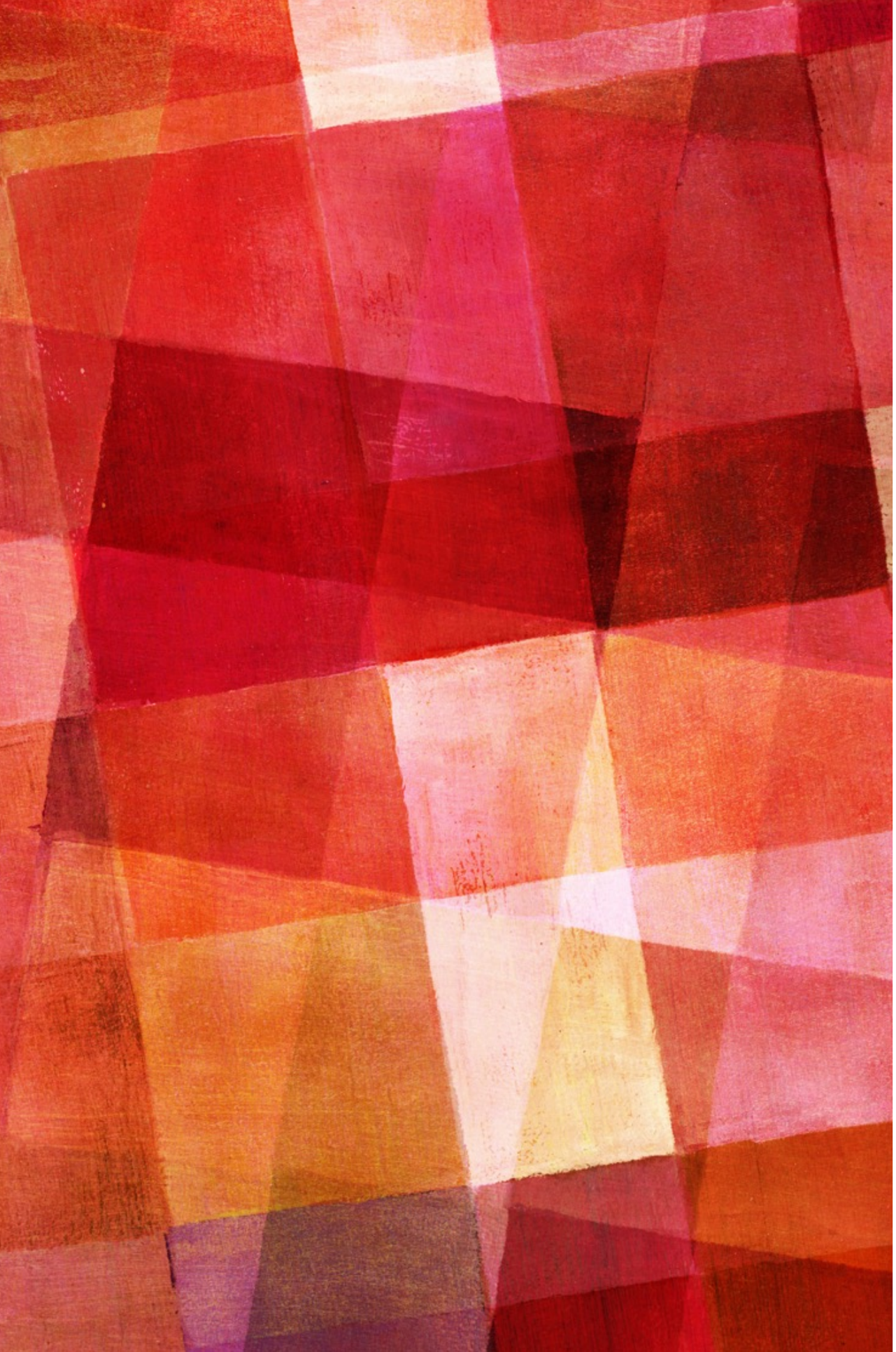
# NFS/RDMA BASICS

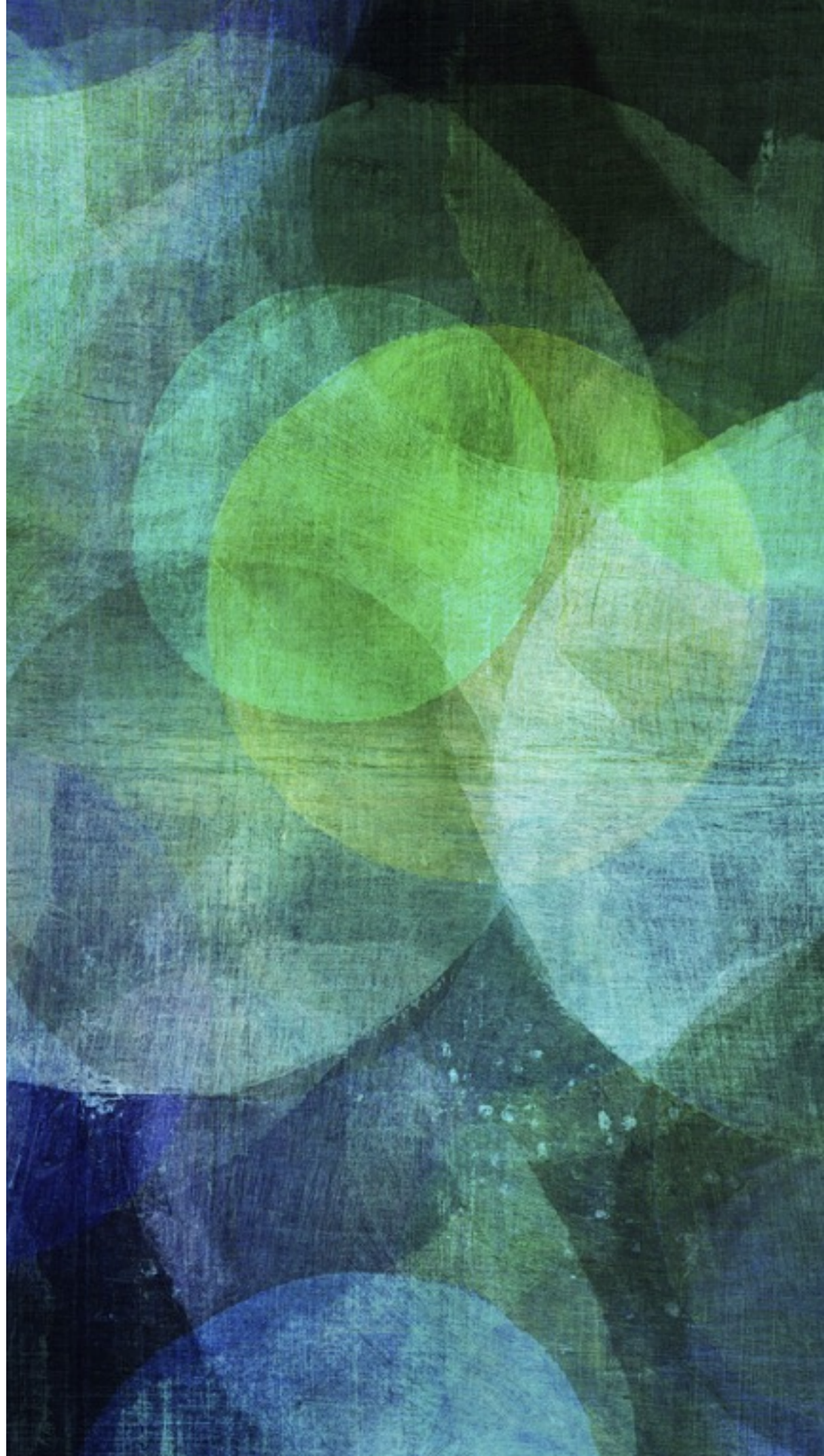
*Part One – The Whys and Wherefores*



# THE WHYS AND WHEREFORES

- Why RDMA?
- The RDMA Verbs API
- RDMA-enabled storage protocols

# WHY RDMA?



# DMA VERSUS RDMA

- *Direct Memory Access* (DMA) – A device transfers data directly to or from host memory
- *Remote Direct Memory Access* (RDMA) – A device transfers data directly between host memory and memory on other hosts on a network fabric

- *NFS/RDMA enables the data payloads of NFS READ and WRITE operations to be transferred between file server and client memory via a third party*

# DIRECT DATA PLACEMENT (DDP)

- Typical DMA networking
  - Data payload alignment can be ideal on the sender
  - The receiver often needs to pull-up incoming payloads
- RDMA
  - Data payload alignment can be made ideal on both the sender and receiver as part of the I/O operation

# OFFLOADED DATA TRANSFER

- Physical HCA
  - ULP and HCA driver set up payload transfer
  - RNICs move payload bytes

- Neither host CPU needs to touch payload bytes during data transfer (important on storage servers, since they are not likely to perform any computation on the payload)
- In a VM guest, the hypervisor is not involved in data transfer. With SR-IOV, data transfer can be handled safely by hardware.

# NETWORK PERFORMANCE CURVE

| InfiniBand technology | Effective throughput (Gb/s) | Adapter latency (microseconds) |
|---|---|---|
| Quad Data Rate | 32 | 1.3 |
| FDR10 | 40 | 0.7 |
| Fourteen Data Rate | 56 | 0.7 |
| Enhanced Data Rate | 100 | 0.5 |
| PCIe gen 3.0 x16 | 120 | |
| High Data Rate | 200 | |

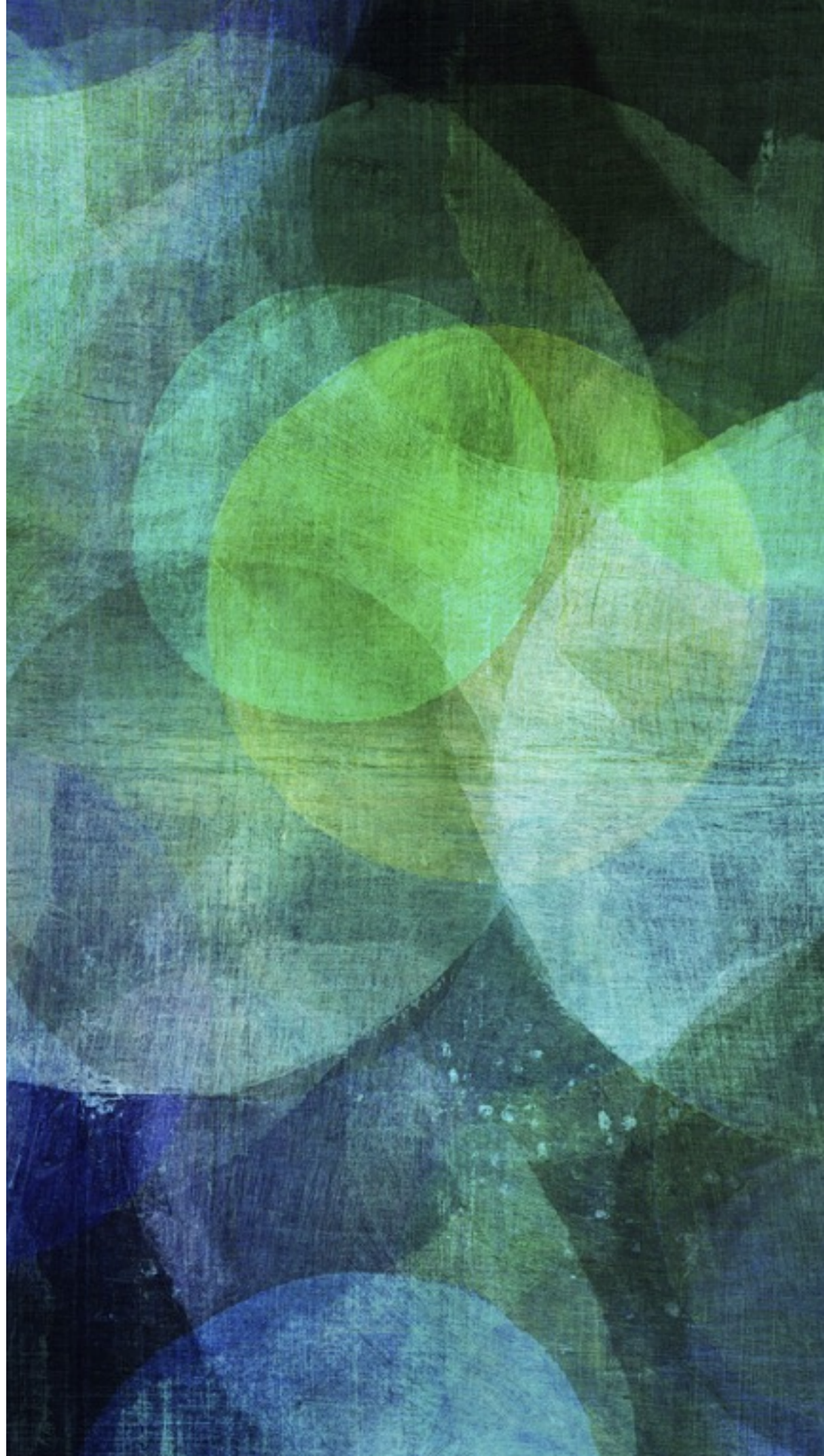
# STORAGE PERFORMANCE CURVE

| Storage technology | Relative latency |
|---|---|
| Hard drive | 3000 us |
| SAS SSD | 300 us |
| PCIe SSD | 30 us |
| Persistent memory | 3 us |
| DRAM | 0.3 us |

# FAST STORAGE AND FAST NETWORKING

- Immediate benefits for NFS:
  - Lower cache-to-cache transfer latency
  - Lower latency of read-only metadata operations
  - Better scaling of data throughput to CPU utilization
- Challenges
  - The I/O round trip time is now as fast as (or faster than) the elapsed CPU time to set it up

# THE RDMA VERBS API



# SOCKETS VERSUS VERBS

- A “verb” is implemented in the Linux kernel as a function call with standardized behavior and side effects.
- As with sockets, programs use the same API for different network fabrics: IB, RoCE, iWARP, Omni-Path.
- RDMA verbs introduce some additional capabilities:
  - Asynchrony
  - Very low overhead
  - Kernel bypass
  - Direct data placement

# VERBS PROVIDERS AND CONSUMERS

- *Verbs consumer* – application software that uses the verbs API
  - Also known as an *Upper Layer Protocol* (ULP)

- *Verbs provider* – software that exposes the verbs API
  - Via a device driver for a specialized hardware
  - Via emulation on conventional network devices

# QUEUE PAIRS

- A *Queue Pair* is a virtual communication port
  - Similar to the socket abstraction, but more “hardware-y”
  - One QP on each endpoint of a connection
  - QP states: init, RTS, error, SQD
- One QP consists of:
  - One *Send Work Queue* – work items are handled in order
  - One *Receive Work Queue* – same order as remote Sends
- Queues contain a fixed number of *Work Queue Entries* (WQEs)

# COMPLETION SIGNALING

- *Work Request* (WR) success may be signaled or silent
- WR errors are always reported via a completion

- A *Completion Queue* notifies ULP of Work Request completions
  - Send, RDMA Read or Write, memory registration
  - Receive – arrival of channel data
  - A CQ is associated with one or more QPs
  - CQs contain a fixed number of Completion Queue Entries (CQEs)

# MEMORY PROTECTION

- We could allow remotes to have read and write access to all of local memory, but this is obviously not desirable in multi-user and multi-process OS environments.
- *Memory registration* is a mechanism that makes a narrow part of memory temporarily visible to local and remote HCAs.
- The protection goals are:
  - To expose only the part of memory involved in the current transaction
  - To expose this memory only for the amount of time it takes for that transaction to complete

# MEMORY REGISTRATION AND INVALIDATION

- *Memory Region* – A set of host memory locations that have been registered

- *Registration* – The HCA assigns a memory key with specific access rights to a memory region on the local host

- *Invalidation* – A previously registered memory key and its associated access rights are made no longer valid

- *Protection Domain* – Security context that binds memory regions to QPs, controlling HCA access to host memory

# MEMORY ACCESS RIGHTS

- Assigned by registration, revoked by invalidation
- A combination of read, write, local, and remote rights
- Two types of memory keys:
  - A *local key* (Lkey) allows the local HCA to access local memory; used for RDMA Send or Receive; not shared with other hosts.
  - A *remote key* (Rkey) allows a remote HCA to access local memory; used for RDMA Read or Write; shared with other hosts.

# REGISTRATION METHODS IN THE KERNEL

- DMA key
  - Two keys per protection domain cover all local memory: one for local access, one for remote
- *Fast Memory Registration* (FMR)
  - Synchronous verbs: map_phys_mr and unmap_fmr
- *FastReg Work Request* (FRWR)
  - Registration performed by work requests posted on Send Queue, thus they complete asynchronously
  - FastRegister and LocalInvalidate WRs
  - Remote invalidation also supported

# COMMUNICATION MANAGEMENT

- Establish a connection
  - Resolve Upper Layer addresses to fabric endpoint addresses
  - CREQ, CREP, RTU (ready-to-use)
  - Request a service type
    - Connected mode: RC, XRC, UC
    - Datagram: UD, RD
- Release connection state and resources
  - DREQ, DREP

# CHANNEL SEND AND RECEIVE

- RDMA Send
  - Transfers data from an untagged local buffer to an arbitrary untagged remote buffer
  - Completes when local HCA is done with the buffer
    - Previously posted operations must not execute after a Send
- RDMA Receive
  - Prepares an untagged local buffer to receive ingress Send data
  - Completes when local HCA has filled the buffer with data sent from a peer
- *No co-ordination between Receive and Send*

# RDMA READ

- Pulls data from a tagged memory region on a remote host into a local memory region
- The Read response, carrying the data, is the remote's ACK
- Read completion signaled on local host when HCA is finished transferring all data in the request
- No completion signaled on remote host (local host must notify the remote when it is finished)

# RDMA READ CAVEATS

- HCA has limited Read responder resources
  - Only a few Reads maybe be processed at a time
  - Less change of overrun

- Remote requires notification when RDMA Reads are complete so that memory can be invalidated

# RDMA WRITE

- Pushes data from a local memory region to a tagged memory region on a remote host
- Write ACK means that the remote HCA has the data payload. Other rules determine when that payload arrives in the remote's memory.
- Write completion is signaled on local host when HCA has finished with the Write source buffers
- No completion signaled on remote host (local host must notify the remote when it is finished)

# RDMA WRITE CAVEATS

- RDMA Write is “fire and forget.” If there is a problem on the receiver, it is reported later.
- RDMA Writes can be streamed with each other or with Send
- No limit on number of outstanding Writes
  - But an HCA can be overrun, resulting in global pause frames
- Receive completion on remote implies data from previous RDMA Writes is placed, and memory can be invalidated

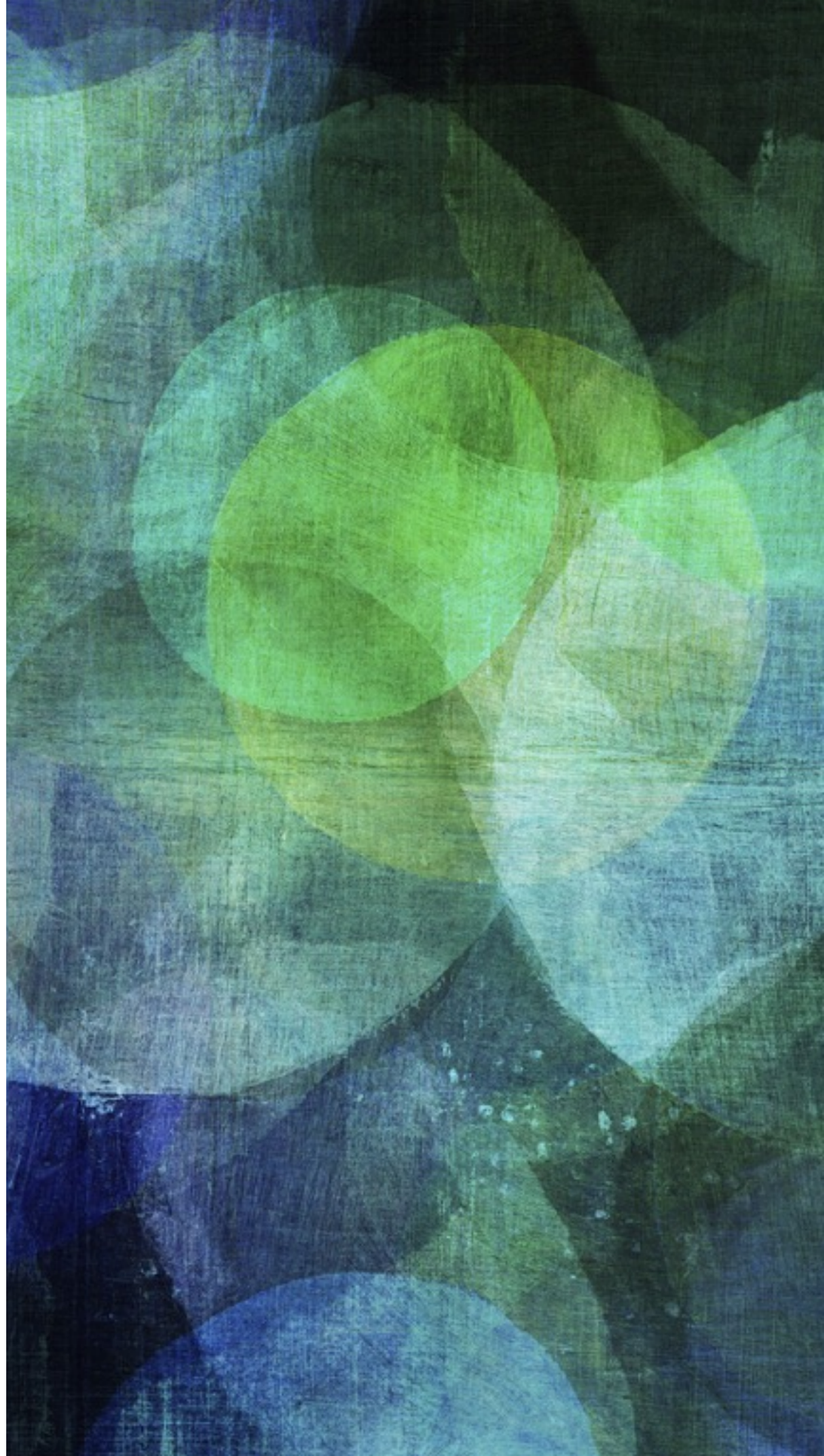
# ORDERING OF OPERATIONS

- *Data placement* – Writing data into memory. Can occur in any order (but before delivery).
- *Data delivery* – Notifying the ULP that a message is available.
- For example, when the local host posts RDMA Write followed by a Send:
  - At the local host, a Send completion means previous RDMA Writes have also completed. The remote HCA has received the payload, but has not written it to memory.
  - At the remote host, a Receive completion means previous RDMA Writes have been written to memory.

# REMOTE INVALIDATION

- Using a special form of RDMA Send, one side can request that a remote HCA perform an invalidation on a memory key
  - The RDMA Send request contains an additional header that contains an Rkey to be invalidated
  - The remote HCA does not have to invalidate that Rkey
- If it does, the Receive completion for that ingress message carries the Rkey as notification to the ULP of the invalidation. The ULP must not invalidate that Rkey again.
- A remotely invalidated Rkey saves the ULP the cost of invalidating that key. DMA unmapping is still required.

# RDMA-ENABLED STORAGE PROTOCOLS



# RDMA-ENABLED STORAGE PROTOCOLS

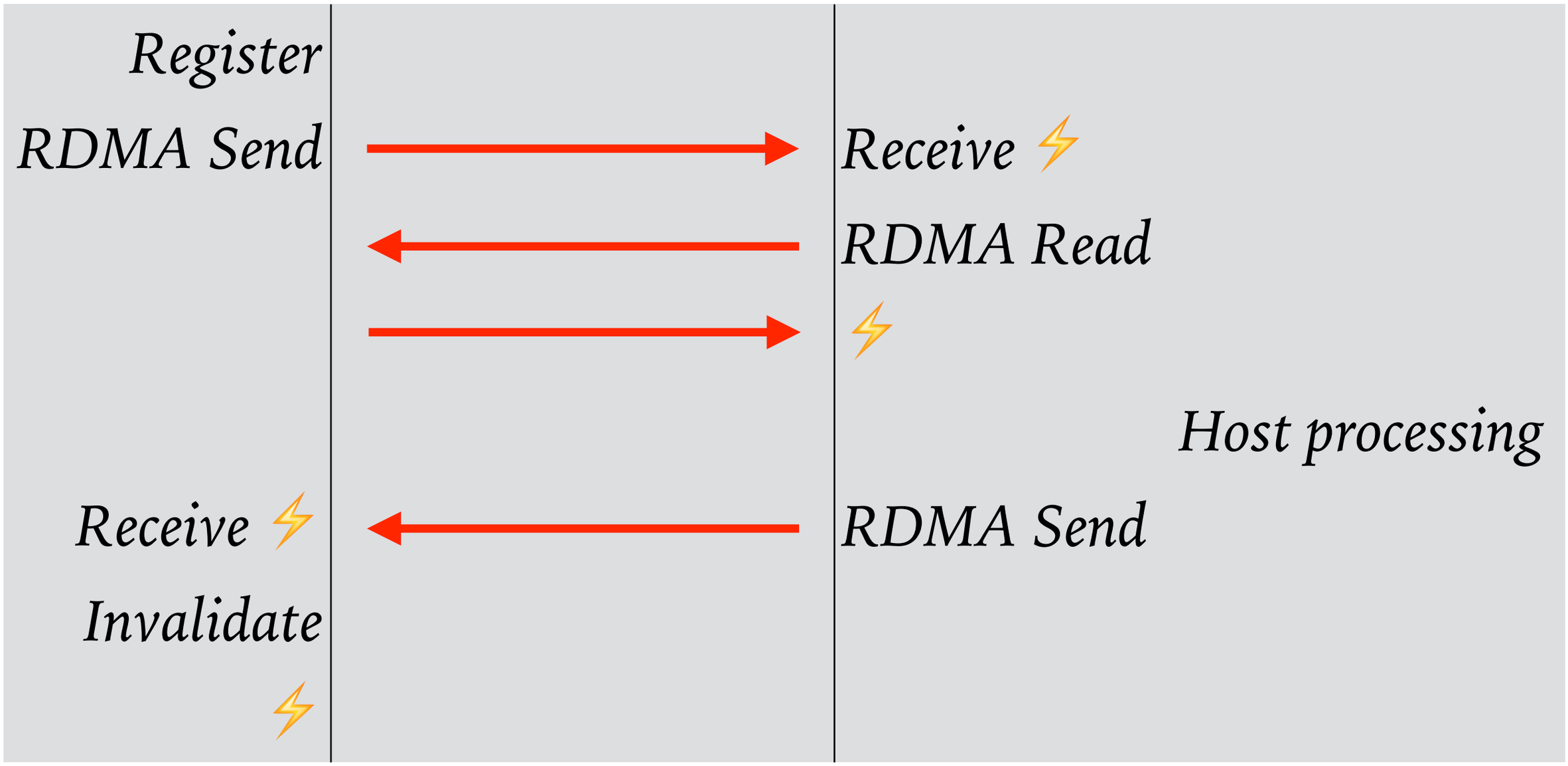
- Block protocols
  - SRP – SCSI RDMA Protocol (ANSI INCITS 365-2007)
  - iSER – iSCSI Extensions for RDMA (RFC 7145)
  - NVMe/F – NVM Express over Fabrics (revision 1.3)
- File protocols
  - SMB Direct in Windows Server (MS-SMB)
  - NFS over RDMA (RFC 5667)

# TRANSFER MODELS

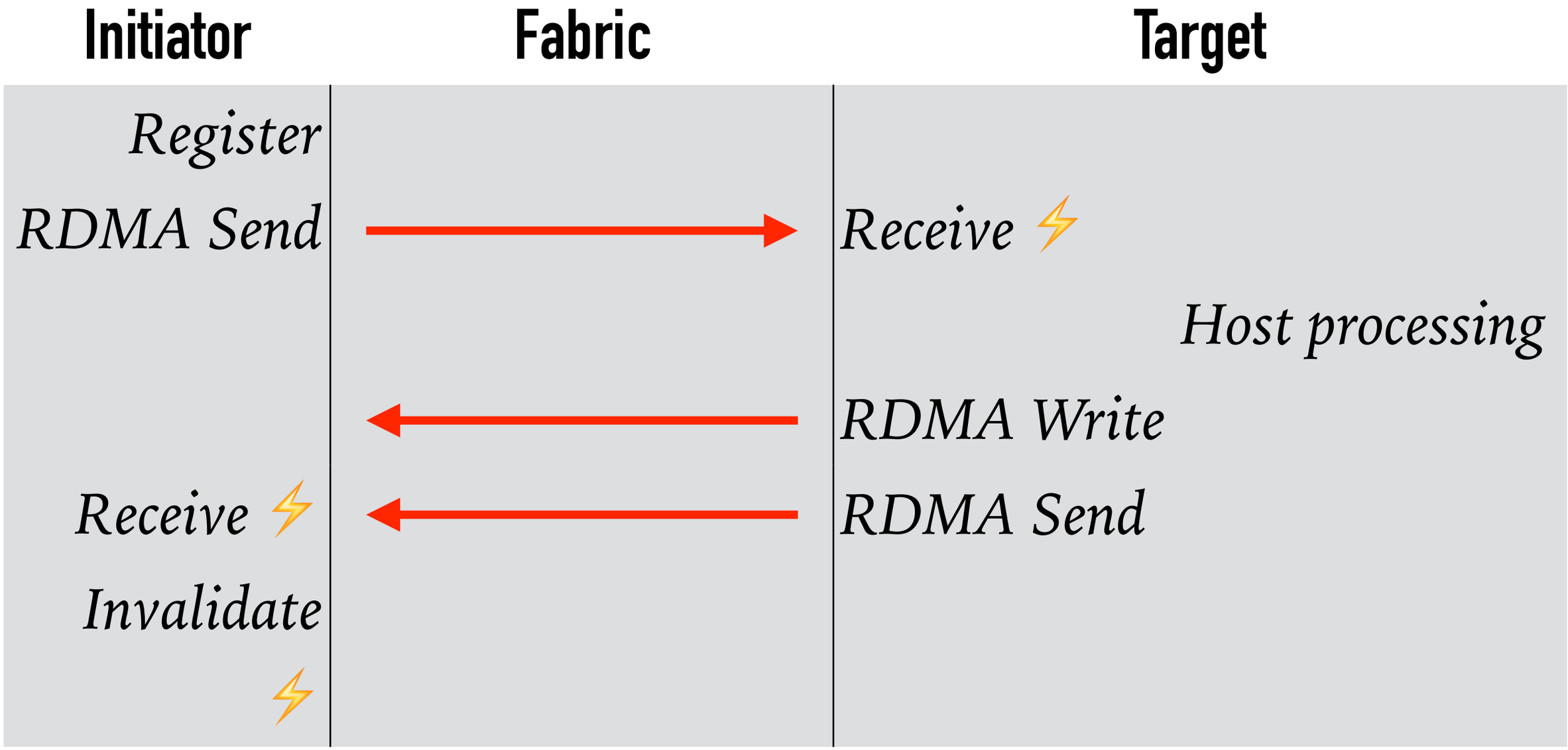
- *Read-Read*
  - Initiator exposes memory
  - Target pulls arguments
  - Target exposes memory
  - Initiator pulls results

- *Read-Write*
  - Initiator exposes memory
  - Target pulls arguments, pushes results

# RDMA READ IN USE



# RDMA WRITE IN USE



# THE COST OF MEMORY REGISTRATION

- RDMA-enabled storage implementations are all about mitigating the cost of registering and invalidating memory
  - Sending a request message has to wait for memory registration to complete
  - An upper layer transaction must not complete until remotely exposed memory regions have been invalidated

- Registration costs are amortized with large payloads, but small-to-moderate I/O resides in a donut hole
  - Trade-off between speed of host CPU data copy versus the expense of building and tearing down memory registration