft Research

Abstract

Using Remote Direct Memory Access (RDMA) is the trend in data centers for achieving high throughput and low latency due to the benefits of hardware offloaded stacks. However, RDMA cannot provide consistently high performance at scale due to the limited RDMA NIC (RNIC) hardware state capacity. We observe that the scalability issue is due to the coupled design of host-RNIC communication channels (queue pairs), and the network connections. In this paper, we propose a novel RDMA transport concept, SRC, which decouples the network connections from queue pairs. SRC introduces a lightweight mapping scheme for efficient forwarding between QPs and connections on the RNIC. Besides, SRC lets software manage the mapping between QPs and connections, which enables inter-application connection sharing, and compatibility with existing RC-based applications. Our results show that SRC can reduce RDMA state size from 146.198 MB to 0.190 MB in a 512-server cluster running RDMA applications.

CCS Concepts

• Hardware → Networking hardware; **• Networks → Network adapters**; **Data center networks**; **• Software and its engineering → Input / output**.

Keywords

Remote Direct Memory Access, Network Hardware, Network Interface Controller, Data Center Networks

ACM Reference Format:
Yiren Zhao, Ran Shu, Yongqiang Xiong. 2025. SRC: A Scalable Reliable Connection for RDMA with Decoupled QPs and Connections. In *9th Asia-Pacific Workshop on Networking (APNET 2025), August 07–08, 2025, Shang Hai, China.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3735358.3735366

1 Introduction

Remote Direct Memory Access (RDMA) has become the cornerstone of modern data center networks due to its high throughput, low latency, and low CPU overhead provided by the hardware offload transport to RDMA Network Interface Cards (RNICs). It has been widely adopted in current applications including Online Transaction Processing (OLTP) [4, 6, 8, 16], in-memory key-value stores [15, 17, 25], distributed file systems [1, 12, 27], and distributed machine learning [9, 19].

The RNIC manages the hardware offloaded network stack and thus achieves extremely high performance. However, storing the stack states is very challenging. The RNIC only has several MBs of on-chip SRAM using current technology [14] which is far less than the up to GB level state size [26]. Thus, RNIC stores states in host memory and uses on-chip memory as a cache. However, at large scale, frequent RNIC cache misses lead to performance degradation. This is a well-known issue of RDMA for decades and can greatly impact application performance at scale [5, 8, 18, 23, 26].

RDMA provides both connection-oriented and datagram-based communications in both reliable and unreliable mode. Among these, Reliable Connection (RC) is the most desired and widely used due to its offloaded reliable delivery, and support for full set of RDMA operations. To fully exploit RDMA's performance advantages, applications often maintain a large number of concurrent connections, which in turn require substantial on-NIC states. Hosts communicate with RNICs through Queue Pairs (QPs), and RC communication is established by connecting a pair of QPs between two nodes. On the other hand, as the QPs are separately owned by different processes running on the same node, it requires a pair of QPs between each pair of processes to enable the full mesh communication between processes. Assuming a cluster of N nodes where each node runs P processes, it requires $P \times P \times (N - 1)$ QPs on each node for a fully connected communication. Moreover, to fully make use of the multi-core CPU capability, RDMA applications usually setup dedicated QPs for each thread to avoid lock overhead for inter-thread synchronization which makes the QP scaling requirement even higher.

Many approaches have been proposed to reduce the number of required QPs thus mitigate RDMA scalability issues. Solutions including sharing RC connections [8, 18, 23], connection grouping to control concurrency [5], proposing new reliable connected QPs with lower scale requirements including XRC [11] and DCT [7], using UD instead of RC [15, 16], reducing on-NIC states [26], and storing states at lower load side [24]. However, all existing solutions have their drawbacks. Software-based solutions (sharing RC [8, 18, 23], connection grouping [5], and using UD [15, 16]) introduce extra CPU overhead which counteract the RDMA benefits. XRC cannot scale well to large clusters, and DCT suffers from frequent connection switching and increased latency, often requiring a large pool of QPs to mitigate runtime delays [16, 21]. Architectural optimizations [26] reduced on-NIC state size, but cannot compress the QPC itself, which remains the dominant contributor to per-connection state. Finally, approaches that store states on the other side require asymmetric number of connections [24].

We try to look at this problem from another aspect by asking: *Is such a high number of QPs (connections) required by nature?* We find that the root cause lies in the tightly coupled design of QP and network connection in RDMA RC. Each QP is treated as an individual pair of host-NIC channel and a network connection. Such redundancy wastes valuable RNIC on-chip storage space which severely limits RDMA scalability. QPs are originally designed as communication channels between the CPU and peripherals, while connections are meant to ensure reliable communication between nodes. If their roles are distinct by design, *why not separate QPs and connections and let them work as originally intended?*

We propose a novel RDMA transport concept, Scalable Reliable Connection (SRC) which has decoupled network connections and QPs. Applications use SRC QPs as the channel to communicate with RNIC. The connections are established only between RNICs rather than directly between applications. By this design, multiple applications can share the same connection, and an application can communicate with multiple peers using the same QP. An on-NIC forwarder switches packets between QPs and connections in both directions.

Separating connections from QPs introduces extra states for managing end-to-end connectivity and securing RDMA communication. To eliminate extra states overhead on RNIC, SRC let RNIC driver manages the states of mappings between QPs and connections in software. SRC library specifies the required connection in the request with the connection index for request sending. SRC NIC only stores minimal connection states in an packed array to enable fast lookups. For the packet receiving path, mapping from connection to QP is only required for 2-sided operation (SEND/RECV). To eliminate table lookup overhead, SRC slightly modifies the RDMA packet format to carry the destination QP number.

SRC supports two connection management modes, depending on whether connections are managed by the application or by the RDMA driver, allowing developers to balance compatibility and optimization flexibility. The SRC managed mode is designed for RDMA driver, which provides compatibility as existing RC interfaces to minimize application modifications. SRC library stores the user QP to SRC QP and connection mappings so that additional data operations overheads are minimized. For connection establishment/termination, states in SRC driver helps identify the existence of connections and help setting up the mappings between user QPs and connections. The application managed mode gives applications the flexibility to deeply optimize for SRC system, but it does not allow sharing connections across applications.

Our evaluation shows that SRC significantly reduces RDMA state size from 146.198 MB to 0.190 MB for a 512-server cluster running RDMA applications. While DCT reduces state size to 0.014 MB per node, it suffers from performance degradation compared to RC due to frequent connection switching. In contrast, SRC achieves substantial state reduction, while further improving upon RC by delivering better performance without such overhead.

2 Background and Motivation

This section outlines the foundational principles of RDMA communication and the challenges faced in scaling RDMA systems, particularly focusing on the limitations of current coupled QP-to-connection design. We then motivate the need for decoupling QPs from connections as a more scalable approach.

2.1 RDMA Communication and QP Management

Remote Direct Memory Access (RDMA) has emerged as a pivotal technology for enabling high-speed, low-latency communication in modern data centers [1, 4, 6, 8, 12, 15–17, 25, 27], cloud [3, 10], and AI [9, 19]. By bypassing traditional networking stacks and allowing direct memory access between hosts, RDMA minimizes CPU overhead and achieves high throughput.

RDMA Primitives. RDMA supports two types of communication, one-sided and two-sided operations. The WRITE, READ and ATOMIC are one-sided primitives, which allows directly access memory on a remote node without involving the remote node's CPU. In contrast, the SEND/RECV is two-sided primitive, both the sender and receiver participate in the data transfer. Specifically, when the sender issues a SEND operation, the receiver must have already posted a matching RECV request to hold the incoming data.

Queue Pair (QP). RDMA hosts communicate through Queue Pairs (QPs), which consist of a Send Queue (SQ) and a Receive Queue (RQ). The SQ manages outgoing Work Requests (WRs), such as SEND, WRITE, and READ operations, while the RQ handles RECV operation to receive incoming SEND requests. Each QP maintains state information that dictates how RDMA operations are executed, including queue status, queue pointers, connection state (for connection-oriented communications), memory protection information, etc.

Transport Modes. RDMA supports both connection-oriented and datagram-based communications in reliable and unreliable mode, including Reliable Connection (RC), Reliable Datagram (RD), Unreliable Connection (UC), and Unreliable Datagram (UD). RC is the most widely used transport type in RDMA deployments due to its strong guarantees of reliability, in-order message delivery, and support for a full set of RDMA operations [5, 18]. In RC, each QP must connect to a single remote QP before communication. UC also follows a one-to-one connection model but does not provide guarantee reliable transmission. Application needs additional computation overhead to deal with communication reliability. RD[1] and UD are datagram QPs which allow one-to-many communication models, enabling communicating with multiple remote endpoints using a single shared QP. However, application needs to handle message dispatching with extra processing overhead.

2.2 RNIC Scalability Issue

Today's RDMA applications require a large number of QPs to fully exploit the performance benefits of RDMA. Figure 1(a) shows the conceptual diagram of the required RC QPs. Each process has its own group of QPs. As a QP is dedicated to a single connection, a process needs to set up multiple QPs to communicate with multiple processes even on a single remote node. The total number of QPs required scales quadratically as $P \times P$ for each pair of nodes. Assuming a cluster with N nodes, the total number of QPs required is $P \times P \times (N - 1)$. Moreover, a process may allocate per-thread QPs to avoid synchronization overhead and fully make use of the multi-core CPU capability. Given the high number of CPU cores per node today, the number of required QPs can be extremely large.

[1]RD is not supported by any commodity RNICs. AWS has developed SRD which has the similar concept as RD but with some minor difference in usage [20].

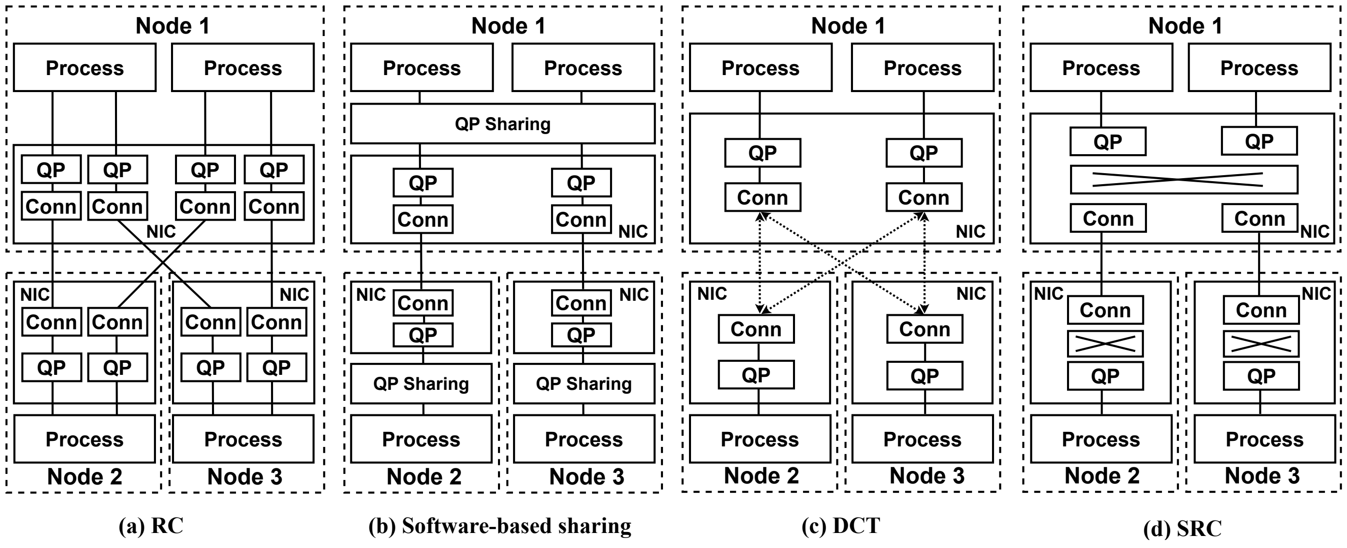

Figure 1: Comparison of different RDMA reliable connection solutions.

Assuming each node runs 20 processes, even a cluster with 32 nodes requires 12,400 of QPs.

The QP states (and other states) are stored and managed by the RNIC for the offloaded network stack. However, due to inherent hardware limitations, RNIC cannot provide consistent high performance at scale. Current chip manufacturing technologies and cost considerations limit the expansion of on-chip memory in RNICs. The RNIC is typically equipped with a few MBs of on-chip SRAM [14], which is far less than the up to GB level state size [26]. Expanding memory capacity with off-chip DRAM is theoretically possible but impractical due to the long DRAM access latency. RNIC processing pipelines need to be completely redesigned to adapt to the long memory access latency and the required circuit size will be greatly increased.

As a result, RNICs store states in the host's DRAM and use on-chip SRAM as a cache [22]. However, as the number of QPs increases, the limited SRAM size leads to frequent cache misses. When a cache miss occurs, the RNIC must fetch the required state from the host DRAM over the PCIe bus which has much higher latency than on-chip SRAM. The RNIC pipeline is hard to absorb all the pending requests. Thus, frequent cache misses cause significant performance degradation of RNICs. The RNIC scalability issue is well known and has been widely studied [5, 8, 14–16, 18, 23, 24, 26].

To mitigate RDMA scalability issues, several solutions have been proposed to reduce the number of required QPs. Software-based approaches aim to reduce QP overhead by sharing QPs across multiple processes/threads [8, 18, 23]. As shown in Figure 1(b), the software-based QP sharing can reduce the number of required QPs. However, sharing QPs introduce contention and software overheads are paid for multiplexing/demultiplexing. Prior work reports that QP sharing reduces per-core one-sided READ throughput by up to 5.4× due to contention and synchronization overheads [16].

RNIC vendors have also introduced new reliabile connection transport modes, eXtended Reliable Connection (XRC) [11] and Dynamic Connected Transport (DCT) [7], to improve QP scalability. XRC optimizes QP usage by allowing a target QP to distribute incoming messages to multiple processes. In the previously mentioned fully connected cluster setting, this capability reduces the per-node QP requirement to $P \times (N-1)$. However, XRC cannot scale well to large clusters [21]. DCT further reduces the QP count by dynamically establishing and maintaining a single reliable QP that connects to multiple remote QPs as needed, allowing each process to maintain only P QPs. The DCT conceptual model is shown in Figure 1(c). However, when communicating with multiple destinations, it frequently switches connections, leading to increased latency and bandwidth inefficiency [16]. Additionally, as a DCT QP can only be connected with a single remote QP, applications often manage a large number of DCT QP pool to reduce waiting time. Thus, the number of required QPs is normally much more than P.

Some approaches try to mitigate the scalability issue with software or hardware enhancements. ScaleRPC [5] reduces RNIC cache misses through connection grouping which limit the concurrency of active connections. Again, software overheads are introduced and the optimizations are application specific. StaR [24] enables storing RDMA states of a connected pair on a single side, but its effectiveness highly depends on asymmetric requirements on number of connections. SRNIC [26] optimizes on-NIC states management through RNIC architecture improvements, however, it cannot reduce the size of connection states.

Other approaches abandon RC QPs but use UD QPs [15, 16] or even not using RDMA [14]. Such a design requires moving the connection and reliability management to host software which introduces high software overhead. Moreover, UD lacks support for one-sided RDMA operations and message size is restricted by not exceeding MTU. Such limitations further increase the software overhead.

2.3 Motivation

The scalability limitations of existing connection-oriented RDMA transports stem from the tight coupling between QPs and connections. It is essential to decouple QPs from connection semantics, and let them handle their original design purposes. QP, by its original concept, is the communication channel between CPU and peripherals like normal NIC [13] and NVMe SSD [2]. The primary goal of supporting multiple QPs is to avoid synchronization between CPU cores and improve the software efficiency. Decoupling QPs from connections allows each of them to work independently as their originally intended. The decoupling can reduce the total number of QPs and connections required, thereby decreasing the storage space required on the NIC and significantly mitigating the RDMA scalability issue. However, achieving this decoupling presents several design challenges that must be carefully addressed.

- **Challenge #1: Minimize Additional States.** Although decoupling QPs and connections can reduce the required number of QPs and connections, extra states are required to manage the end-to-end connectivity and protect RDMA against malicious users. To maintain consistent high performance, the size of these additional states should be small enough to be stored in RNIC's on-chip SRAM.
- **Challenge #2: Efficient State Lookup.** State lookup is required when RNIC maps a request from a QP to its corresponding connection and vice versa when receiving packets from the network. To support fast lookups and meet high performance requirements, efficient system architectures and data structures must be designed.
- **Challenge #3: Efficient and Friendly Abstraction.** Introducing new types of QPs like XRC [11] and DCT [7] come with new application programming abstractions. The new abstraction must be efficient for application programming. Moreover, there are already many RDMA applications designed with the current RDMA RC conceptual model. Reducing programming modifications is also a challenge in the design.

3 Design

The issues of native RDMA outlined in the previous section stem from the per-QP connection tracking burden on the limited on-NIC memory, which severely constrains scalability in large-scale deployments. This section presents SRC, a novel RDMA architecture that fundamentally redesigns RDMA by decoupling QPs from network connections. The following subsections provide a detailed description of SRC's design and architecture.

3.1 Architectural Overview

As SRC decouples the QP and connection, it redefines their respective roles. The following explains how each component functions independently under the new design.

SRC QP. A SRC QP serves as a communication channel between the host and the RNIC. Specifically, the SRC QP acts as an interface for submitting work requests (WRs) and receiving completions, without storing any connection-specific information (e.g., address or remote QP information). Like in RC, multiple SRC QPs can be created by applications to support multi-threaded execution and optimize performance.

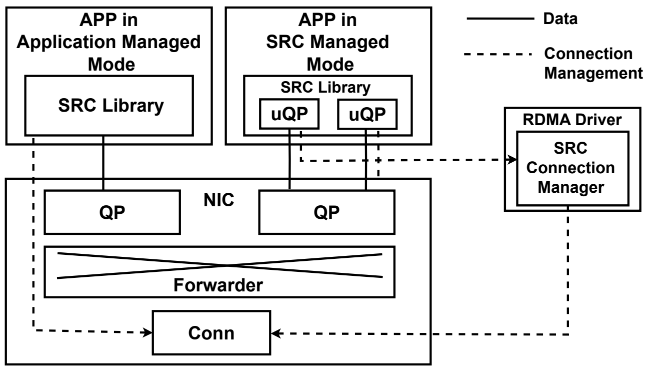

Figure 2: SRC Architecture.

Connections. A connection refers to a network channel between two RNICs, responsible for reliable RDMA data transmission. Multiple connections can exist between the same pair of endpoints. The number of connections is determined by software and can be configured based on application requirements.

Figure 2 illustrates the architecture of SRC, which includes both hardware and software modifications to RDMA. On the RNIC, SRC QPs and connections are decoupled and connected through the forwarder that selects the appropriate connection for each SRC QP and vice versa. The SRC QPs data path is exposed to the SRC library to support kernel-bypassing I/O. Both SRC QPs and connections expose management interfaces to the RDMA driver, which handles queue initialization and release, as well as connection establishment and termination. Applications manage SRC QPs through a management API provided by the SRC library.

SRC provides two operation modes for connection management: SRC managed and application managed. Supporting both modes allows developers to trade off compatibility and flexibility.

The SRC managed mode is designed to be highly compatible with traditional RC. In this mode, applications use user QPs (uQPs) as handles for the SRC managed connections. The uQP follows a conceptual model similar to that of RC. From the application's perspective, each uQP behaves same to a standard RC QP, where the application assumes a one-to-one mapping between a local QP and a remote QP, as well as between a QP and a connection. Internally, however, a uQP does not directly map to a physical connection in SRC. Instead, it is an abstraction that the RDMA library translates into a connection. Multiple uQP instances can share the same SRC QP with minimal overhead. The mapping between uQP and SRC QP is managed by the SRC library.

The application managed mode exposes a simpler interface, where applications directly manage both SRC QPs and connections. This mode gives applications the flexibility to co-optimize application and RDMA system. For example, eliminating the uQP to QP mapping overhead. However, it lacks the ability to share connections across applications, which may limit scalability under extreme scale.

3.2 SRC RNIC

With the decoupling of QP and connection, the SRC QP no longer contains any connection-related states. A mechanism is needed to determine which connection to use for the WRs, and to manage connection states and authentication. To achieve such management, connection contexts must be stored and maintained. The key question is whether the RNIC should store and maintain such information. Although SRC successfully reduces memory usage compared to RC QPs by decoupling QPs from connections, the mapping between QPs and connections is still considerably large. As the number of connections increases, the state requirements inevitably exceed what the RNIC can efficiently handle. In addition, managing connection states on the RNIC introduces unnecessary complexity. The RNIC needs to update the states at connection establishment and termination, requiring additional hardware logic. Instead, this complexity is better handled in software, which offers greater flexibility in managing connection reuse and policy enforcement.

Efficient connection lookup is crucial, as RNICs run at extremely high speeds. The RNIC needs to map requests from SRC QPs to their corresponding connections, and vice versa when processing incoming network packets. The remote address is not a good connection indicator due to its sparsity. Instead, SRC packs all connection states into an array and uses the index to locate a connection. Specifically, when the RDMA driver calls RNIC for a new connection, RNIC returns an idle connection ID. The RDMA driver then uses this connection ID to establish a connection with the remote side. When a SRC QP's WR needs to access its corresponding connection, it uses the connection ID to directly locate the entry in the connection state table, ensuring constant-time (O(1)) retrieval.

3.3 Connection Management and SRC QP

SRC provides two connection management modes. Here, we mainly introduce our design for nearly compatible RC QPs, whose connections are managed by the SRC Connection Manager in RDMA driver. The SRC library includes special designs for both connection establishment and request processing. For application managed mode, the main difference is the absence of connection management related features in SRC driver and library. The rest parts are the same.

When an application requests to connect a local uQP to a remote uQP, it calls the SRC library. The SRC library forwards the call to the RDMA driver as the connection management is done in the kernel driver. The SRC Connection Manager first checks whether an existing connection to the specified remote address is already established. If not, the SRC Connection Manager informs the RNIC to establish a new connection. Once the connection is established, or if it already exists, the connection information and remote uQP are stored in the local uQP's metadata for fast mapping in request processing.

When a user posts a WR to an uQP as using RC, the RDMA library maps the WR in the appropriate SRC SQ and connection before submitting the request to the RNIC. The translation process relies on metadata stored within the library for each RC QP.

For SRC QP design, we do not add additional Completion Queue (CQ) mapping for scalability improvements. CQ management remains unchanged from the existing RDMA conceptual model. The main reason for this is that the QP to CQ mapping is managed by applications. Applications usually map multiple QPs to the same CQ for efficient completion dispatching and processing. Adding another layer of mapping not only introduces extra overhead but also harms the existing software efficiency optimizations. In existing RC QPs, the mapping between QP and CQ is stored in the QPC and cached by the NIC for fast lookup. In contrast, in SRC, NIC does not maintain such information in the QPC to save storage space. Instead, we let packets carry the selected CQ number.

The RDMA responder for SRC have different workflows for one-sided and two-sided operations. For one-sided operations, since the connection alone is sufficient for providing reliable delivery, QP is not involved at responder side for one-sided operations. For two-sided operations, handling the RECV operation is more complicated. We choose not to share the RQ as we do not share CQ, for similar reasons. First, for users who care about scalability, RDMA already provides SRQ to reduce the number of required RQs. Second, although we can reduce the number of RQs by sharing, the RNIC needs to map the incoming SEND operation to the corresponding uQP. As a result, RNIC still have to maintain states at the scale of the number of uQPs to avoid head-of-line blocking.

3.4 Discussion and Open Questions

Although SRC can greatly mitigate the RDMA scalability issue, there are still several unsolved problems and opportunities for further improvement. We list them here and discuss possible approaches.

Security. SRC slightly modifies the RDMA system architecture, and the changes introduce new security considerations. First, denial-of-service (DoS) attacks may occur when multiple uQPs share the same SRC QP or connection. A malicious user can intentionally stall its own WRs, for example by not polling completions or exhausting buffer resources. This stalls the shared queue and indirectly denies service to other legitimate users sharing the same resource. Second, because connection and QP mappings are managed in software, a malicious application could potentially manipulate these mappings using a modified SRC library. For example, it may redirect its uQPs to unauthorized connections or SRC QPs, impersonating another application or violating RDMA memory access boundaries. These threats are mitigated in traditional RC, where all connection bindings are enforced in hardware. Although SRC currently does not provide strong defenses against these attacks, exploring lightweight isolation and mapping validation mechanisms is an important direction for future enhancement.

Head-of-Line Blocking. The use of shared QPs and connections in SRC may introduce head-of-line (HoL) blocking risks. When multiple uQPs are mapped to the same SRC QP, a work request at the front of the send queue may be stalled due to flow control, network congestion, or RNIC resource constraints. In such cases, all subsequent requests in the queue are also delayed, including those from other uQPs that target uncongested destinations. Similarly, when several SRC QPs share the same connection, congestion

Nodes (N)	RC (MB)	XRC (MB)	DCT (MB)	SRC (MB)
32	8.869	0.443	0.014	0.019
128	36.335	1.817	0.014	0.053
512	146.198	7.310	0.014	0.190

Table 1: Memory usage per node for different RDMA transport models (20 threads per node).

on that connection can pause the transmission of all packets assigned to it. Addressing such head-of-line blocking is an important direction for future system design.

QoS model. The original RDMA RC transport uses QP as the basic unit of QoS management. However, as we decouple the QPs and connections, the QoS management unit becomes QPs and connections. For example, multiple QPs have a fair sharing of bandwidth in existing RDMA. However, in SRC, some of the QPs can share the same connection and the current SRC is unable to provide fair sharing among QPs. In future enhancement of SRC, we should either design schemes to preserve the original QoS model or propose new QoS models with application requirement analysis.

Compatible with RNIC architectures. Existing RNIC architecture is highly optimized for the current RDMA workflow [26]. Although SRC achieves a simple architecture for decoupling QPs and connections on the RNIC, integrating these modifications into existing commodity RNICs with minimal changes without affecting the current RDMA workflow remains one of the key problems to be solved in the future.

Application RNIC co-design. ScaleRPC [5] and Flock [18] provide specialized application-aware software optimizations for efficient RPC over RDMA. The current SRC is designed as a general scalability enhancement for RDMA RC. However, it is possible to achieve greater performance gains with Application RNIC co-design. This is one of the future directions we plan to explore.

4 Preliminary Results

Table 1 presents the memory consumption per node for different RDMA transport models as the number of nodes increases. RC requires the highest memory usage due to its per-QP connection model, leading to quadratic growth in memory consumption as the cluster scales. For example, with 512 nodes, RC consumes 146.198 MB per node, compared to only 8.869 MB for 32 nodes, illustrating the severe scalability limitations of RC. XRC reduces memory overhead by allowing multiple processes to share a single QP per destination node, thereby lowering memory consumption to 7.310 MB per node with 512 nodes, a 20 $\times$ reduction compared to RC. While DCT reduces state size to 0.014 MB per node, it suffers from performance degradation compared to RC due to frequent connection switching. In contrast, SRC achieves substantial state reduction, while further improving upon RC by delivering better performance without such overhead. SRC significantly reduces RDMA state size, decreasing memory consumption from 146.198 MB (RC) to just 0.190 MB per node for a 512-server cluster.

5 Conclusion

We propose a novel RDMA transport concept, Scalable Reliable Connection (SRC), which has decoupled network connections and QPs to improve RDMA scalability. By introducing a software-managed multiplexing mechanism, SRC reduces RNIC state overhead and eliminates the need for per-QP connection tracking on the NIC. SRC employs a handle-based connection resolution mechanism, removing expensive lookup operations and minimizing processing overhead. Our evaluation shows that SRC can reduce RDMA states size from 146.198 MB to 0.190 MB for a 512-server cluster running RDMA applications.

Acknowledgments

The authors gratefully acknowledge the anonymous reviewers for their valuable and constructive comments.

References

[1] 2017. Crail: A Fast Multi-tiered Distributed Direct Access File System. In *Proceedings of the 2017 IEEE International Conference on Big Data (Big Data)*. https://github.com/zrlio/crail

[2] 2022. NVM Express Base Specification 2.1. https://nvmexpress.org/wp-content/uploads/NVM-Express-Base-Specification-Revision-2.1-2024.08.05-Ratified.pdf.

[3] Wei Bai, Shanim Sainul Abdeen, Ankit Agrawal, Krishan Kumar Attre, Paramvir Bahl, Ameya Bhagat, Gowri Bhaskara, Tanya Brokhman, Lei Cao, Ahmad Cheema, et al. 2023. Empowering azure storage with RDMA. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 49–67.

[4] Haibo Chen, Rong Chen, Xingda Wei, Jiaxin Shi, Yanzhe Chen, Zhaoguo Wang, Binyu Zang, and Haibing Guan. 2017. Fast in-memory transaction processing using RDMA and HTM. *ACM Transactions on Computer Systems (TOCS)* 35, 1 (2017), 1–37.

[5] Youmin Chen, Youyou Lu, and Jiwu Shu. 2019. Scalable RDMA RPC on reliable connection with efficient resource sharing. In *Proceedings of the Fourteenth EuroSys Conference 2019*. 1–14.

[6] Yanzhe Chen, Xingda Wei, Jiaxin Shi, Rong Chen, and Haibo Chen. 2016. Fast and general distributed transactions using RDMA and HTM. In *Proceedings of the Eleventh European Conference on Computer Systems*. 1–17.

[7] Diego Crupnicoff, Michael Kagan, Ariel Shahar, Noam Bloch, and Hillel Chapman. 2012. Dynamically-connected transport service.

[8] Aleksandar Dragojević, Dushyanth Narayanan, Miguel Castro, and Orion Hodson. 2014. FaRM: Fast remote memory. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*. 401–414.

[9] Adithya Gangidi, Rui Miao, Shengbao Zheng, Sai Jayesh Bondu, Guilherme Goes, Hany Morsy, Rohit Puri, Mohammad Riftadi, Ashmitha Jeevaraj Shetty, Jingyi Yang, et al. 2024. RDMA over ethernet for distributed training at meta scale. In *Proceedings of the ACM SIGCOMM 2024 Conference*. 57–70.

[10] Yixiao Gao, Qiang Li, Lingbo Tang, Yongqing Xi, Pengcheng Zhang, Wenwen Peng, Bo Li, Yaohui Wu, Shaozong Liu, Lei Yan, et al. 2021. When cloud storage meets RDMA. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. 519–533.

[11] InfiniBand Trade Association. 2009. InfiniBand Architecture Specification, Release 1.2.1, Annex A14: Extended Reliable Connected Transport Service. https://www.infinibandta.org

[12] Nusrat Sharmin Islam, Md Wasi-ur Rahman, Xiaoyi Lu, and Dhabaleswar K Panda. 2016. High performance design for HDFS with byte-addressability of NVM and RDMA. In *Proceedings of the 2016 International Conference on Supercomputing*. 1–14.

[13] EunYoung Jeong, Shinae Wood, Muhammad Jamshed, Haewon Jeong, Sunghwan Ihm, Dongsu Han, and KyoungSoo Park. 2014. mTCP: a highly scalable user-level TCP stack for multicore systems. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*. 489–502.

[14] Anuj Kalia, Michael Kaminsky, and David Andersen. 2019. Datacenter RPCs can be general and fast. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*. 1–16.

[15] Anuj Kalia, Michael Kaminsky, and David G Andersen. 2014. Using RDMA efficiently for key-value services. In *Proceedings of the 2014 ACM Conference on SIGCOMM*. 295–306.

[16] Anuj Kalia, Michael Kaminsky, and David G Andersen. 2016. FaSST: Fast, scalable and simple distributed transactions with Two-Sided RDMA datagram RPCs. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. 185–201.

[17] Christopher Mitchell, Yifeng Geng, and Jinyang Li. 2013. Using One-Sided RDMA Reads to Build a Fast, CPU-Efficient Key-Value Store. In *2013 USENIX Annual Technical Conference (USENIX ATC 13)*. 103–114.

[18] Sumit Kumar Monga, Sanidhya Kashyap, and Changwoo Min. 2021. Birds of a feather flock together: Scaling rdma rpcs with flock. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*. 212–227.

[19] Kun Qian, Yongqing Xi, Jiamin Cao, Jiaqi Gao, Yichi Xu, Yu Guan, Binzhang Fu, Xuemei Shi, Fangbo Zhu, Rui Miao, et al. 2024. Alibaba HPN: A data center network for large language model training. In *Proceedings of the ACM SIGCOMM 2024 Conference*. 691–706.

[20] Leah Shalev, Hani Ayoub, Nafea Bshara, and Erez Sabbag. 2020. A cloud-optimized transport protocol for elastic and scalable hpc. *IEEE micro* 40, 6 (2020), 67–73.

[21] Hari Subramoni, Khaled Hamidouche, Akshey Venkatesh, Sourav Chakraborty, and Dhabaleswar K Panda. 2014. Designing MPI library with dynamic connected transport (DCT) of InfiniBand: early experiences. In *International Supercomputing Conference*. Springer, 278–295.

[22] Sayantan Sur, Abhinav Vishnu, H-W Jin, DK Panda, and W Huang. 2005. Can memory-less network adapters benefit next-generation infiniband systems?. In *13th Symposium on High Performance Interconnects (HOTI'05)*. IEEE, 45–50.

[23] Shin-Yeh Tsai and Yiying Zhang. 2017. LITE kernel RDMA support for datacenter applications. In *Proceedings of the 26th Symposium on Operating Systems Principles*. 306–324.

[24] Xizheng Wang, Guo Chen, Xijin Yin, Huichen Dai, Bojie Li, Binzhang Fu, and Kun Tan. 2021. StaR: Breaking the scalability limit for RDMA. In *2021 IEEE 29th International Conference on Network Protocols (ICNP)*. IEEE, 1–11.

[25] Yandong Wang, Li Zhang, Jian Tan, Min Li, Yuqing Gao, Xavier Guerin, Xiaoqiao Meng, and Shicong Meng. 2015. HydraDB: a resilient RDMA-driven key-value middleware for in-memory cluster computing. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–11.

[26] Zilong Wang, Layong Luo, Qingsong Ning, Chaoliang Zeng, Wenxue Li, Xinchen Wan, Peng Xie, Tao Feng, Ke Cheng, Xiongfei Geng, et al. 2023. SRNIC: A scalable architecture for RDMA NICs. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 1–14.

[27] Bohong Zhu, Youmin Chen, Qing Wang, Youyou Lu, and Jiwu Shu. 2021. Octopus+: An rdma-enabled distributed persistent memory file system. *ACM Transactions on Storage (TOS)* 17, 3 (2021), 1–25.