Imperial College London

LSDS
Large-Scale Data & Systems Group

RDMA Tutorial

Jana Giceva

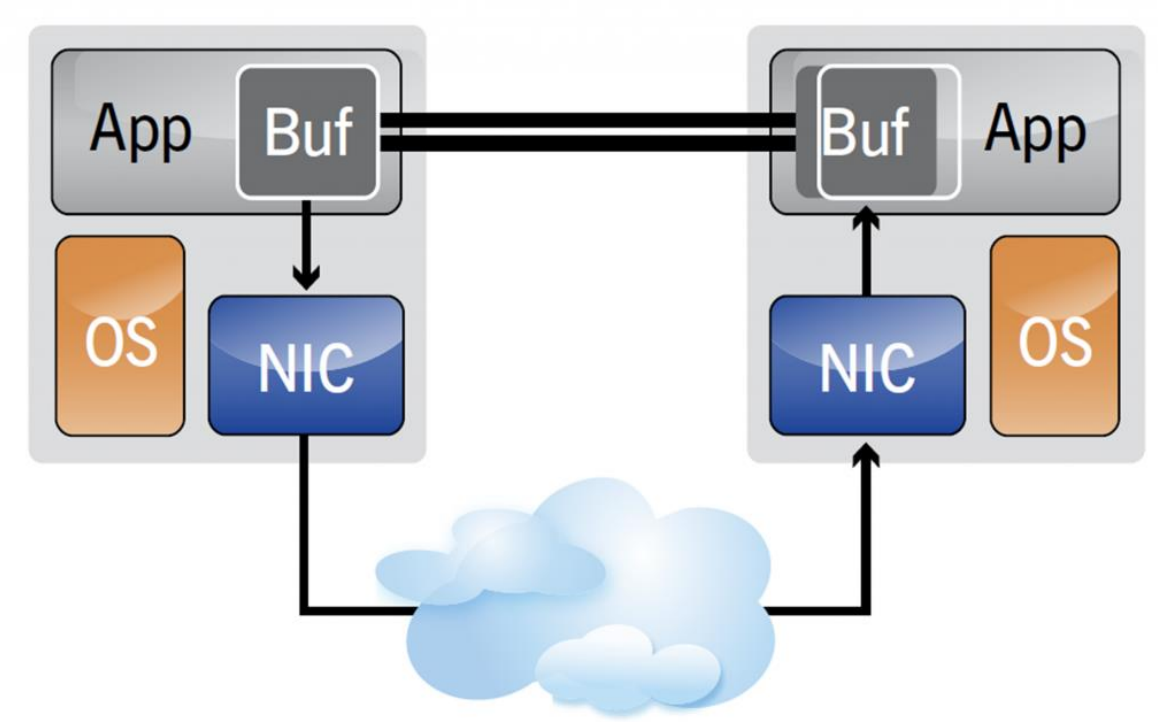
Large-Scale Data & Systems (LSDS) Group
Imperial College London

http://lsds.doc.ic.ac.uk
<jgiceva@imperial.ac.uk>

What is *RDMA*?

Remote Direct Memory Access

RDMA is a hardware ***mechanism*** through which the network card (***NIC***) can ***directly access*** all or parts of the ***main memory*** of a ***remote node without involving the processor***.



RDMA properties

Remote – data is transferred between nodes in a network

Direct – no CPU or OS kernel is involved in the data transfer

Memory – data transferred between two apps and their virtual address spaces

Access – support to send, receive, read, write, and do atomic operations

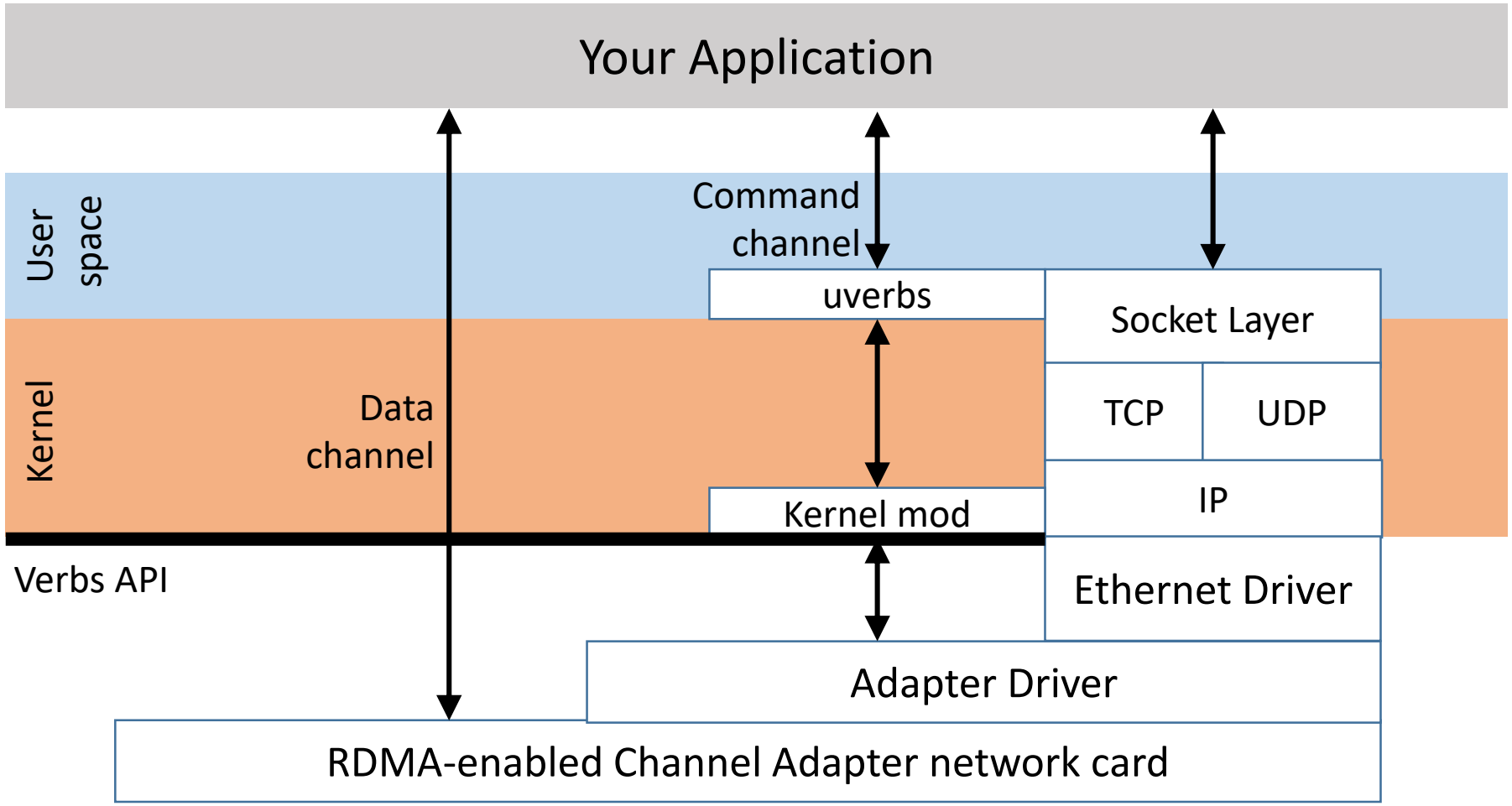
Main highlights of RDMA

- Zero-copy data
- Bypasses the CPU
- Bypasses the OS kernel
- Message based transactions

Benefits of using RDMA

- ✓ High throughput (bandwidth)
- ✓ Low end-to-end latencies
- ✓ Low CPU utilization
  One-sided RDMA operations do not involve the remote CPU at all.
- ✓ Low memory bus contention
  No data is copied between the user space and kernel, and the other way around.
- ✓ Asynchronous operations
  Great for overlapping communication and computation.

Traditional TCP/IP sockets vs RDMA



src: InfiniBand Trade Association: Introduction to IB for end users

Setting up the RDMA data channels

Buffers need to be ***registered*** with the ***network card*** before used

During the registration process:

- ***Pin memory*** so that it cannot be swapped by the Operating System.
- ***Store*** the ***address translation*** information ***in the NIC***.
- ***Set permissions*** for the memory region.
- ***Return*** a ***remote*** and ***local key***, which are used by the adapters when executing the RDMA operations.

Work Queues

RDMA communication is based on a set of three queues

- ***Send***
- ***Receive***
- ***Completion***

} work queues, always created as a Queue Pair (QP)

The ***send*** and ***receive*** queues are there to schedule the ***work*** to be done.

A ***completion*** queue is used to ***notify*** when the work has been completed.

Queue Elements

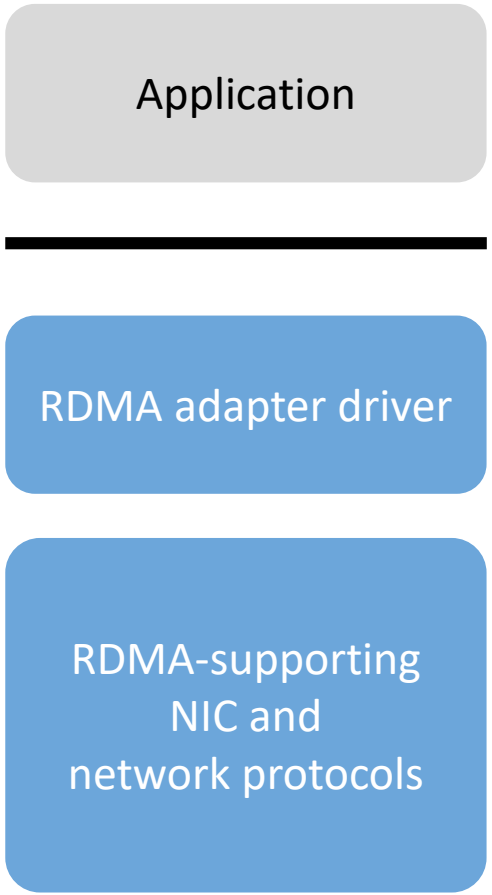
Applications issue a job using a ***work request*** or a ***work queue element***

A work request is a small ***struct*** with a ***pointer to a buffer***:

- In a *send queue* – it's a pointer to a message to be sent.
- In a *receive queue* – it's shows where an incoming message should be placed.

Once a work request has been completed, the adapter creates a ***completion queue element*** and enqueues it in the *completion queue*.

RDMA's network stack overview



- Posts work requests to a queue
- Each work request is a message, a unit of work

- Verbs interface – allows the application to request services

- Maintains the work queues
- Manages address translation
- Provides completion and even mechanisms

- Transport layer: reliable/unreliable, datagram, etc.
- Packetizes messages
- Implements the RDMA protocol
- Implements end-to-end reliability
- Assures reliable delivery

src: InfiniBand Trade Association: Introduction to IB for end users

Network protocols supporting RDMA

- InfiniBand (IB)
  - QDR 4x – 32 Gbps
  - FDR 4x – 54 Gbps
  - EDR 4x – 100 Gbps
- RoCE – RDMA over Converged Ethernet
  - 10 Gbps
  - 40 Gbps
- iWARP – internet Wide Area RDMA Protocol

RDMA is just a *mechanism*

Does ***not*** specify the ***semantics*** of a data transfer

RDMA networks support two types of memory access models:

- ***One sided*** – RDMA read and write + atomic operations
- ***Two sided*** – RDMA send and receive

RDMA Send and Receive

Traditional message passing where ***both*** the ***source*** and the ***destination*** processes are ***active***ly involved in the communication.

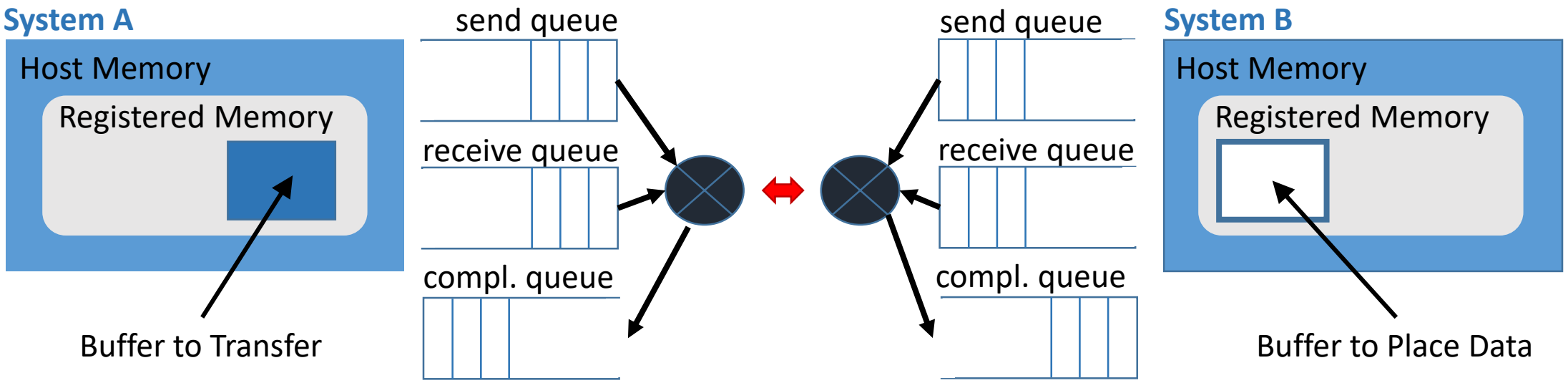
Both need to have ***create***d their queues:

- A ***queue pair*** of a ***send*** and a ***receive*** queue.
- A ***completion queue*** for the queue pair.

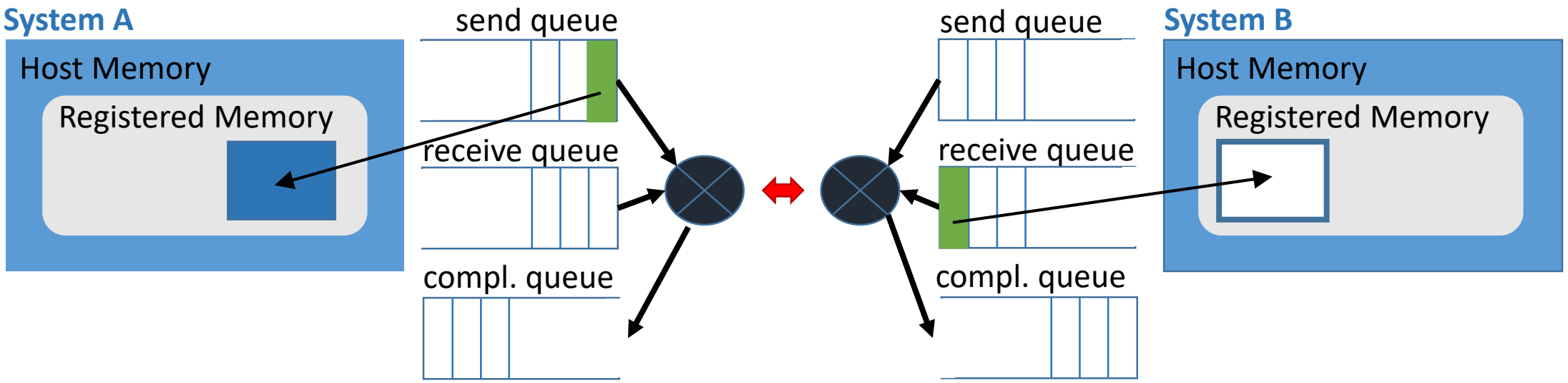
Sender's work request has a pointer to a buffer that it wants to send. The WQE is enqueued in the send queue.

Receiver's work request has a pointer to an empty buffer for receiving the message. The WQE is enqueued in the receive queue.

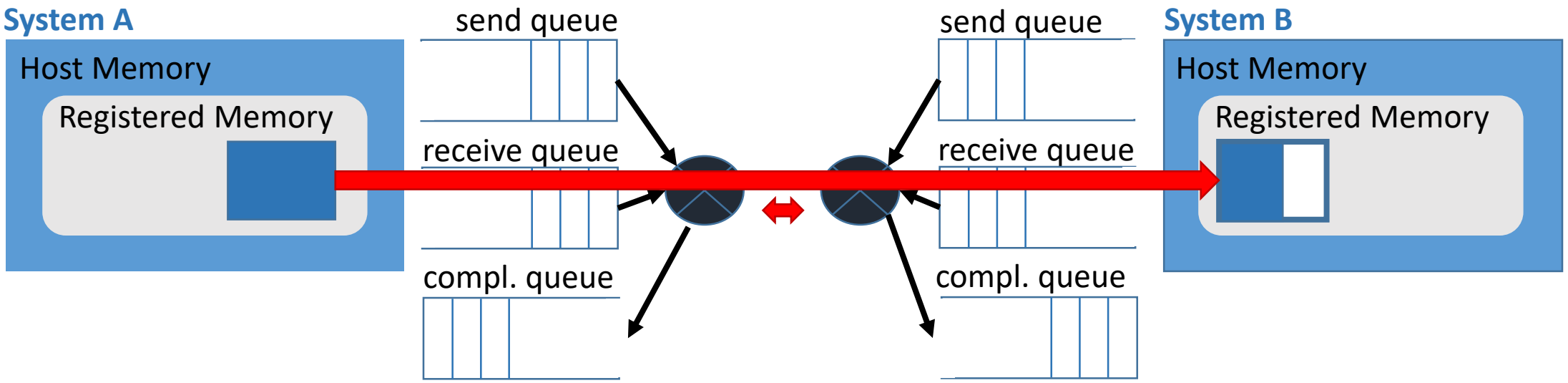
Example RDMA send



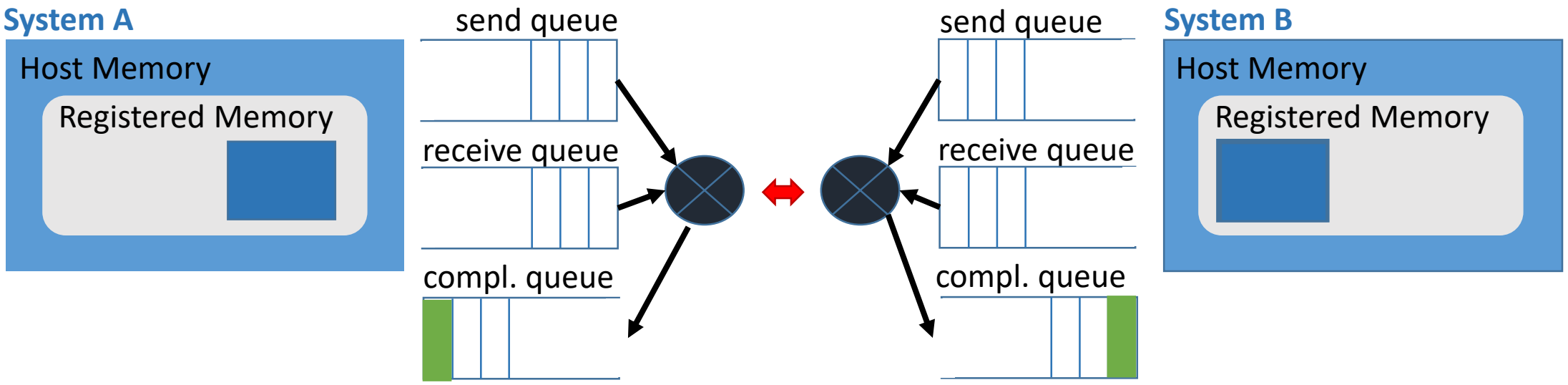
Example RDMA send



Example RDMA send



Example RDMA send



RDMA Read and Write

Only the ***sender*** side is ***active***; the ***receiver*** is ***passive***.

The passive side issues no operation, uses no CPU cycles, gets no indication that a “read” or a “write” happened.

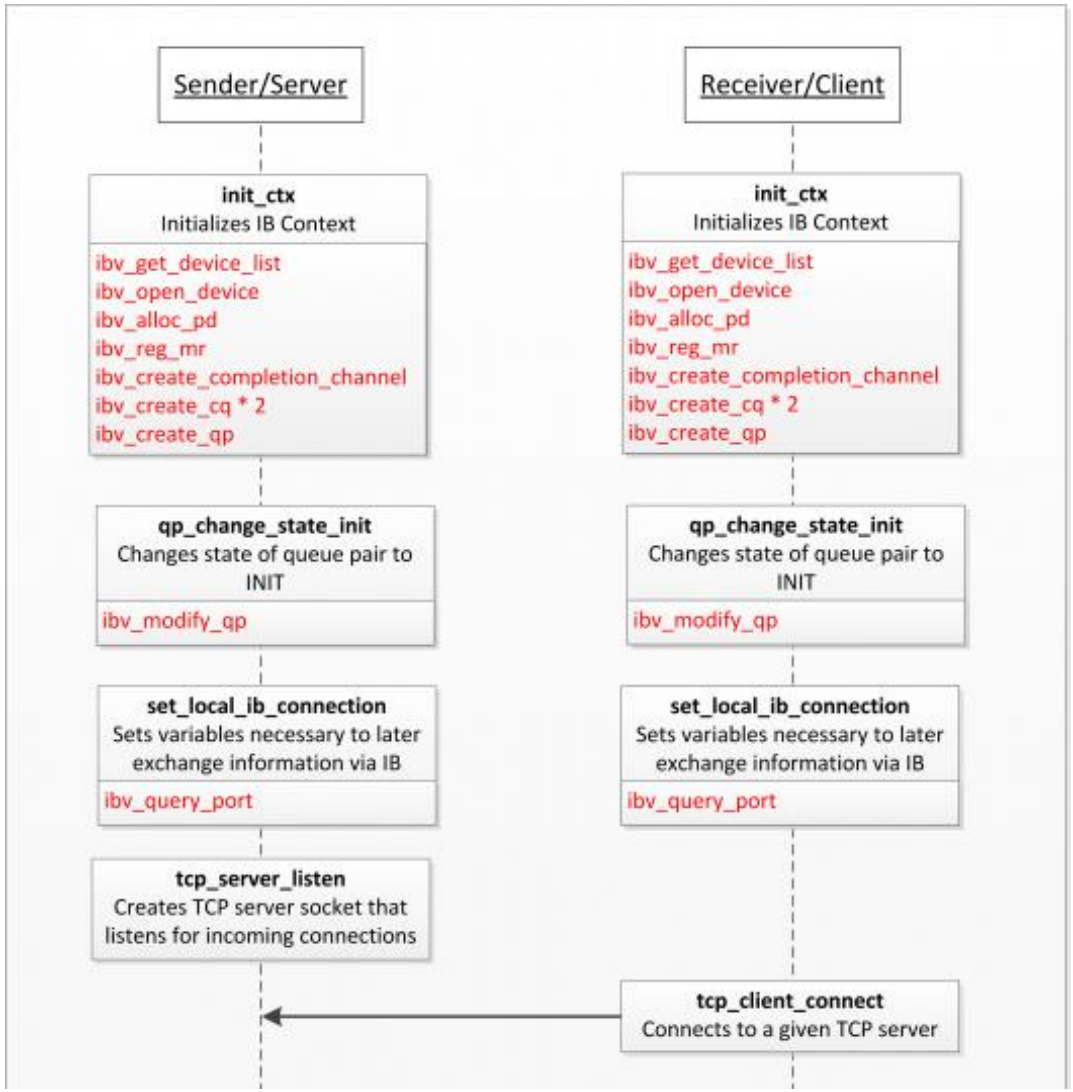
To issue an RDMA ***read*** or a ***write***, the work request ***must include:***

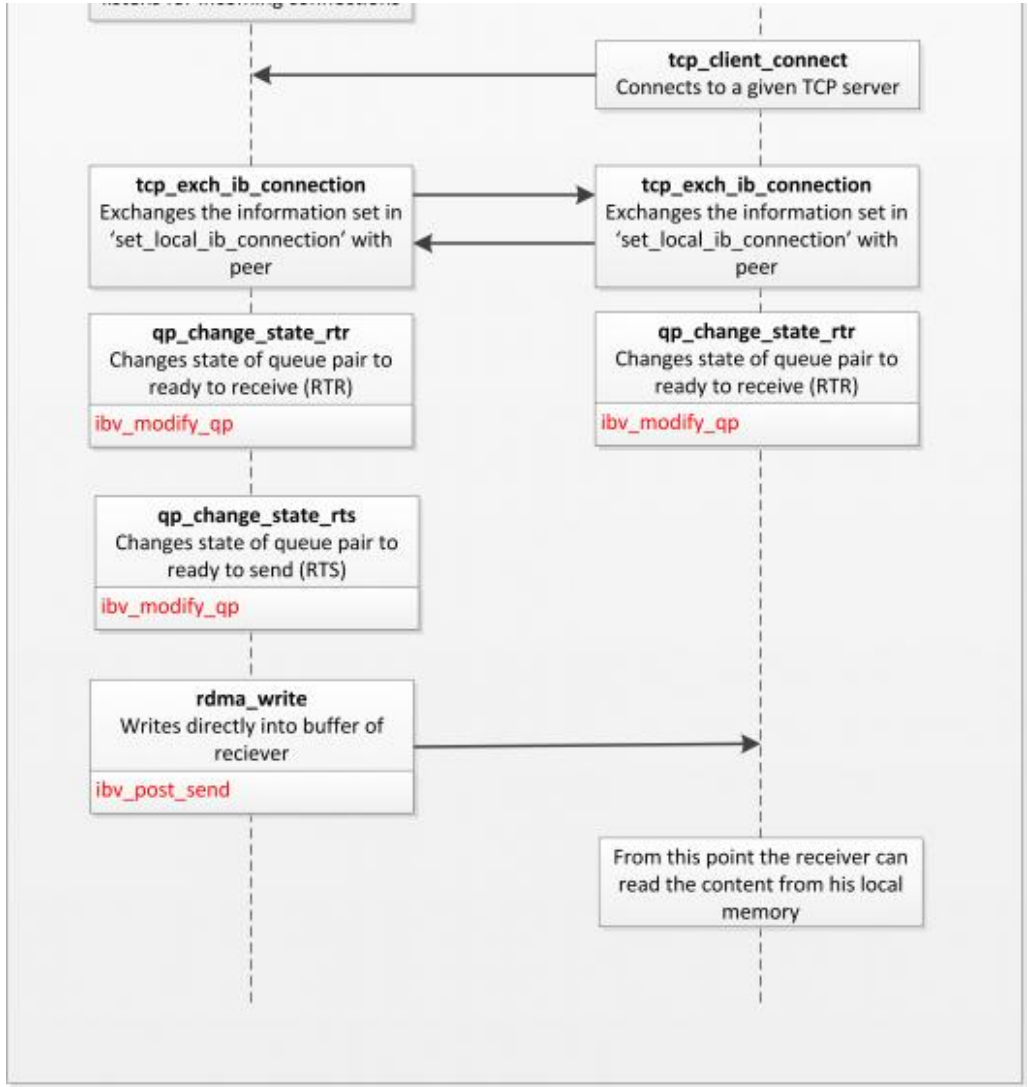
1. the ***remote*** side’s ***virtual memory address*** and
2. the ***remote*** side’s ***memory registration key***.

The active side must obtain the passive side’s address and key beforehand.

Typically, the traditional RDMA send/receive mechanisms are used.

Using the verbs API

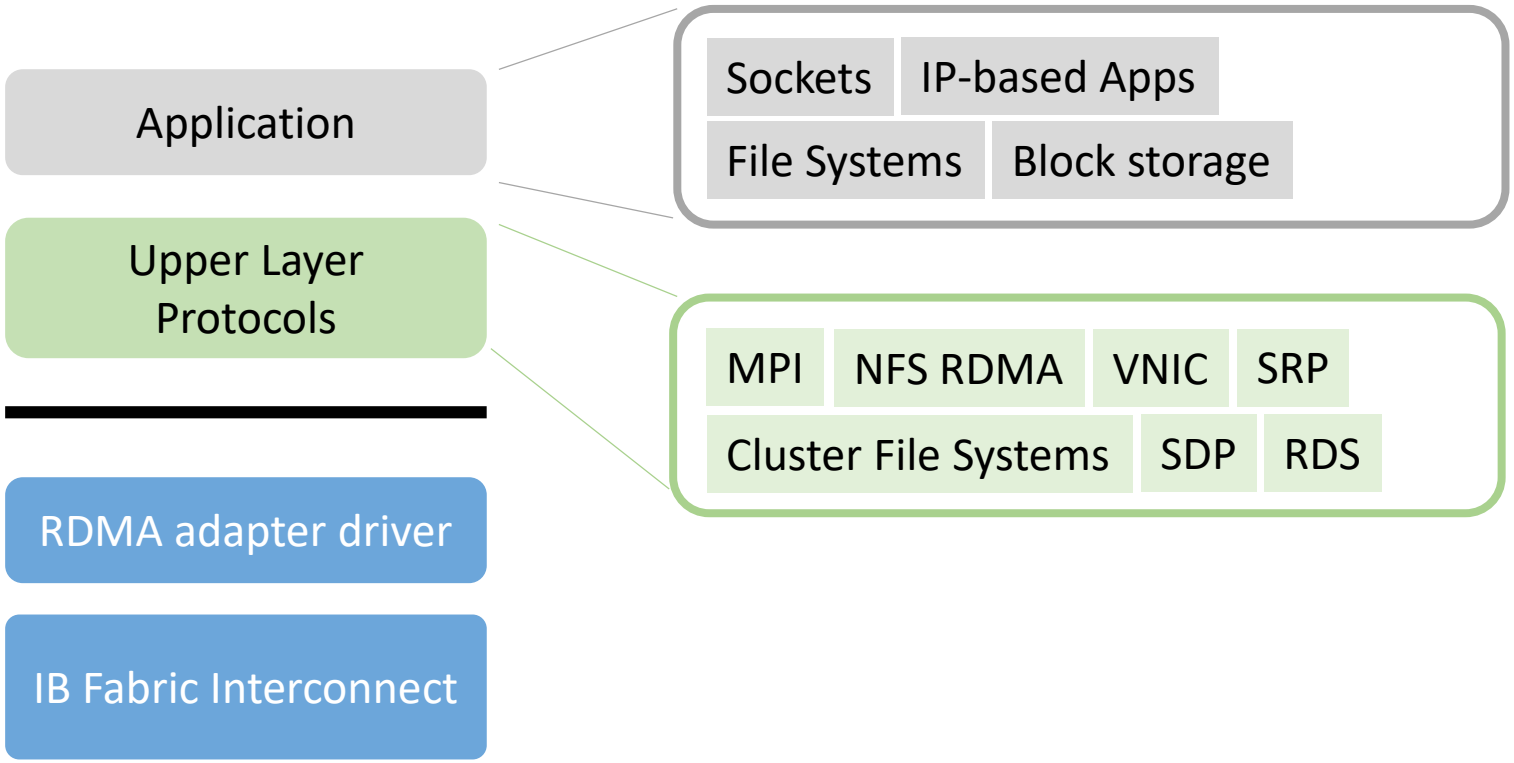




src: https://blog.zhaw.ch/icclab/infiniband-an-introduction-simple-ib-verbs-program-with-rdma-write/

Challenges of using RDMA

Added extra complexity for the developer to use the Verbs API



src: InfiniBand Trade Association: Introduction to IB for end users

MPI : Message Passing Interface

- Widely used in HPC
- Example: OpenMPI, MVAPICH, Intel MPI, etc.

File Systems:

- Lustre – parallel distributed FS for Linux
- NFS_RDMA – Network FS over RDMA

RDMA References

- IB trade introduction https://cw.infinibandta.org/document/dl/7268
- First steps for programming with IB verbs https://thegeekinthecorner.wordpress.com/2010/08/13/building-an-rdma-capable-application-with-ib-verbs-part-1-basics/
- Figures from https://zcopy.wordpress.com/category/getting-started/
- More details at http://www.mellanox.com/related-docs/prod_software/RDMA_Aware_Programming_user_manual.pdf

Overview of our new EDR cluster

- **EDR InfiniBand**
- **36-port** Mellanox switch
- **18 nodes cluster** (EDR NICs)
- **1 server with 4 Xeon E5-5660 v4 processors:**
  - 64 cores (128 with HT enabled)
  - 512 GB RAM
  - 2 EDR NICs, 1 x 10G NIC, 1 x 1G NIC
- **8 servers with 2 Xeon E5-2630 v4 processors:**
  - 20 cores (40 with HT enabled)
  - 32 GB RAM
  - 2 EDR NICs