**About The RDMA NIC In Lossless Network**

## Chapter 1: The Background

The TCP specification was defined in RFC793 by IETF in September of 1981, and TCP become the mainstream of communication protocol from then. When we were in 1Mb/s, 10Mb/s, 100Mb/s even 1Gb/s network speed period, the application communication protocols were very slow, such as the MPI protocol for computing, NFS and SCSI/iSCSI protocol for storage, etc, they usually took milliseconds or even seconds level time to communicate. Nobody felt that network communication protocol, such as TCPs were slow and impacted the performance. Also using host CPU to drive the network to reach the 1Mb/s, 10Mb/s, 100Mb/s and 1Gb/s line rare was not a big issue for CPU. CPU could only consume very little resource to reach the network line rate. According to Microsoft’s paper in 2002, it costed 0.5 CPU in Tx side and 1.2 CPUs in Rx side in order to reach 769Mb/s throughput with 1Gb/s network adapter and TCP protocol in Windows2000 SP2 platform. Microsoft forecasted when the network speed reach to 10Gb/s, it need cost 5 CPUs in Tx side and 12 CPU cores in Rx side to reach the same efficiency of 10Gb/s network adapter with TCP protocol.

As we knew, CPU was from single core processor becoming multi-core processor in order to resolve the scability issue, one physical CPU can support 20 or even more cores. However, the cost of 5 cores or 12 cores in communication with TCP is still too expensive. Many users want to have the CPU resource to be used in their application, not in communication.

How to resolve this issue to reach the network line rate and use as less as possible CPU cores? Some of resolutions were come to market. The most direct way is to offload the TCP protocol to somewhere, such as a dedicated processor to run TCP protocol and leave the host CPU resource to application. The 2nd way is to find another more efficiency protocol to do the communication, not use TCP any more, such as RDMA protocol. Obviously, RDMA become the mainstream at present and had been used in more and more applications.

Why the 1st solution to do the direct TCP offload was failed, there are some reasons for your reference.

Use dedicated processor to offload TCP will add extra cost for the adapter card. If we put one CPU or FPGA in the network card, you need run the offload engine in the CPU or FGPA. This increased the software develop workload and hardware cost.

If you use the dedicated host CPU cores to run the TCP offload, you can reduce the network adapter cost, but you still need consume CPU cores. This is conflicted with your original target to reduce CPU’s utilization in communication.

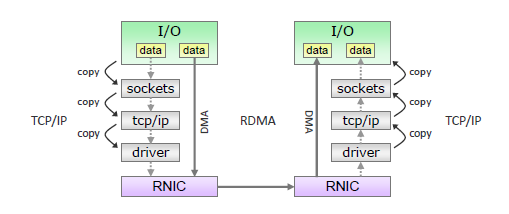
Even use TCP offload, the communication time of TCP protocol is still high, the best latency is about few microseconds. When the application latency is milliseconds, the few microseconds latency in network is acceptable. Now the average MPI communication latency is about 20 microseconds to 40 microsecond, the latest NVMe drive access latency is less than 10 microseconds. Few microseconds become too much for both computing and storage. Stay in TCP to do the further optimization seems hardly to resolve this issue.

Let’s talk about the 2nd solution, use the new protocol to resolve the CPU’s communication utilization issue.

RDMA, the full name is Remote Direct Memory Access. It can enable server-to-server, server to storage and storage to storage’s data movement directly between application memory without any CPU involvement. RDMA allows real CPU offloads and kernel bypass, and, RDMA provides the closer interface to talk with network adapter hardware to get better utilization of network HW resources while increasing speed and lowering link latencies.

What’s difference between RDMA and TCP, why we said RDMA can provide lower latency and CPU utilization and higher throughout?

Please refer to Picture 1, TCP does not define how to communicate with application, we usually use socket to create the API between TCP and application. When we want to use TCP protocol to send a data from one server to another server, we need create the TCP connection between the socket buffers from client to server. As we have the socket buffer and TCP connection, then we can copy the application data to socket buffer, then copy data from socket buffer to TCP/IP buffer, then copy to network adapter buffer and send the data to network. In receive side, we need the vice versa copy.

****

**Picture 1: RDMA Vs. TCP**

From this picture, you can also understand that TCP offload does not resolve the multiple copy cost, it was just move the cost from one processor to another processor.

What is RDMA? All we knew DMA(direct memory access) offers an ability of a device to access host memory directly, without the intervention of the CPU(s). RDMA (Remote DMA) likes an extension of DMA, to give the ability of accessing (i.e. reading from or writing to) memory on a remote server/storage without interrupting the CPU(s) on that system.

Let’s go back the Picture 1, we can create the direct channel between the application memory in any server and storage, send the data from application memory to remote memory, bypass all of buffer copies.

Using RDMA has the following major advantages:

* Zero-copy - applications can perform data transfer without all of those buffer copies which are consuming CPU resource.
* Kernel bypass - applications can perform data transfer directly from user space without the need to perform context switches with kernel.
* No CPU involvement - application can access remote memory without consuming any CPU in the remote machine. The remote memory can be directly written data or read from there without any intervention of remote process (or processor). The caches in the remote CPU won't be filled with the accessed memory content.
* Message based transaction - the data is handled as discrete messages and not as a stream, which eliminates the need of the application to separate the stream into different messages/transactions.

Based on these advantage of RDMA protocol, we are not only get the much lower latency(below 1 microsecond) since RDMA can eliminate many times buffer and the intervention with CPU, but also release the CPU resource from communication to applications. The application get both lower latency and CPU cores. These are just what we expect to reach from beginning.

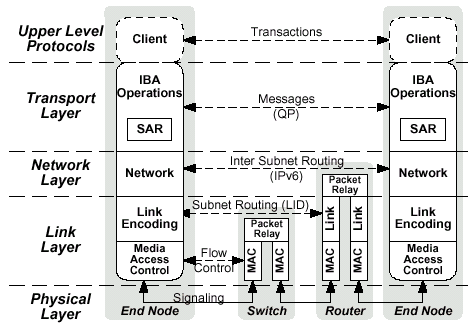
## Chapter 2: RDMA NIC

From the background discussion, RDMA can bring the great benefit for both communication and computing. More and more applications and data centers had started to deploy the system with RDMA NIC to reach the most efficient performance and scalability. So what kind of NIC can be RDMA NIC, how to ensure RDMA NIC to get the best performance?

Before we talk about RDMA NIC, let’s understand how RDMA NIC work and why RDMA NIC can reach the best performance.

### RDMA specification was defined by IBTA([InfiniBand Trade Association](https://www.baidu.com/link?url=lJUXWDGft-2rUeWRdkNftzaFWbfFEuhgeWvCMZkqtsi&wd=&eqid=d6a93017000117c5000000065f0538a3)) which is about 20 years mature protocol. It was defined for large scale, high performance and low CPU utilization. Same as TCP protocol, RDMA communication was defined with 5 layers.

1. Upper level protocols, application layer. RDMA is message based transaction, different application can choose different message to reach the best performance. RDMA protocol can support up to 2GByte message size. Upper level protocol can use different interface to talk with RDMA network adapter, such as RDMA interface, MPI, SCSI…
2. Transport layer. This is the most important layer for RDMA. Each RDMA packet contains a transport header. The transport header contains the information required by the endnode to complete the specified operation. The client of RDMA NIC communicates through “queue pair” (QP) includes a send work queue and a receive work queue. Similar as TCP protocol, RDMA didn’t define the specific API for application. RDMA defined the verbs as the abstract layer of RDMA protocol. Any application can create the interface to talk with verbs and use RDMA. User can define the application directly talk with verbs which is the closest path to network adapter hardware and usually get the best performance. User can also create the shim layer between verbs and other interface to get the flexibility to user RDMA but without changing their application. Such as MPI, iSCSI… We will talk about more about the transport layer in next chapter.
3. Network layer. Traditional kernel layer, including unicast and multicast operations. Specify the packet relay between different subnets.
4. Link layer. Mainly for the packet relay in same subnet. The link layer handles the sending and receiving of data across the links at the packet level. Services provided by the link layer include addressing, buffering, flow control, error detection and switching.
5. Physical layer. This is link technology dependent functions, such as link width, data encoding, voltage, packet framing. The physical layer is responsible for establishing a physical link, reporting and monitoring the status of the physical link, delivering received control and data bytes to the link layer, and transmitting control and data bytes from the link layer.



**Picture 2: RDMA Architecture Layers**

The RDMA NIC is the NIC which can support the specification based on those 5 layers definition, can offload the communication from host processor to network adapter, can communicate between endnodes without kernel involving. Such as NVIDIA Mellanox, Avago Broadcom and Intel are offering the RDMA NIC in the market.

## Chapter 3: How Does RDMA NIC Work

RDMA specification was from IBTA, IBTA defined the data format for InfiniBand RDMA and Ethernet RDMA(RoCE).

### 3.1 The Data Format Of RDMA NIC

Picture 3 shows the data format of InfiniBand RDMA, including the payload, different kinds of headers and ICRC(Invariant CRC) and VCRC(Variant CRC).



**Picture 3: InfiniBand RDMA Data Format**

Let’s start from RDMA payload, RDMA data supports the MTU(Maximum Transfer Unit) up to 4KB. User can define the MTU size under 4KB according to the application’s payload. Larger MTU can get better bandwidth and smaller MTU size can get better latency.

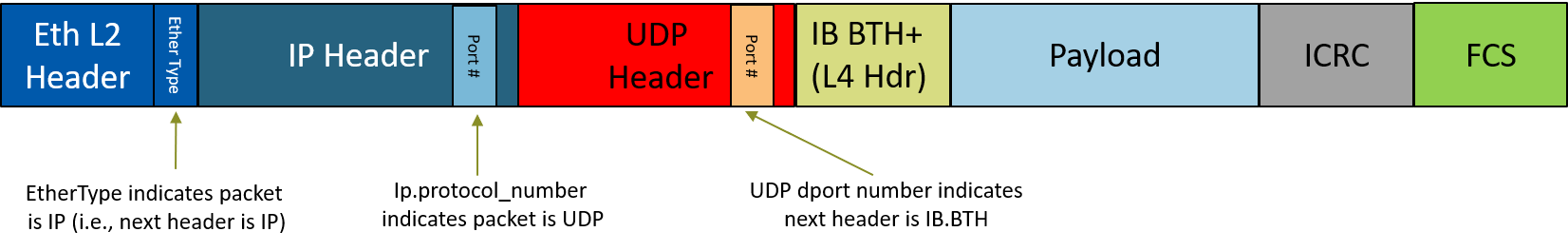
When the application message is transferred in RDMA network, the message need be decapsulated to the payload under the size of MTU in transmit side and encapsulated from payload to message in receive side. Some RDMA NIC can offload this decapsulation and encapsulation operation to hardware, some RDMA NIC requests host CPU to do this.

In order to ensure the integrity of RDMA data, the CRC computing is also mandatory in the datagram. This can also be offloaded to RDMA NIC or done by CPU.

The ETH(Extended Header) and BTH(Basic Transport Header) are part of transport layer. ETH includes the definition of datagram, RDMA operation type, acknowledge of RDMA operation, etc. BTH can be used for PKey(partition key) and Destination QP(queue pair).

For InfiniBand packet, IBTA defined to use GRH(Global Route Header) for the routing cross subnet; LRH(Local Route Header) for the routing inside same subnet. Please refer to picture 3.

For RoCE packet, IBTA defined to use InfiniBand BTH and payload as the payload of RoCE, use IB transport layer to guarantee the data reliability from hardware instead of TCP which is software based mechanism. The RoCE payload use UDP port to connect to IP header. RoCEv2 can support the routing cross different subnet, RoCE can support the routing within same subnet. Please refer to picture 4.



**Picture 4: Ethernet RoCE RDMA Data Format**

### 3.2 The Working Principles Of RDMA NIC

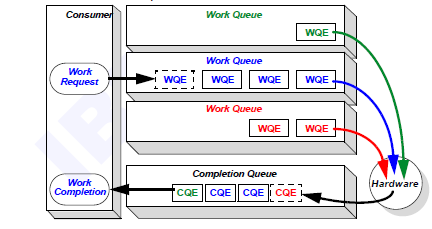
The basic RDMA working principles include following.

1. RDMA traffic sent directly to NIC without interrupting CPU
2. A remote memory region registration with the NIC first
3. The NIC records virtual to physical page mappings.
4. When NIC receives RDMA request, it performs a Direct Memory Access into memory and returns the data to client.
5. Kernel bypass on both sides of traffic

Let’s talk about how RDMA does work. Work queues(WQ) and Completation Queue(CQ) are the basic units of RDMA data transfer.

A Work Request(WR) are the work items which the RDMA NIC hardware should execute. As the WR is created, RDMA NIC will add the WR to Work Queue. When a WR is completed, it may create a Work Completion which provides information about the ended WR, such as data transfer type, opcode, amount of data send/received, and so on.

The Work Queues(WQ) is scheduled by the RDMA NIC hardware. WR execution ordering is guaranteed within the same WQ; but no guarantee about the order between different WQs. The WQ can be either Send Queue or Receive Queue. The operation to add a WR to a WQ is called “posting a WR”. Every WR that was posted is considered “outstanding” until it ends with Work completion. During the outstanding status, it can not know if it was scheduled by the hardware or not; the send buffer cannot be (re)used or freed; the receive buffer content is undetermined. Please refer to picture 5.



**Picture 5: RDMA Data Transfer Model**

A Send Queue (SQ) is a Work Queue that handles sending messages. Every entry is called a Send Request (SR). It specifies how data is used; what memory buffers to use; how to send or receive data based on the opcode; how much data is sent and some more attributes. Adding a SR to an SQ is called “posting a Send Request” and SR may end with a Work Completion.

A Receive Queue (RQ) is a Work Queue that handles incoming messages. Every entry is called a Receive Request (RR). It specifies which memory buffers to be used if RR is consumed based on opcode. Adding a RR is to the RQ called “posting a Receive Request”. An RR always ends with a Work Completion. This queue may send data as a response based on opcode.

A unified SQ and A RQ are always created at same time and called Queue Pair. A SQ is always consumed and the RQ is consumed for some operations and is not used for some operations.

The Queue Pair is the actual object that transfers data. It encapsulates both Send and Receive Queue. Each of them is completely independent. Send Queue can generate Work Completion for every Send Request or for specific Send Requests. Receive queue generates Work Completion for every completed Receive Request. The QP is full duplex. A QP represents a real hardware resource. Please refer to picture 6.



**Picture 6: Queue Pair**

A Work Completion holds information about a completed Work Request. Polling for Work Completion checks if the processing of a Work Request has ended. Every Work Completion contains information about the corresponding completed Work Request

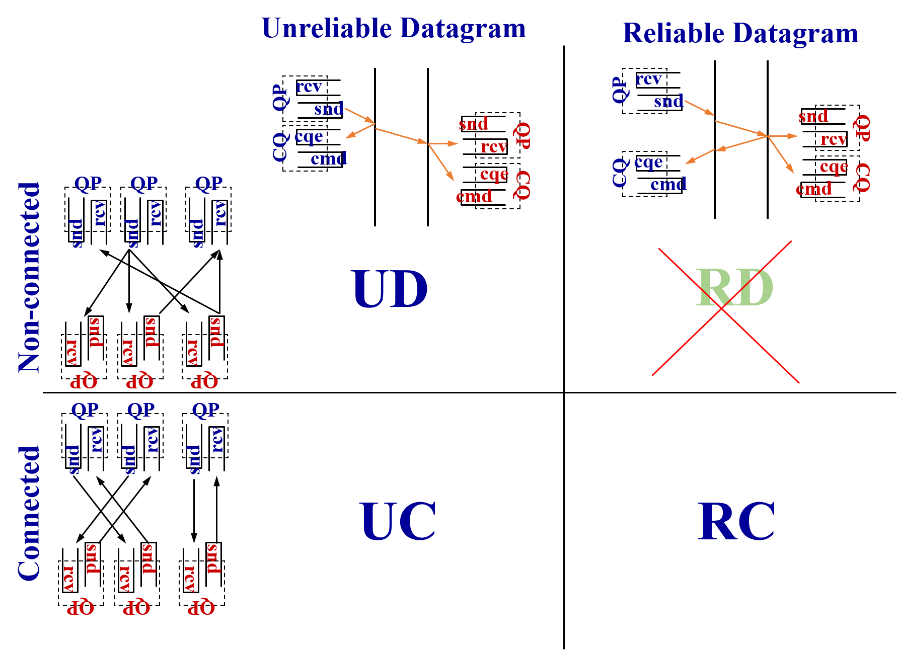
The Work Completion of Send Requests marks a Send Request was performed and its memory buffers can be (re)used. For reliable transport QP, this means that the message was written in the buffers if status is successful; for unreliable transport QP, this means that the message was sent from the local port.

The Work Completion of Receive Requests marks an incoming message was completed and its memory buffers can be (re)used. It contains some attributes about the incoming message, such as size, origin, etc.

### 3.3 The Transport Type Of RDMA NIC

As we talked in chapter 2, we talked about transport layer is the most important layer in RDMA communication. So what kind of services can the Transport Layer provide?

1. Connection oriented
2. Reliability
3. In-order delivery
4. Data integrity
5. Flow control / rate control / congestion control
6. Multiplexing (of multiple data sources)



**Picture 7: IBTA Defined Transport Type**

There are three major transport types was defined by IBTA,

1. The RC (Reliable Connected ) is which an RC QP is connected to a single RC QP. The end to end reliability is guaranteed by both connection and datagram, such as ordering, integrity and arrival of all packets. The operations with RC need ACK.
2. The UC(Unreliable Connected) is which an UC QP connected to a single UC QP. The reliability is not guaranteed by datagram.
3. The UD(Unreliable Datagram) is which an UD QP can send/receive messages to/from any UD QP. The reliability is not guaranteed by neither datagram nor connection. Multicast is supported by UD. Each UD message is limited to one packet.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **UD** | **UC** | **RC** |
| Send / Receive | Yes | **Yes** | Yes |
| RDMA Write | No | Yes | Yes |
| RDMA Read / Atomic | No | No | Yes |

**Form 1: The Mapping Between RDMA NIC Operation And Transport Type**

Based on the different transport type, RDMA NIC can implement the different RDMA operations. Such as send/receive operation can use any kind of transport type; RDMA write can not use UD transport type and RDMA Read abd RDMA Atomic can use RC transport type only.

### 3.4 The Data Transfer Operation Of RDMA NIC

For RDMA NIC, it can use following 4 operations to transfer the data according to the application requirement.

1. Send / Receive - Two-sided communication
2. RDMA Write - One-sided communication, responder is not involved
3. RDMA Read - One-sided communication, responder is not involved
4. Atomic Operations - One-sided communication, read-modify-write transaction

#### 3.4.1 The Send/Receive Operation Of RDMA NIC

Requester

Responder

**Post RR**

**Post SR**

data

ACK

**Poll CQ**

**Poll CQ**

sync

**Picture 8: Send/Receive Operation**

The responder application allocates receive buffer on the User Space Virtual Memory  
register it with the RDMA NIC, and place a receive Work Request on the Receive Queue.

The requester allocates a send buffer on the user space virtual memory, register it with the RDMA NIC, place a Send Request on the Send Queue.

The responder send Ready To Receive notification to requester.

Then requester RDMA NIC executes the Send Request, read data from the buffer of the host memory and send to remote side RDMA NIC.

When the packet arrives the responder RDMA NIC, it executes the receive WQE commands and place the buffer content in the appropriate location, and generate a Completion Queue.

#### 3.4.2 The RDMA Write Operation Of RDMA NIC

Requester

Responder

**Post SR**

data + addr + rkey

ACK

**Poll CQ**

**Picture 9: RDMA Write Operation**

Application performs memory registration in responder and passes address and access keys to   
requester. No Receive Queue is assigned in responder side.

The requester allocates a receive buffer on the user space virtual memory and register it with the RDMA NIC, then place a Send Request on the Send Queue with the remote side’s virtual address and the remote permission key.

The requester RDMA NIC executes the Send Request commands, reads the buffer and send to responder, then generates the Send Completion.

When the packet arrives to the responder RDMA NIC, it already knows the responder’s memory virtual address and access key, just need look up the memory table and write to right host memory directly. No use of responder RDMA NIC queues.

From the RDMA write operation, there is only requester side involved the data transfer operation, that’s why we call it one sided operation.

#### 3.4.2 The RDMA Read Operation Of RDMA NIC

Requester

Responder

**Post SR**

addr + rkey

data

**Poll CQ**

**Picture 10: RDMA Read Operation**

Application performs memory registration in responder and passes address and access keys to   
requester. No Receive Queue is assigned in responder side.

The requester allocates a receive buffer on the user space virtual memory and register it with the RDMA NIC, then place a Send Request on the Send Queue with the responder’s virtual address and the permission key.

The responder is passive. The responder RDMA NIC looks up memory table based on the permission key, virtual address and length from incoming RDMA request, then read the data from responder’s memory region and send to requester.

From the RDMA read operation, there is also only requester side involved the data transfer operation and responder is passive, it is one sided operation, too.

#### 3.4.4 The RDMA Atomic Operation Of RDMA NIC

Requester

Responder

**Post SR**

data + addr + rkey

data

**Poll CQ**

**Picture 11: RDMA Atomic Operation**

The atomic operation is a combination transaction of, RDMA read + modify + RDMA write.

Application performs memory registration in responder and passes address and access keys to   
requester. No Receive Queue is assigned in responder side.

The requester post the Send Request with the remote side’s virtual address and the remote permission key.

The responder side RDMA NIC read the data from memory region, modify the data, then write the new data to responder memory and send the original data to requester memory.

The data modify includes “Fetch and Add” operation and “Compare and Swap” operation.

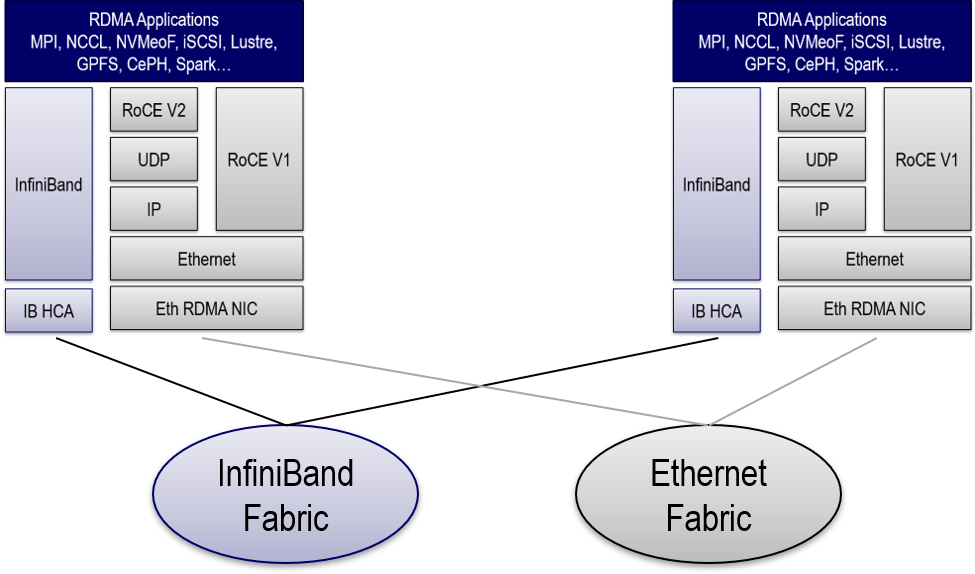
The “Fetch and Add” operation is done in atomic way for 64-bits numbers at responder’s memory. RDMA NIC fetch data from memory, adds a value, stores the new number to memory and sends the original value to the requester.

The “Compare and Swap” operation is done in atomic way for 64-bits numbers at responder’s memory. RDMA NIC fetch data from memory, compares the data with number1. If they are equal, then store number2 in memory and send the original value to the requester

From the RDMA atomic operation, even there are the read, modify and write operations in responder side, but still only requester side involved the data transfer operation, responder is passive. It is also one sided operation.

### 3.5 The Lossless Network For RoCE

RDMA communication can be run on top of both InfiniBand fabric and Ethernet fabric. For InfiniBand network, it is a nature lossless network with the well-defined flow control and congestion control technologies to guarantee the data integrity. Ethernet was defined at lossy network from beginning. With the data growth rapidly and computing powering becoming more and more strong in the data center, more and more applications couldn’t accept the performance loss which the packet drop in the network caused. The lossless Ethernet network become very critical in order to ensure the application performance.

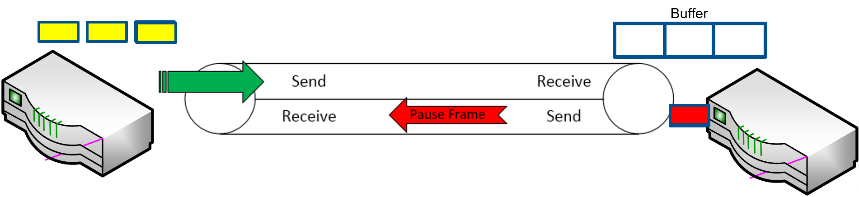


**Picture 12: RDMA End To End Fabric**

The lossless network doesn’t mean that packets are absolutely never lost. Moreover, the RDMA/RoCE transport protocol has the end-to-end reliable delivery mechanism with built-in packet retransmission logic, such as how to handle the data packet drop, how to handle the acknowledge packet drop... This logic is typically implemented in hardware and is triggered to recover from lost packets without the software stack.

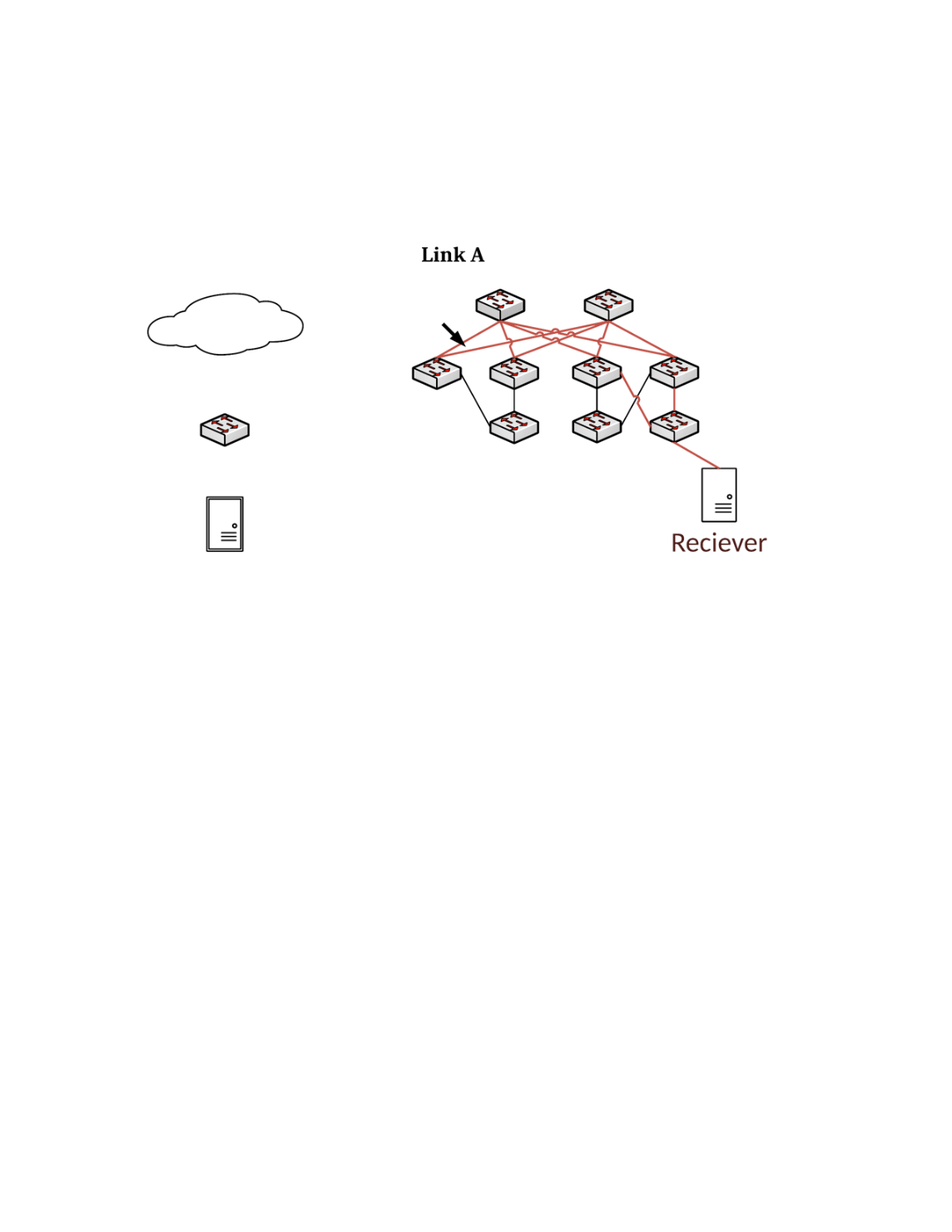
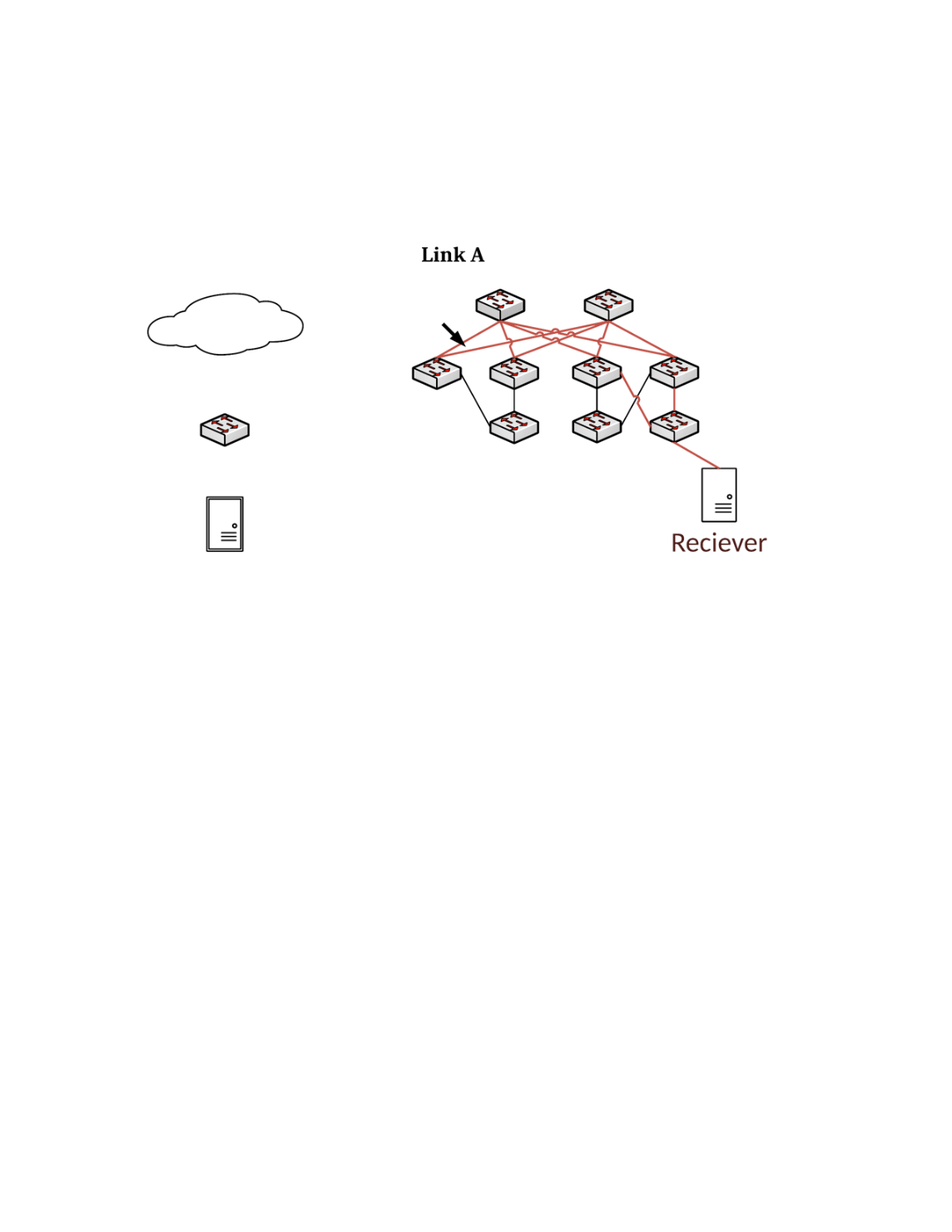
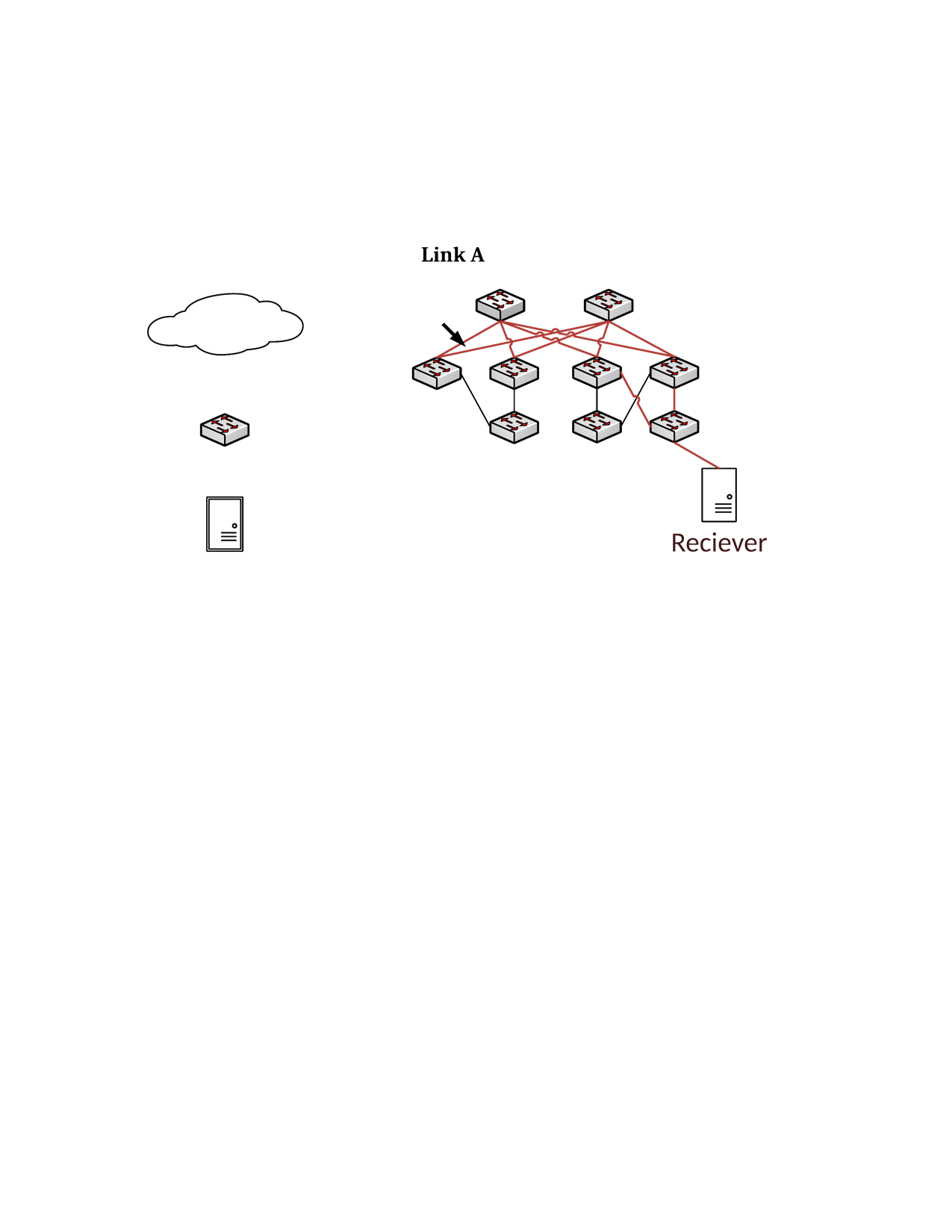
Though RDMA NIC can run on both lossy network and loss network, however, an underlying lossless network may provide the lower packet loss rate to reduce the retransmission for RoCEv2.

In DCE(Data Center Ethernet) specification, the lossless network is achieved through the use of Link-Layer Flow-Control. IEEE802.1Qbb specifies PFC(per-priority link-layer flow-control) for Ethernet Networks was widely used for RoCEv2 traffic. However, this might arise a new problem called “congestion spreading” because the pause frame which we use for PFC flow control cannot distinguish between the flows. An small buffer implies more link pauses which imply in congestion spreading and then reduce the effective link bandwidth. We must have a trade-off of increasing buffers or increasing the link bandwidth.



**Picture 13: PFC Work Flow**

In order to reduce the impact of “congestion spreading” from PFC, ECN(Explicit Congestion Notification) is usually used to manage the bandwidth of each flow and works with PFC in RoCEv2 Lossless network . We can significantly reduce the PFC pause numbers through optimize the ECN. Since we are talking about NIC, we will not talk about the detail for PFC and ECN in switch working behavior.



Sender RDMA NIC

Reaction Point (RP)

Switch

Congestion Point (CP)

Receiver RDMA NIC

Notification Point (NP)

Congestion Notification

Congested Traffic

Congested Traffic  
(ECN marked)

**Picture 14: ECN Work Flow**

In order to classify different traffic in Ethernet lossless network, we need configure the QoS of each traffic, give the different priority for different traffic, such as give RoCE high priority and TCP lower priority; configure the buffer for different traffic, such as configure RoCE traffic and TCP traffic with different and dedicated buffer; and PFC. The traffic classification is indicated by DSCP (Differentiated Service Code Point, layer 3, in IP header) for RoCEv2 and PCP (Priority Code Point, layer2, in Vlan tag) for RoCE. DSCP is often used by data center applications. Please pay attention to give the CNP(Congestion Notification Packet) the highest priority to ensure CNP can arrive at sender from receiver as soon as possible. Then the DSCP or PCP of each traffic will be mapped to switch port’s traffic classification.

## 

**Picture 15: Lossless Network Traffic Classification**

## Chapter 4: The Applications Of RDMA NIC

We learned what’s the RDMA NIC and how RDMA NIC works. Where can RDMA NIC be used?

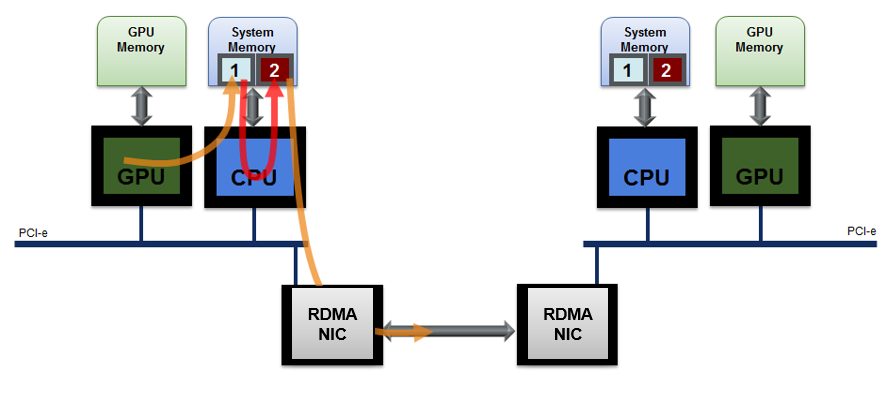
RDMA NIC had been used in different fields, such as HPC(High Performance Computing), AI(Artificial Intelligence), storage(block storage, file storage, object storage and NVMeoF) and others. We will use one computing over RDMA scenarios and one storage over RDMA to introduce how RDMA benefit the applications.

### 4.1: GPU DirectRDMA

The GPU DirectRDMA is comprised by the PeerDirect technology of PCIe and RDMA technology of network. Support for any PCIe peer which can provide access to its memory, such as NVIDIA GPU, XEON PHI, AMD GPU, FPGA, and so on.

Why GPU and GPU communication need GPU DirectRDMA? As we know, GPU communications use “pinned” buffers for data movement, RDMA NIC also use “pinned” memory to communicate with remote side “pinned” memory. A section in the host memory that is dedicated for the GPU, and the second section in host memory dedicated for RDMA NIC.

In general case, when one GPU wants to copy the data to another GPU in remote server, it need copy the data from GPU memory to the CPU memory which is pinned by GPU, then host CPU will copy the data from GPU pinned memory to the memory which is pinned by RDMA NIC. Afterwards, RDMA NIC can transmit the data from local server to remote server. In remote server side, the data will arrive at the memory which RDMA NIC pinned first, then CPU copy the data to the memory which is pinned by GPU, eventually, the data can arrive at the remote GPU memory from host memory. Please refer to picture 16 for the GPU to GPU data copy process.

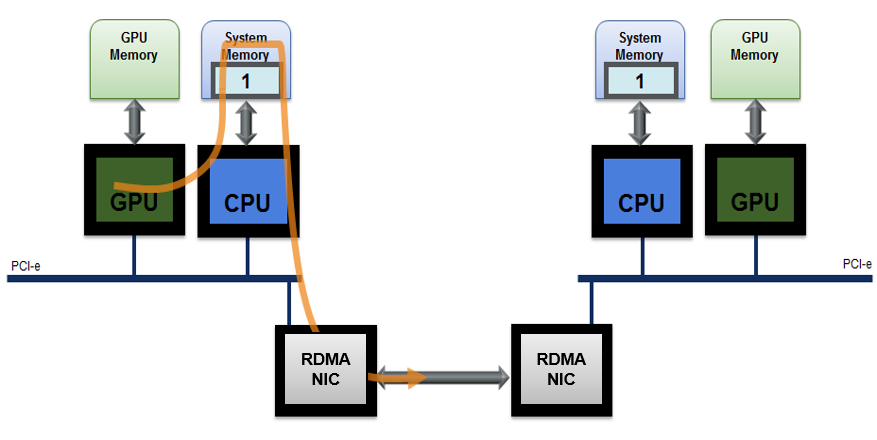


**Picture 16: The Data Transfer Before GPU DirectRDMA**

Of course, the copy cost of this process is much lower then using TCP to pass the data between GPUs. But it still caused lots of issues for the data transfer.

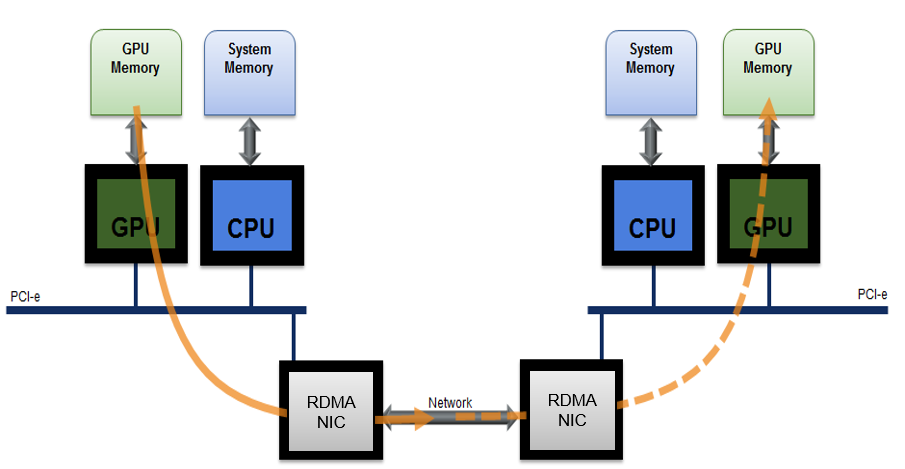
1. It consumes the GPU resource. CPU may become the bottleneck of data copy.
2. This brings the higher latency and lower bandwidth between GPU to remote GPU.
3. The data copy consumes lots of host memory. This may impact the application performance and increase the system TCO.

If we can do the design optimization such as write-combining and overlapping GPU computation and data transfer which allow network and GPU can share “pinned” (page-locked) buffers, this can eliminate the need to make a redundant copy in host memory. In this case, the data can be directly sent out through RDMA when the data arrive at host memory from GPU memory; then the data can be directly write to GPU pinned host buffer when the data arrives at remote server through RDMA. There is no buffer copy any more inside host memory. This technology is called GPU Direct technology.



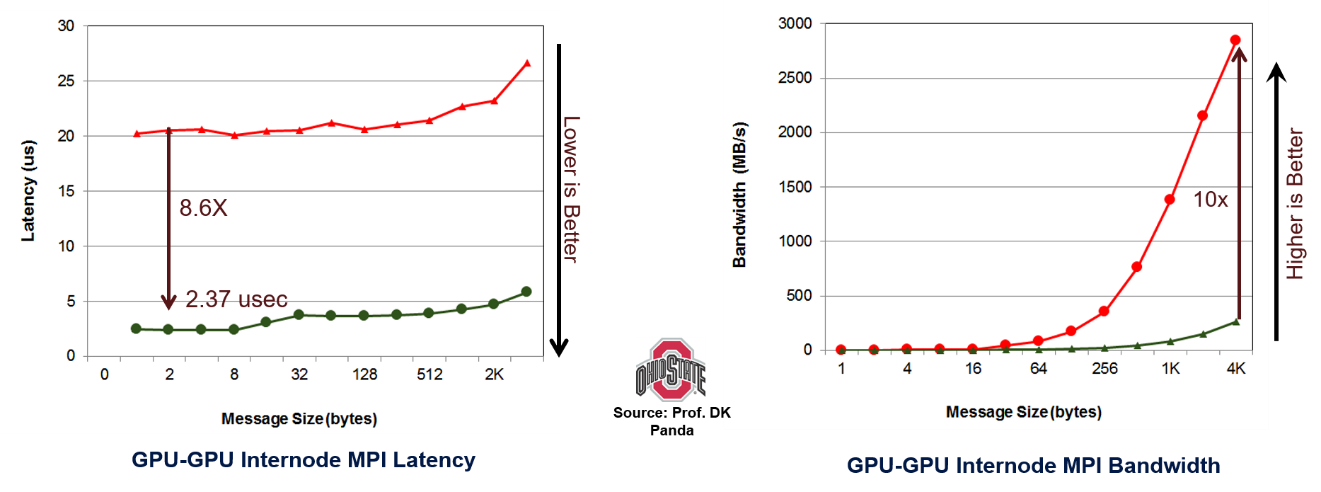
**Picture 17: The Data Transfer Based On GPU Direct**

In order to eliminates CPU bandwidth and latency bottlenecks when we uses RDMA transfers between GPUs, we can do further optimization to create the RDMA channel between GPU memory and remote GPU memory. This results in significantly improved communication efficiency between GPUs in remote nodes. We can use CPU prepares and queues communication tasks on GPU and use GPU to trigger the communication on RDMA NIC, then the RDMA NIC directly accesses GPU memory to send/recv or read/write data. This is the GPU DirectRDMA technology.



**Picture 18: The Data Transfer Based On GPU DirectRDMA**

With the optimization of GPU DirectRDMA, the GPU communication can get 10 times better performance than traditional communication way(the performance data from OSU’s paper). This technology become the mandatory technology in HPC and AI applications for the performance and scability purpose. All MPIs and NCCL had integrated RDMA in native support.



**Picture 19: The Performance Of GPU DirectRDMA(From OSU)**

### 4.2: NVMe Over Fabric Over RDMA

NVMe(Non-volatile Memory Express over PCI Express) is an efficient programming interface for accessing NVM devices over a PCIe bus by redesigning and standardizing the storage interface. The new programming interface can lock-free multi thread/process NVM access.

The legacy storage stacks for accessing storage device over the network could be used to operate NVMe devices. However, their synchronizations and command translation requirements defeat the benefits of NVMe devices for remote access. NVMeOF(Non-volatile Memory Express over Fabrics) was designed for taking advantage of the new and efficient architecture.

The NVMe over RDMA Fabrics protocol was defined by default in NVMeOF specification in order to extend the benefits of the streamlined interface, extend the simplified command set and the performance of NVMe supporting devices over the network and provide efficient access to remote NVMe devices.

A system for accessing storage over the network is comprised of an initiator, where the application that initiates the storage command reside, a target where the non-volatile memory resides, and a network for the communication between them, which we call the fabric. A storage protocol running over the fabric is used to transfer data between the initiator and the target.

The initiator runs an operating system and a file server which implements storage access through a block device that uses an NVMe standard block device driver. Initiators can access both local devices through PCIe bus and remote NVMe devices through fabric as a standard file system or block device.

An NVMe target consists of RDMA NICs and NVMe devices which are connected to each other, and to system memory by the processing unit’s PCIe root-complexes. Storage input/output commands from the networks are transformed by the CPU or RDMA NIC to local NVMe operation on local NVMe drives.

Login is the process of creating an NVMe over Fabrics controller in a remote NVMe over Fabrics target. The controller is comprised by an admin submission queue and completion queue exposed by a single RDMA channel, and an I/O submission queue and completion queue exposed by a single channel.

The initiator may post create I/O queue capsule to the admin queue in order to create other channels and queues for submitting I/O commands.

The initiator starts an authentication process which is defined by NVMe over Fabrics specification.

The RDMA connections are established by the initiator assigning the proper queue size to satisfy the number of outstanding IO/Admin commands on these queues.

The initiator posts a login command capsule which is transferred through the RDMA connection to create the controller in the target and specify the type of each connection (IO/Admin).

The target creates the controller, associates the IO and Admin queues to the appropriate RDMA channels and sends a login response capsule.

When the initiator receives the response, the RDMA queue pair becomes an active controller AQ and IO channel. At that stage, the initiator has an administrative queue and an IO queue and is a functional controller over RDMA fabrics. The administrative queue may be used to open more IO queues.

The RDMA connection above has to support reliable delivery of RDMA WRITE, RDMA READ and RDMA SEND to satisfy NVMe over Fabrics requirements. These operations are supported on both the RC and the DC transport services, therefore the NVMe over Fabrics application is agnostic to these services.



## Chapter 5: Conclusion

RDMA technology was a game changer for large scale system. It resolved the performance issue and scability issue through its complete software stack, kernel bypass and CPU offload. It decouples the communication computing and CPU and makes the data-centric computing to become the reality. The RDMA network had started to become the ideal network in more and more data center, AI center and HPC center.

However, there are still lots of challenge to be resolved in order to utilize the RDMA to reach the best performance, such as the congestion control in RDMA lossless network, RDMA reduces the communication time to below 1us, the switch buffer may be full very fast when the congestion happen; the integration between high performance RDMA and other features, like the security feature, data compress feature, etc; how to help the user to easy use RDMA; how to guarantee the RDMA performance cross data centers…

With the data growing, there might bring more challenge for RDMA, just like TCP was facing the so many challenges 15 years ago, RDMA gave the direction to address those challenges and open a mind to go to new generation data center. The more and more new technologies around RDMA NIC are driving the mature RDMA eco-system, such as the lossy RDMA, different congestion control/flow control technologies, UCX(Unified Communication X)…