InfiniBand Software Architecture and RDMA

David A. Deming
Solution Technology

Topics

- Documentation and Trade Association
- IB architectural components
- IB protocol layers
- IB enhancements
- Physical layer
- Link layer
- Network layer
- Transport layer
- ULP
- Software Interface
- Management

Upper Layer Protocol

Software interface

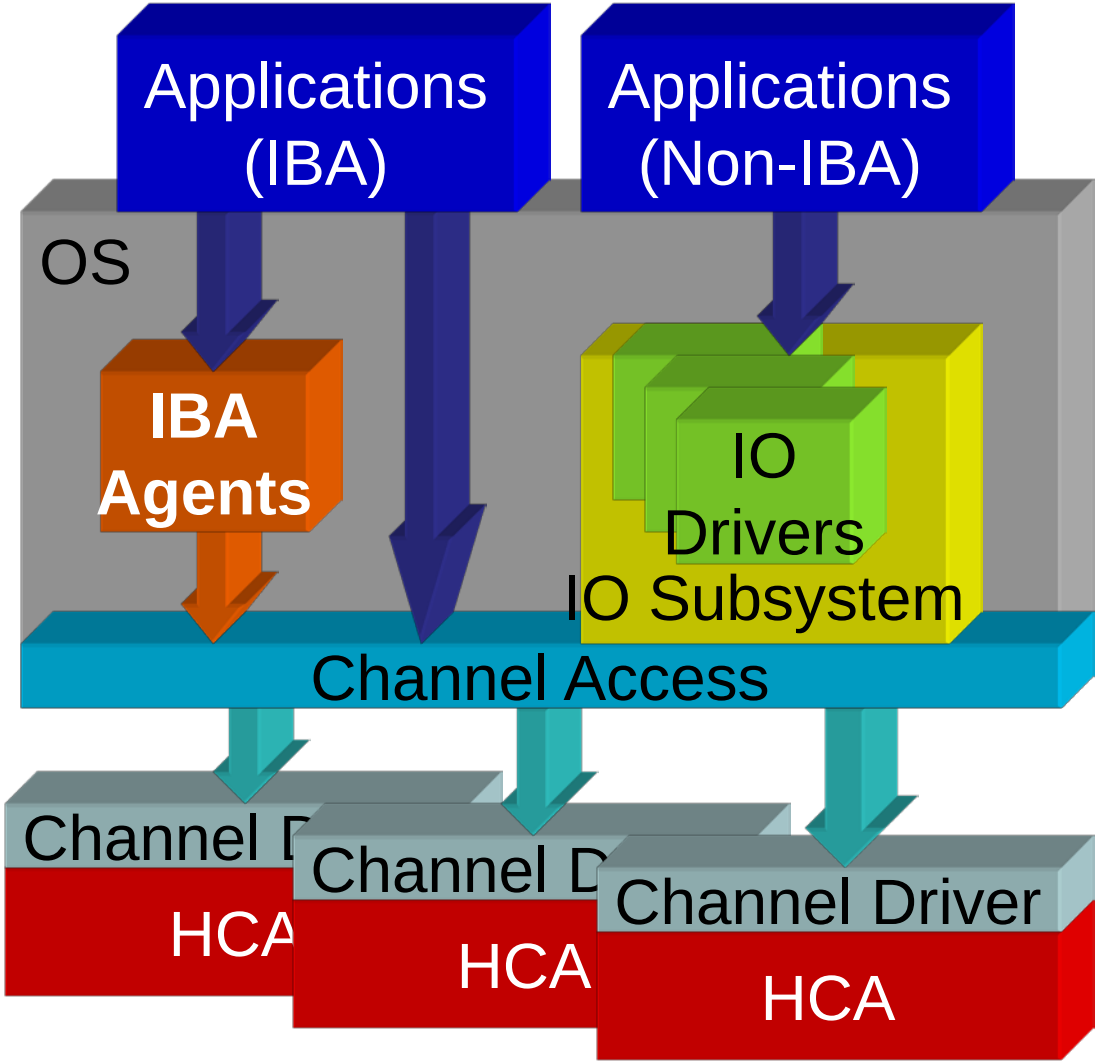
Processor Nodes

- Consumers
  - Optimized for units that contain multiple independent processes and threads
  - Translation – multiple concurrent independent I/O processes can be executed by a processor node
- Message and Data Service
  - Outside the scope of the IBA architecture
- IBA specifies the semantic interface between the message and data service and a channel adapter
  - The semantic interface is referred to as *IBA Verbs*

IB Software Transport Interface

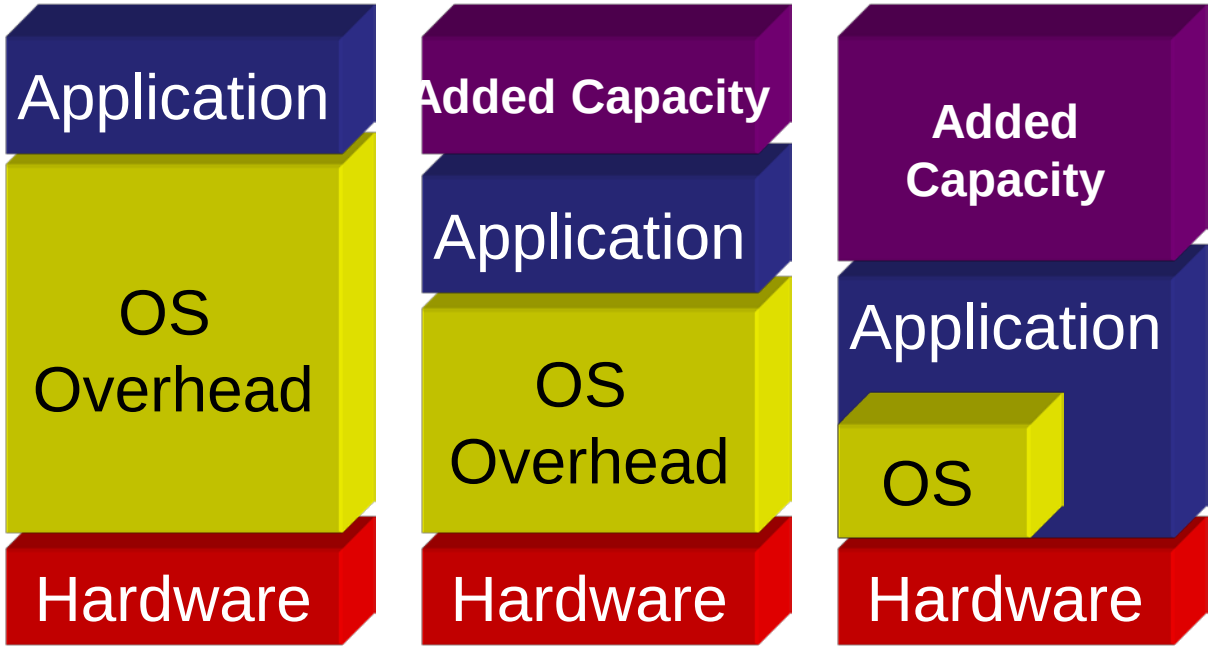
- Describes the behavior of InfiniBand hardware and software as viewed by the host
  - Not the host itself
- Channel Interface (CI)
  - A combination of hardware, firmware, and software which provides services to the host
- Verbs
  - Operations which a CI is expected to perform
  - Create, Modify, Query, and Destroy
  - Queue Pairs, Work Requests, Completion Queues
- Just as with traditional I/O buses, operations are defined, but software interface to those operations may vary

Host Environment



- IBA Applications
  - IPC clients
  - IBA services
- IBA Agents
  - IBA specific services
    - Communication and Resource Management
    - Subnet Management Agent
- IO Drivers/Subsystem
  - Controls non-IBA hardware attached via IBA fabric
- Channel Access
  - OSV Provided
  - Single interface to all users
- Channel Driver
  - Provided by HCA vendor

Conceptual Performance Benefits



Traditional HW/SW Stack

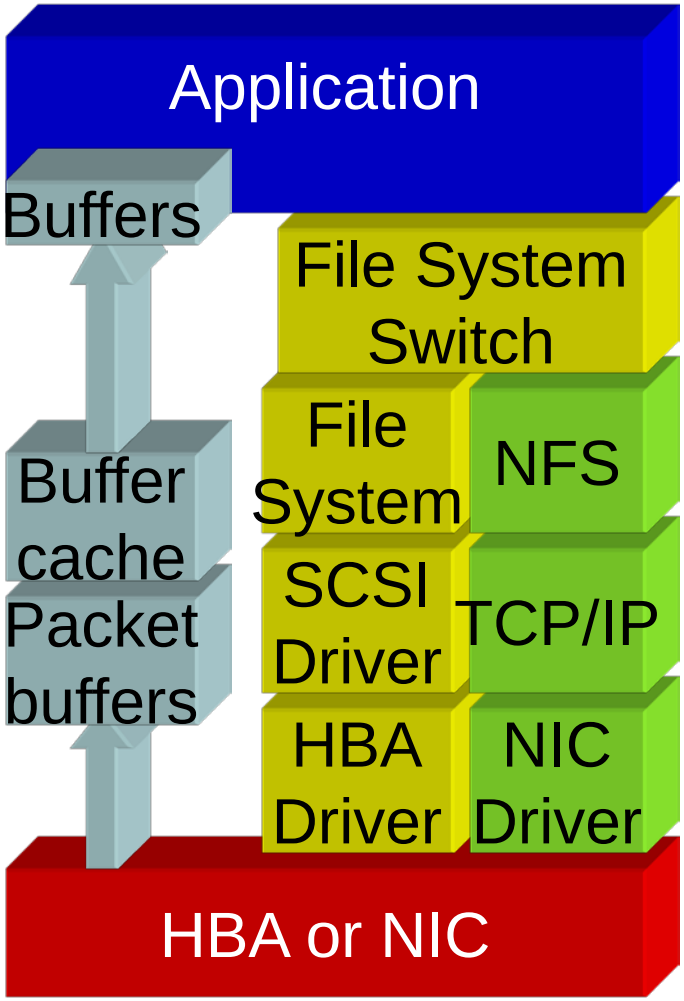
Hardware Acceleration

InfiniBand HW/SW Stack

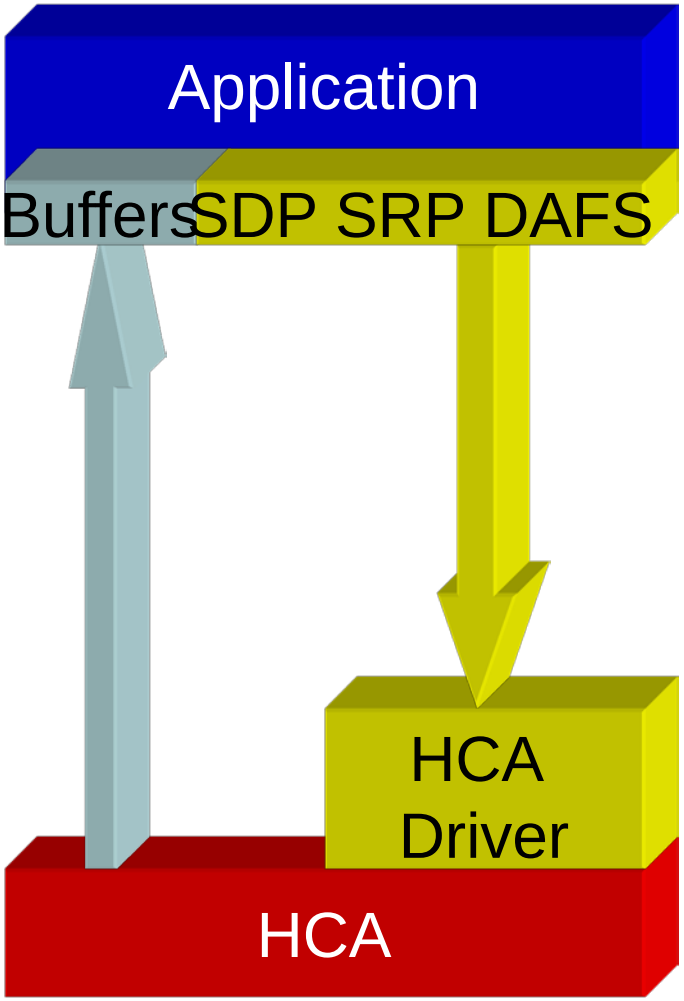
SCSI RDMA Protocol
Sockets Direct Protocol
Direct Access File System

Comparing File Access Methods

Traditional storage or network stack

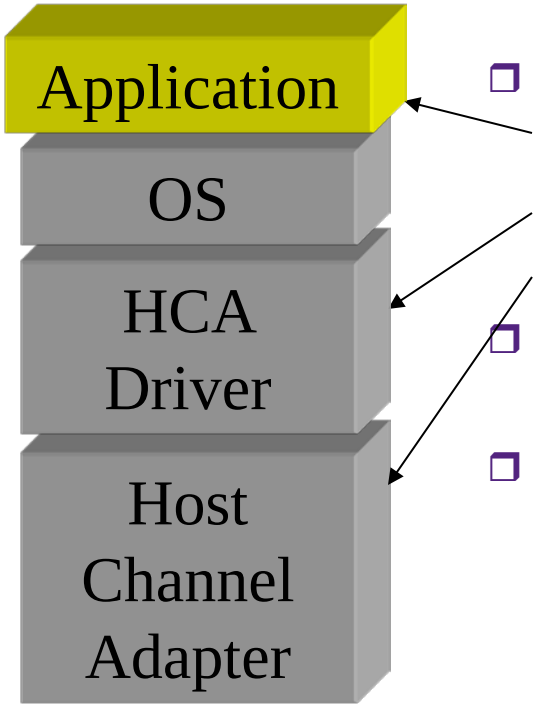


IBA storage stack



Verbs

6.5 Verbs Influence All Interfaces



- Verbs influence the
  - Application,
  - Driver,
  - and Hardware interfaces
- OS vendors can define their own interfaces, just as they define APIs and device driver interfaces
- Verbs allow for concurrent development of hardware and OS interfaces by multiple vendors
  - Intended to influence/guide design of hardware, device driver interfaces, Kernel Programming Interfaces (KPIs), and APIs
- Verb concept borrowed from VI architecture
- Transport Layer's ULP is defined for an HCA but undefined for a TCA

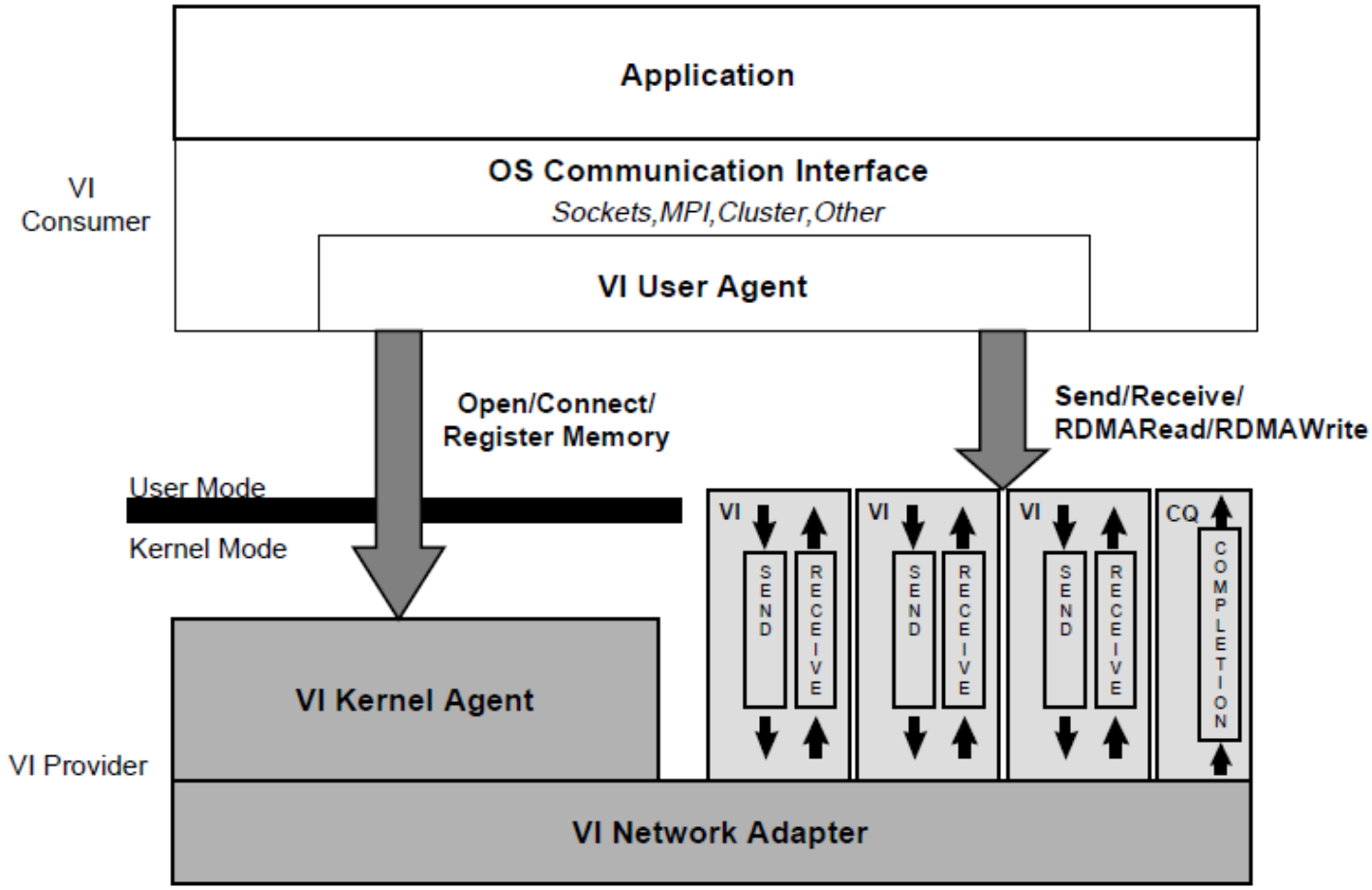
The Verbs and VI Architecture

- Many VI concepts used in InfiniBand™ Architecture and the Verbs in particular
- Goal: VI implementation on top of InfiniBand Architecture should be efficient
- However, there are important differences between the Verbs and VI behavior:
  - Enhanced features
  - Semantic changes (for instance, to allow more efficient implementations)
- Virtual Interface Architecture Specification
  - Compaq, Intel and Microsoft: December 16,1997
  - http://www.cs.uml.edu/~bill/cs520/VI_spec.pdf

Verb Concepts

- Host Channel Adapter (HCA)
  - Represents local channel interface (CI)
  - Also known as plain old channel adapter (CA) or even a Target Channel Adapter (TCA) if on receiving end
- Queue Pair (QP):
  - Represents communications endpoint, like a socket
  - Consists of a SEND Queue and a RECEIVE Queue
- Work Request Element (WQE): requests communications operation
- Completion Queue (CQ): provides completed operation status
- Memory Regions: system memory is “registered” to allow access to local and remote channel adapters to source/sink communications data

VI Architectural Model



Verb categories

Software transport verbs

- HCA
- HCA Resource
- Queue Pair
- Memory Management
- Multicast
- Processing
- Completion
- Event Handling

Verbs category example

- Queue Pairs
  - Create
  - Modify
  - Query
  - Destroy
- Work Requests
  - Post Send
  - Post Receive
- Completion Queue
  - Create
  - Query
  - Resize
  - Destroy
  - Poll for Completion
  - Request Completion Notification
  - Set Completion Event Handler

All IB verbs

HCA verbs

Open HCA
Query HCA
Modify HCA Attributes Access Violation Counters
Close HCA
Allocate Protection Domain
Deallocate Protection Domain
Allocate Reliable Datagram Domain RD Service
Deallocate Reliable Datagram Domain RD Service

HCA resource verbs

Create Address Handle
Modify Address Handle
Query Address Handle
Destroy Address Handle

Queue Pair verbs

Create Queue Pair
Modify Queue Pair
Query Queue Pair
Destroy Queue Pair
Get Special QP
Create Completion Queue
Query Completion Queue
Resize Completion Queue
Destroy Completion Queue
Create EE Context RD Service
Modify EE Context Attributes RD Service
Query EE Context RD Service
Destroy EE Context RD Service

Memory Management

Register Memory Region
Register Physical Memory Region
Query Memory Region
Deregister Memory Region
Reregister Memory Region
Reregister Physical Memory Region
Register Shared Memory Region
Allocate Memory Window
Query Memory Window
Bind Memory Window
Deallocate Memory Window

Multicast verbs

Attach QP to Multicast Group UD Multicast Service
Detach QP from Multicast Group UD Multicast Service

Processing verbs

Post Send Request
Post Receive Request

Completion Queue verbs

Poll for Completion
Request Completion Notification

Event handling

Set Completion Event Handler
Set Asynchronous Event Handler

NVM commands

Streamlined Command Set

Queue management

I/O operations RDMA R/W

Admin Commands for Queues & Transport

Admin Command	Description
Create I/O Submission Queue	Queue Management
Create I/O Completion Queue	
Delete I/O Submission Queue	
Delete I/O Completion Queue	
Abort	
Asynchronous Event Request	Status & Event Reporting
Get Log Page	
Identify	Configuration
Set Features	
Get Features	
(Optional) Firmware Activate	Firmware Management
(Optional) Firmware Image Download	
(Optional) Security Send	Security
(Optional) Security Receive	
(Optional) Format NVM	Namespace Management

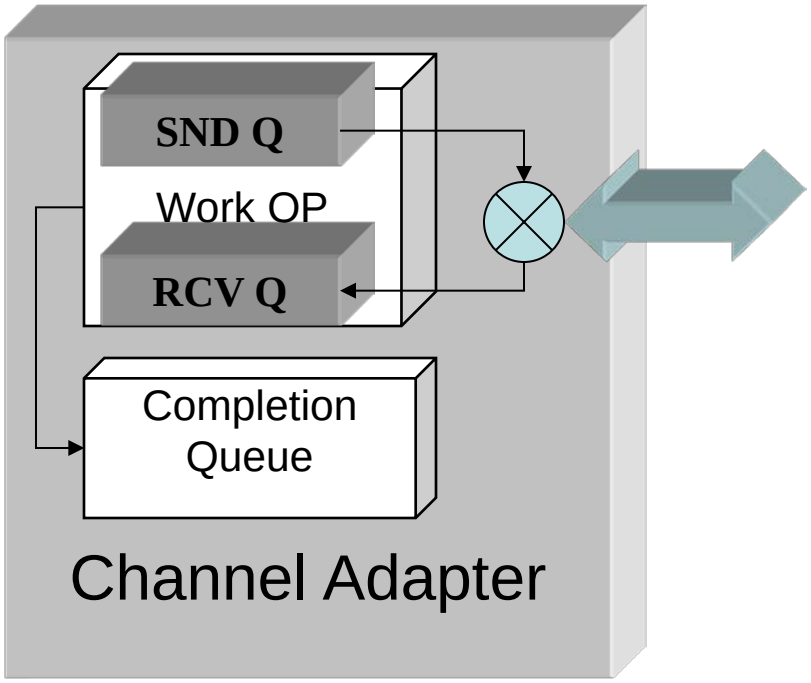
I/O Commands for SSD Functionality

NVM Command	Description
Flush	Data Ordering
Read	Data Transfer, Including end-to-end data protection & security
Write	
(Optional) Write Uncorrectable	
(Optional) Compare	
(Optional) Dataset Management	Data Usage Hints

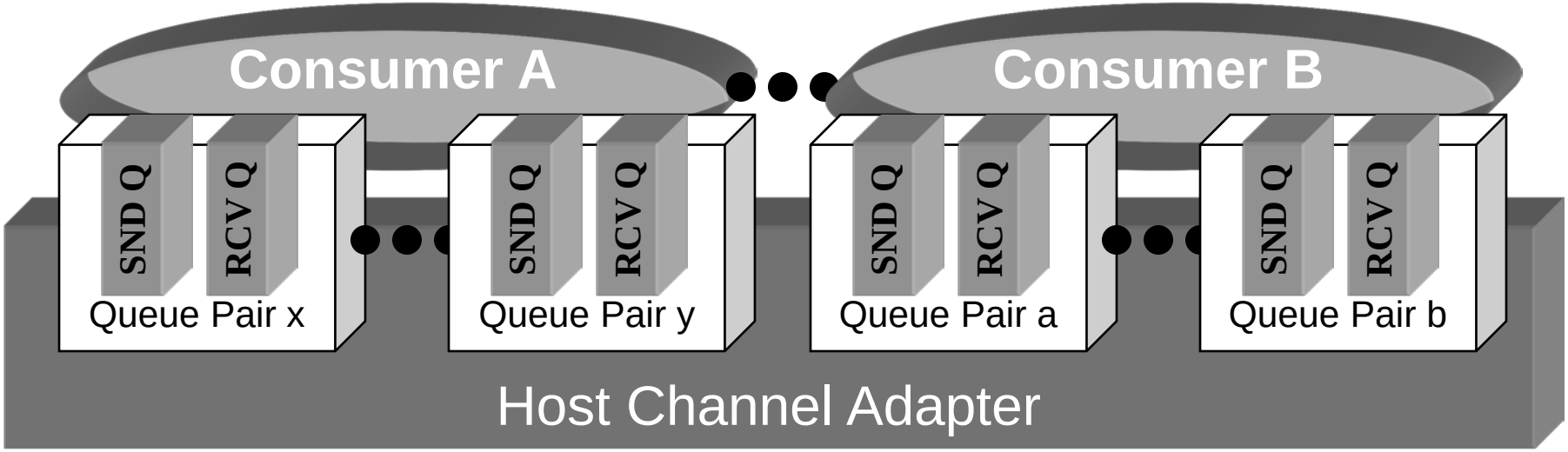
13 Required Commands Total (10 Admin, 3 I/O)

Work Queues

- Work Queue Pair (QP)
  - A Send Queue and a Receive Queue
  - Allocation and Control is on a Queue Pair basis
  - All traffic flows through Work Queues
- Completion Queue
  - Work Queues put results in an associated completion queue
  - Holds status of completed operations



QP Characteristics



- The QP is the virtual interface
  - That the hardware provides to an IBA consumer
  - It provides a virtual communication port for the consumer
- Architecture supports up to 2^{24} QPs per channel adapter
  - Operation of each QP is independent from the others
  - Each QP provides a high degree of isolation and protection from other QP operations and other consumers

Virtual Interface

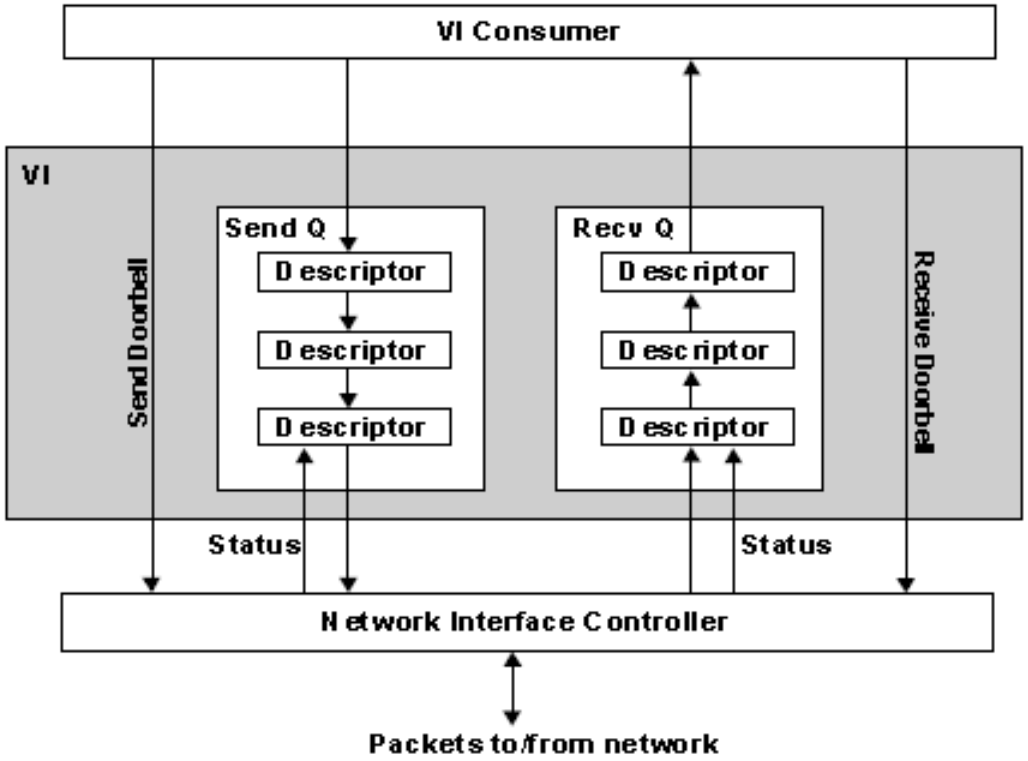
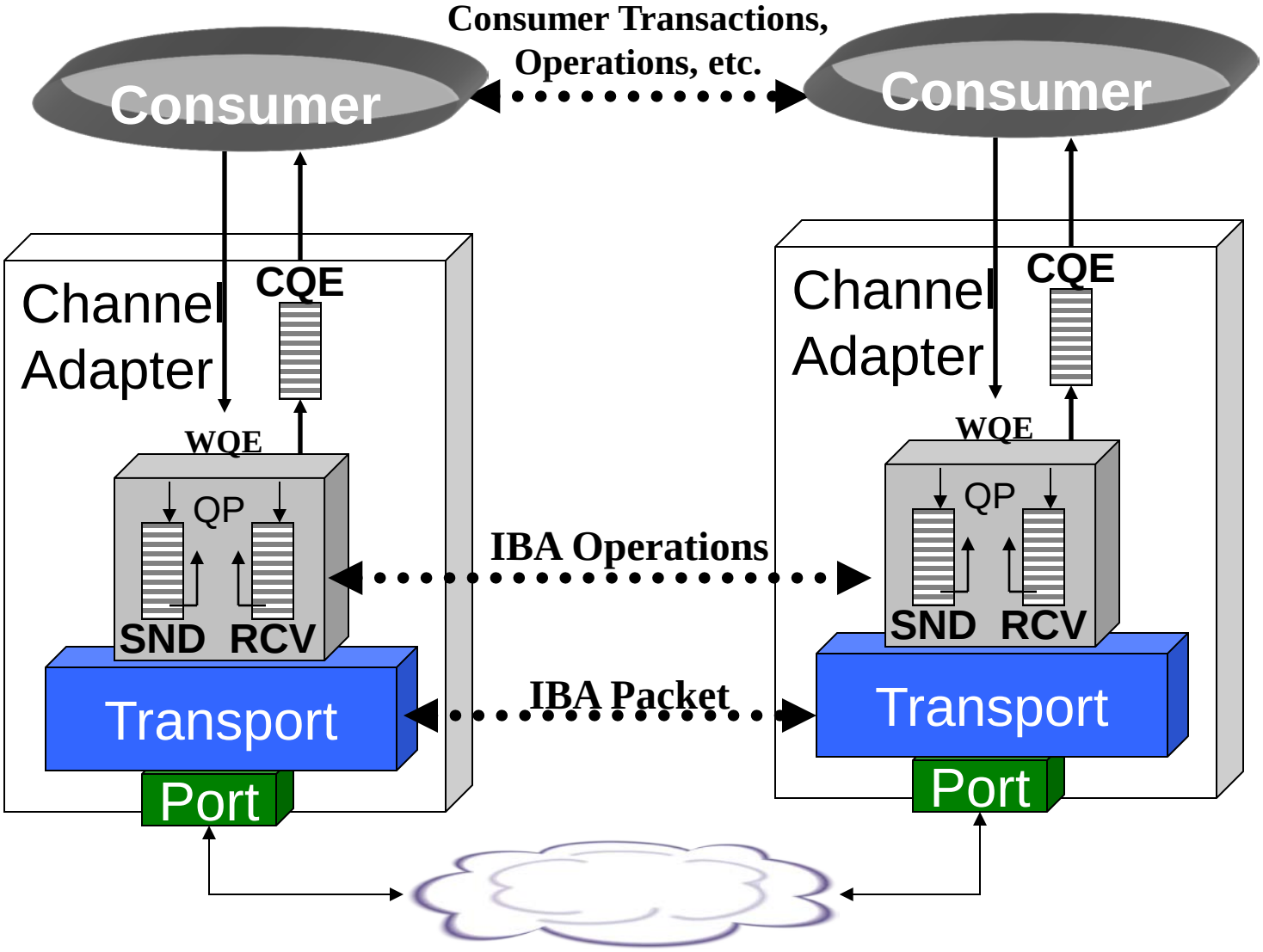


Figure 2: A Virtual Interface

Communications Stack



Open Fabric Enterprise Distribution

Working

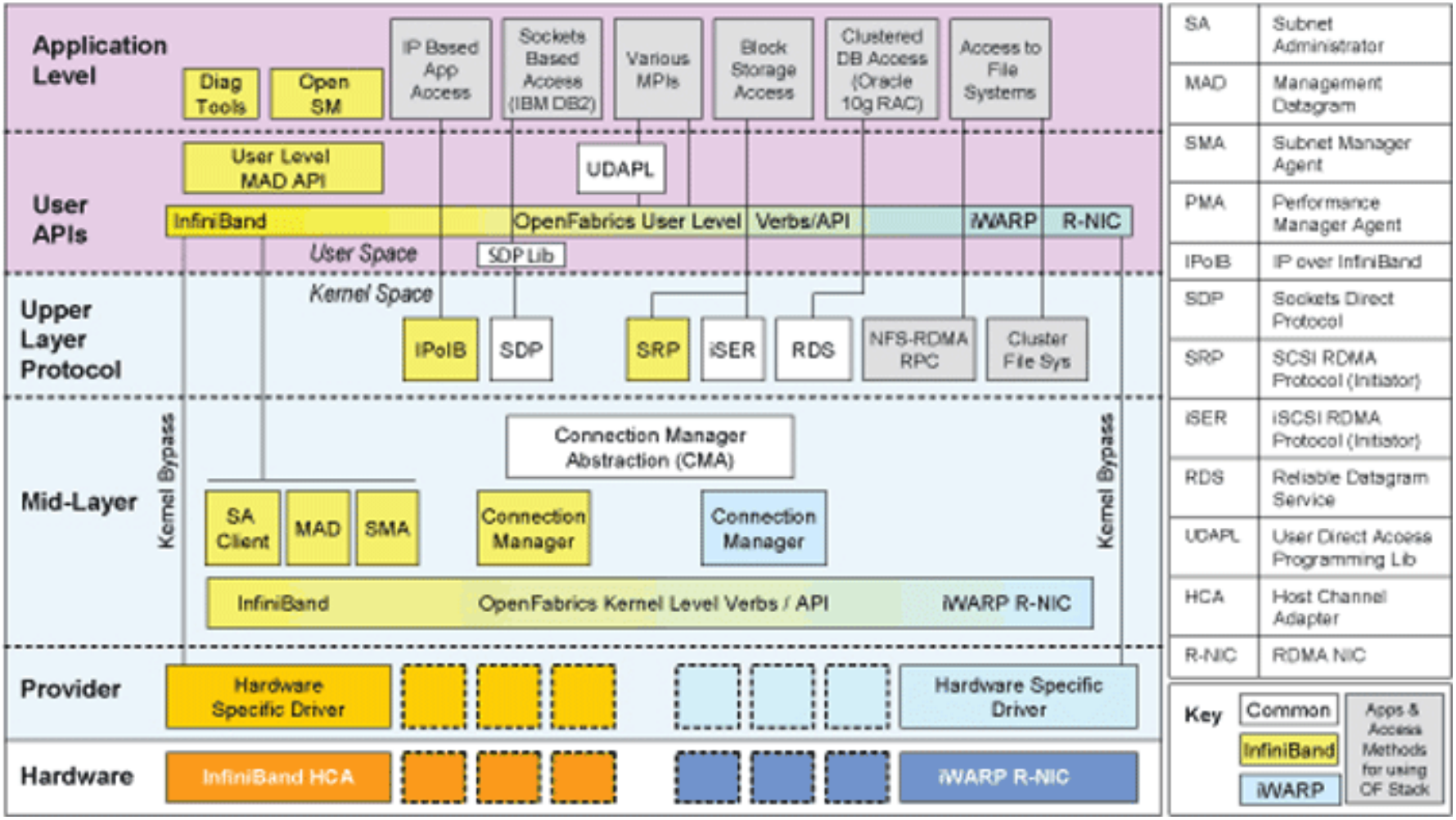
OFED (Open Fabric Enterprise Distribution)

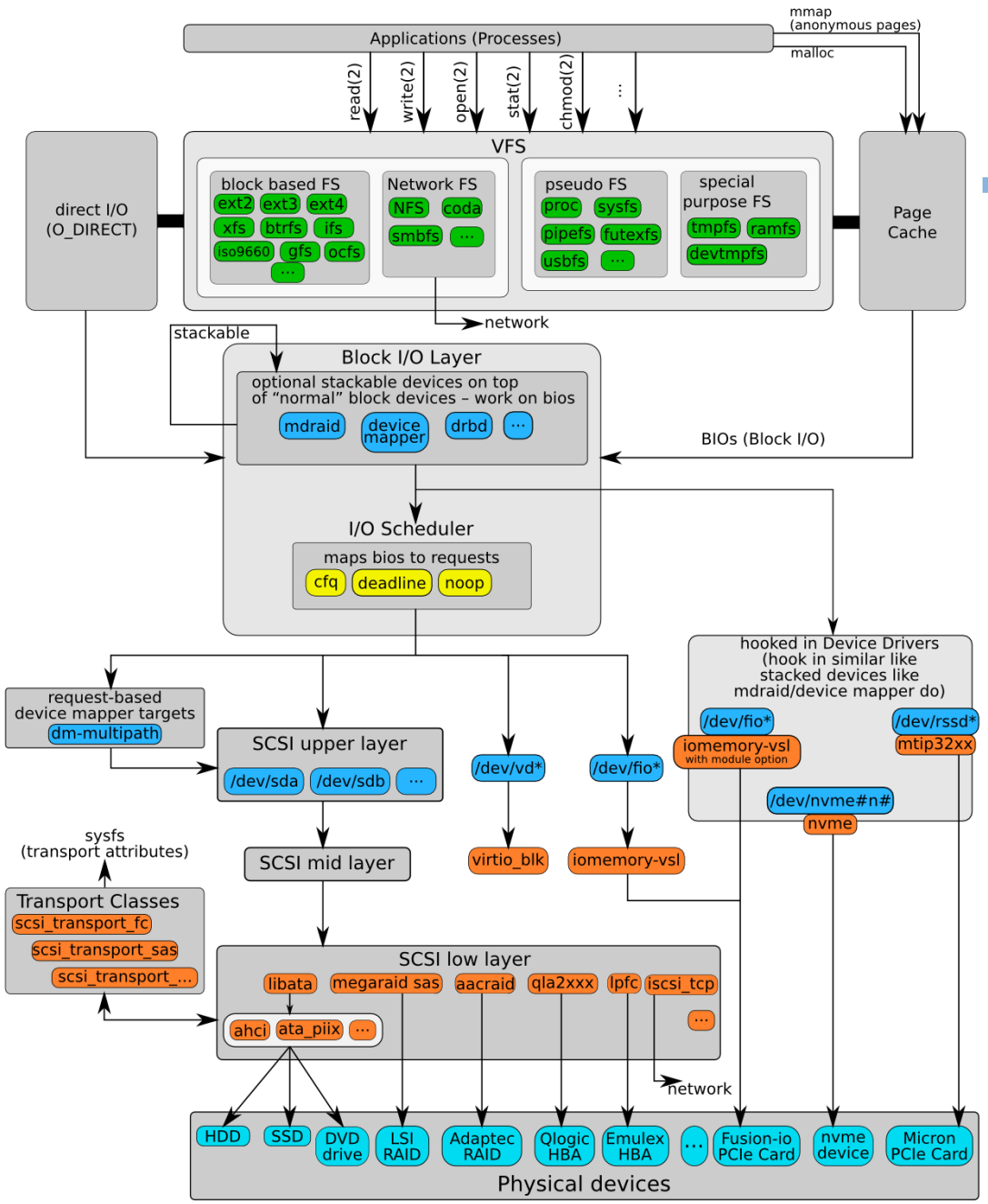
- OFED™ is open-source software for RDMA and kernel bypass applications.
- OFED is used in business, research and scientific environments that require highly efficient networks, storage connectivity and parallel computing.
- The software provides high performance computing sites and enterprise data centers with flexibility and investment protection as computing evolves towards applications that require extreme speeds, massive scalability and utility-class reliability.

OFED includes

1. kernel-level drivers,
2. channel-oriented RDMA and send/receive operations,
3. kernel bypasses of the operating system,
4. both kernel and user-level application programming interface (API) and services for parallel message passing (MPI),
5. sockets data exchange (e.g., RDS, SDP),
6. NAS and SAN storage (e.g. iSER, NFS-RDMA, SRP)
7. and file system/database systems.

OFED architecture

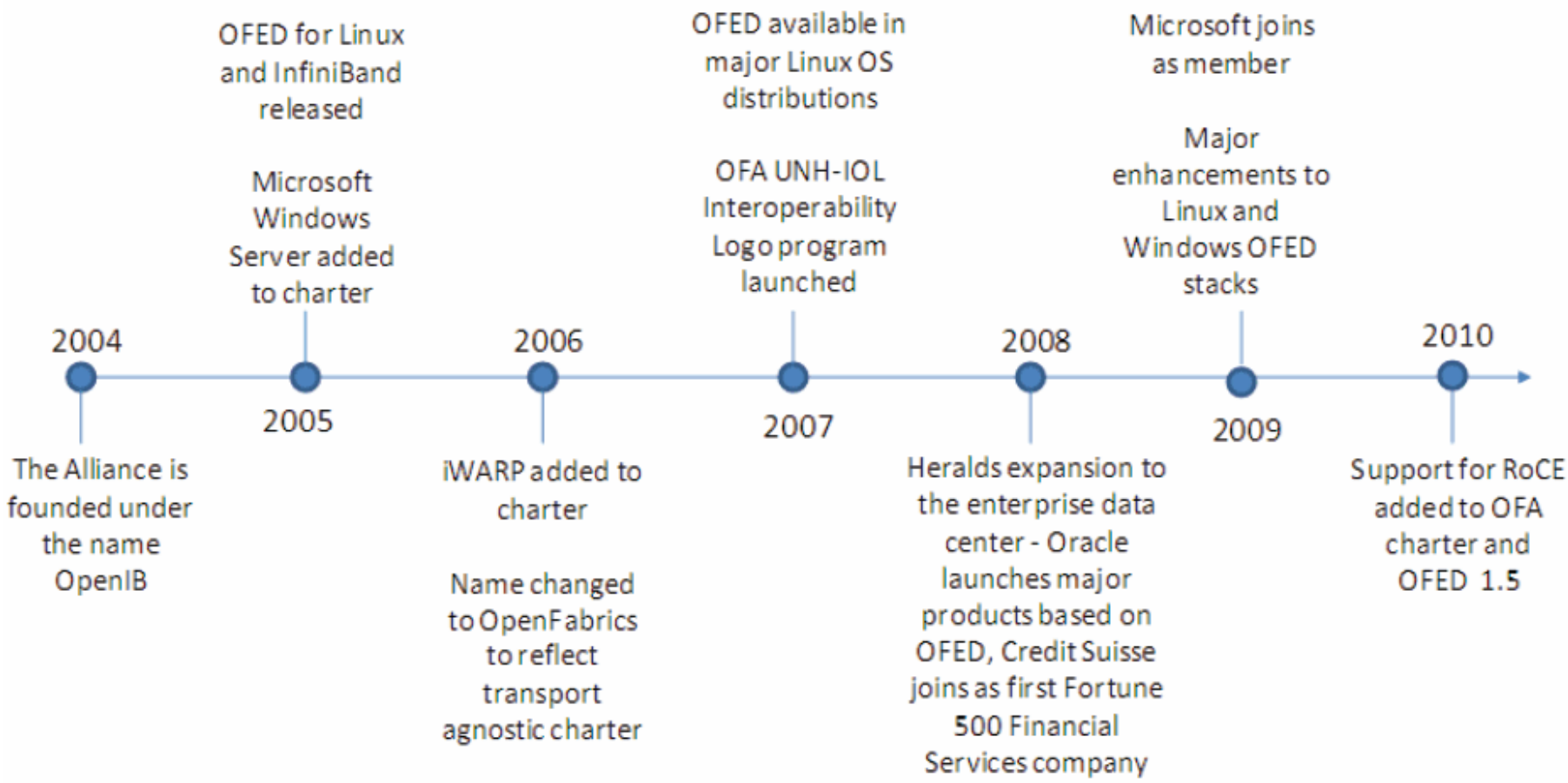




Operation system support

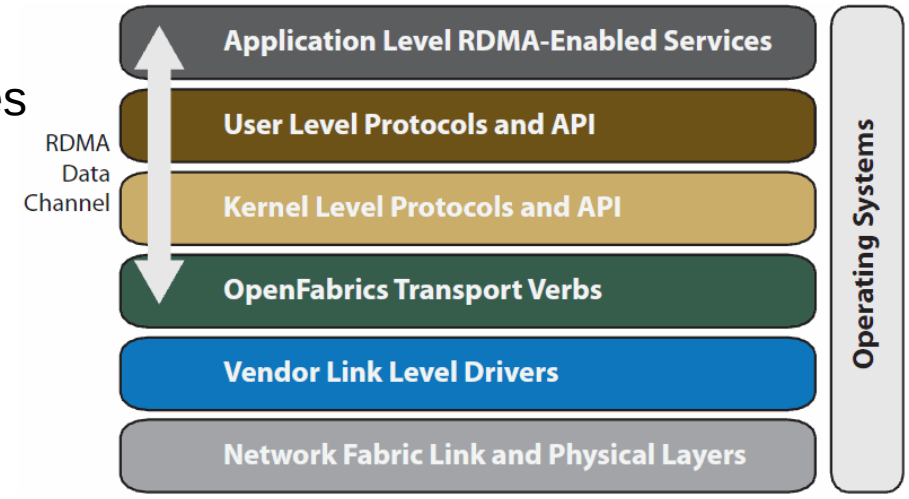
- OFED is available for many Linux and Windows distributions, including:
  - *Red Hat Enterprise Linux (RHEL)*
  - *Novell SUSE Linux Enterprise Distribution (SLES)*
  - *Oracle Enterprise Linux (OEL)*
  - *Microsoft Windows Server* operating systems
- The entire set of OpenFabrics Software
  - from which modules and patches are selected to form OFED releases
  - resides on the OpenFabrics servers and is available for download.

OFED for HPC

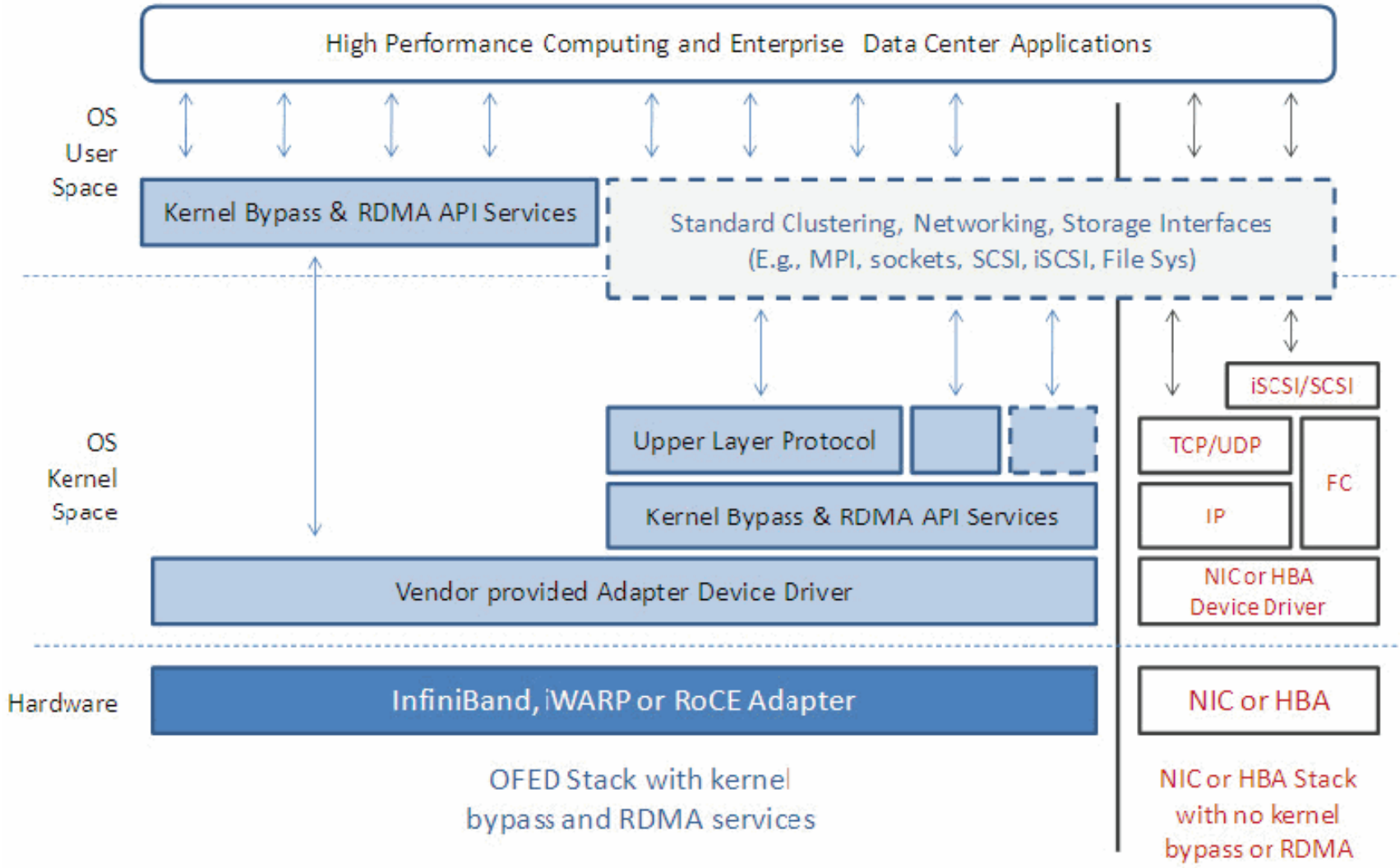


OFED Software Technology

- Architecturally, OFED implements:
  - a set of kernel bypass and remote DMA mechanisms for delivering the latency, utilization and bandwidth benefits for all data center traffic types (e.g. inter-processor, networking, clustering and storage)
- OFED goal: to deliver a set of APIs
  - both at kernel (called verbs) for maximum performance
  - and protocol level for excellent performance and application compatibility
  - in a way that is independent to the underlying interconnect and transport technology



OFED software stack



Kernel level ULPs and relevant applications

- SRP (SCSI RDMA Protocol): This storage protocol is available as both initiator and target components for OFED and is suitable for high performance block storage communications between servers and storage systems akin to Fiber Channel.
- iSER (iSCSI Extensions over RDMA): This storage protocol is available as an initiator component within the OFED package and the target component is available from several members of the OFA.
- RDS (Reliable Datagram Service): This low latency and high performance IPC (inter processor communication) and storage protocol is used:
- IPoIB (Internet Protocol over IB): This protocol enables any IP application to operate over InfiniBand without requiring any change.
- SDP (Sockets Direct Protocol): The purpose of SDP is to provide an RDMA accelerated alternative to the TCP protocol on IP.

Specialized ULPs

- Lustre Parallel File System: Lustre is the world's #1 parallel file system and is designed to enable I/O performance and scaling beyond the limits of traditional storage technology.
  - Often used in High Performance Computing environments with InfiniBand and OFED.
  - Lustre is also applicable to any enterprise storage environment where very high I/O bandwidth is required.
- PureScale and GPFS: The GPFS InfiniBand Remote Direct Memory Access (RDMA) code uses RDMA for NSC client file I/O requests.
  - RDMA transfers data directly between the NSD client memory and the NSD server memory instead of sending and receiving the data over the TCP socket.
  - Using RDMA improves performance, enhances bandwidth and decreases CPU utilization.

Message Oriented Middleware (MOM)

1. Allows for communication between applications situated on heterogeneous operating systems and networks.
2. Allows developers to by-pass the costly process of building explicit connections between varied systems and networks.
3. Advanced Message Queue Protocol (AMQP) has emerged as an open standard for MOM communication and utilizes OFED in a product called MRG from RedHat.
4. Other messaging systems utilizing OFED are available from IBM, Tibco, Microsoft and other suppliers.
    a. Besides above kernel level ULPs, OFED also includes user level components such as the following.
    b. These components are transport neutral implementations (i.e., agnostic to the use of InfiniBand, iWARP or RoCE) that provide RDMA capabilities in user space.

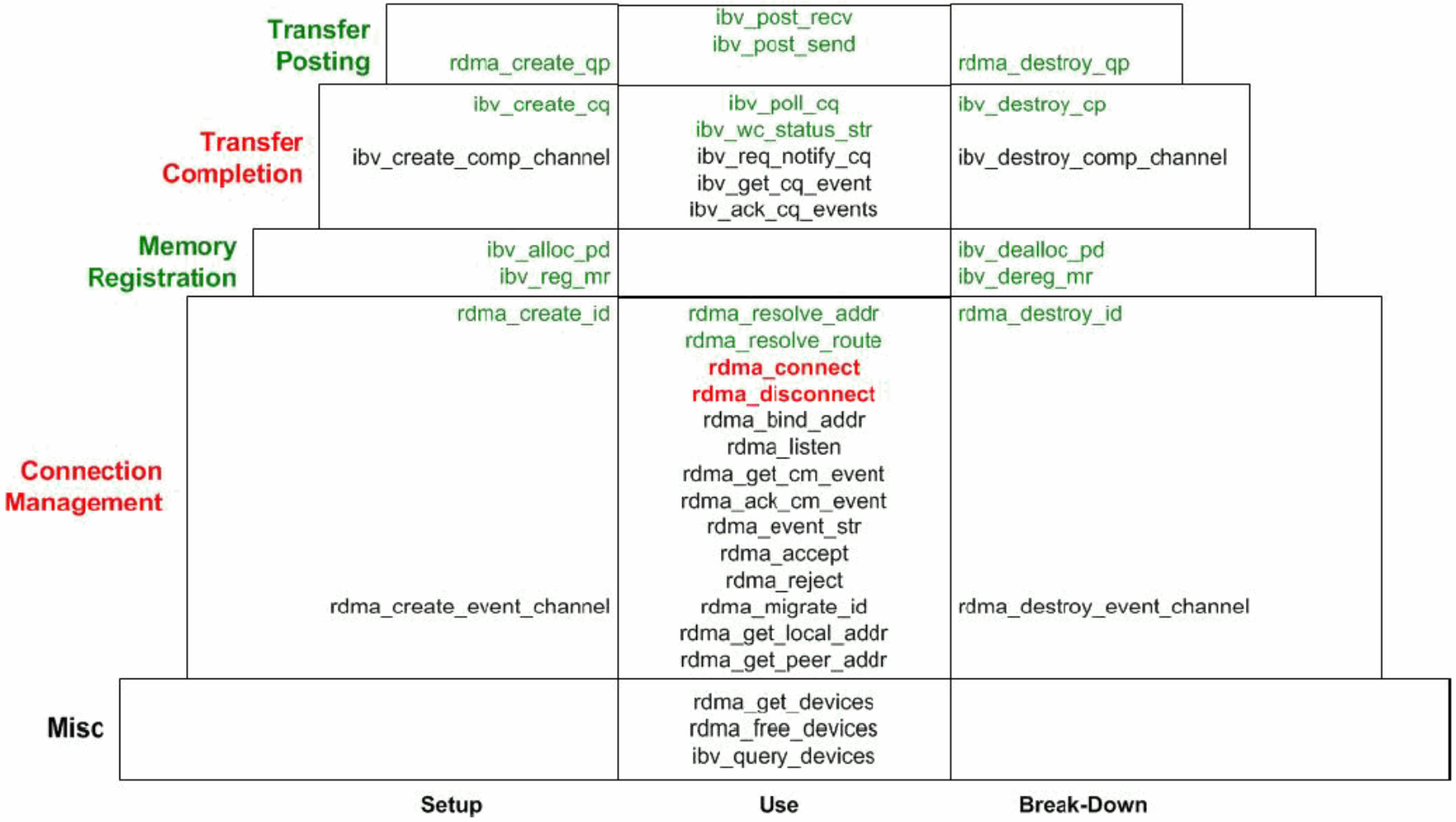
User Direct Access Programming Library

- UDAPL is a specification defined by the DAT (Direct Access Transport) Collaborative (www.datcollaborative.org).
  - It defines a single set of user APIs for all RDMA capable transports.
- UDAPL mission
  - Define a Transport-independent and Platform-standard set of APIs that exploits RDMA capabilities in various RDMA capable interconnects.
- UDAPL is included with OFED and is tested with OFA supported RDMA transports and interconnects, namely InfiniBand, iWARP and RoCE.

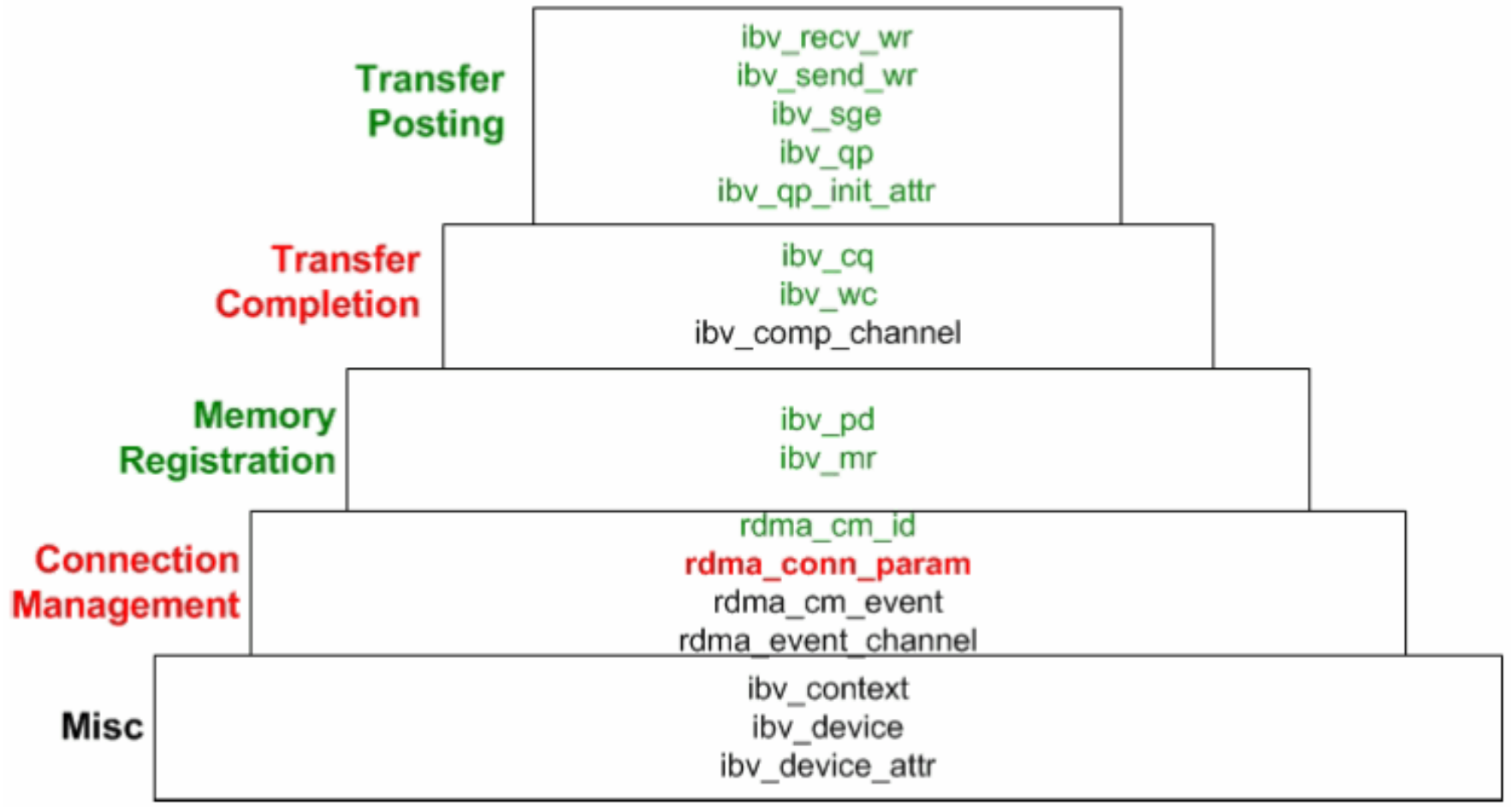
Message Passing Interface

- MPI is a language-independent communications protocol used to program parallel computers.
  - Both point-to-point and collective communication are supported.
  - MPI is a message-passing application programmer interface, together with protocol and semantic specifications for how its features must behave in any implementation.
  - MPI's goals are high performance, scalability, and portability.
  - MPI remains the dominant model used in high-performance computing today.
  - Various implementations of MPI are available in the industry.
  - OFED includes the Ohio State University implementation of MVAPICH/MVAPICH2 (http://mvapich.cse.ohio-state.edu/) and the Open MPI implementation (available from www.Open-MPI.org).

OFED verbs



Data structure verbs



Client setup phase

- rdma_create_id()
  - create struct rdma_cm_id – identifier
- rdma_resolve_addr()
  - bind struct rdma_cm_id to local device
- rdma_resolve_route()
  - resolve route to remote server
- ibv_alloc_pd()
  - create struct ibv_pd – protection domain
- ibv_create_cq()
  - create struct ibv_cq – completion queue
- rdma_create_qp()
  - create struct ibv_qp – queue pair
- ibv_reg_mr()
  - create struct ibv_mr – memory region
- rdma_connect()
  - create connection to remote server

Client break-down phase

- rdma_disconnect()
  - destroy connection to remote server
- ibv_dereg_mr()
  - destroy struct ibv_mr – memory region
- rdma_destroy_qp()
  - destroy struct ibv_qp – queue pair
- ibv_destroy_cp()
  - destroy struct ibv_cq – completion queue
- ibv_dealloc_pd()
  - deallocate struct ibv_pd – protection domain
- rdma_destroy_id()
  - destroy struct rdma_cm_id – identifier

Send Work Request (SWR) example

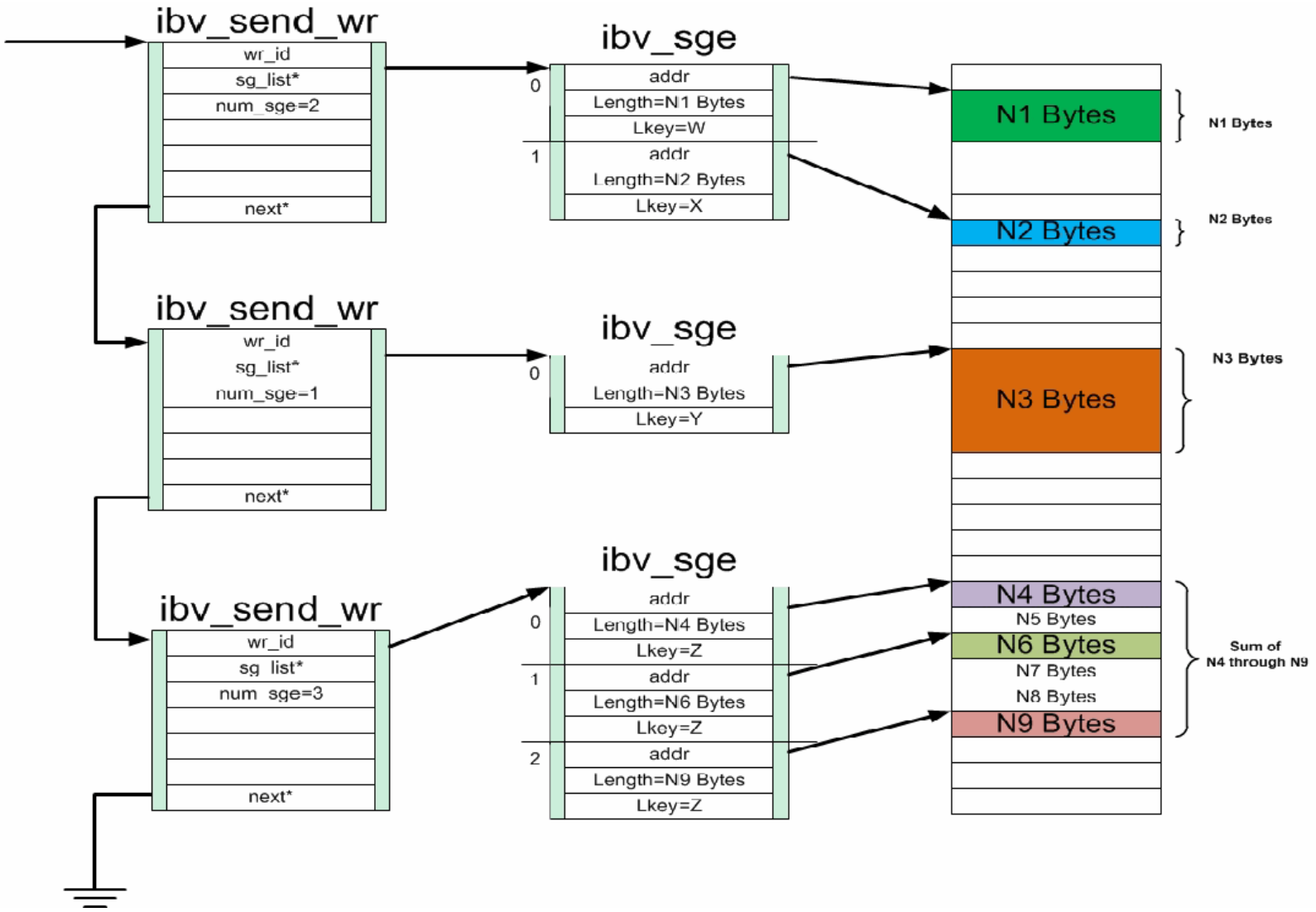
- Purpose: tell network adaptor what data to send
- Data structure: struct ibv_send_wr
- Fields visible to programmer:
  - See table
- Programmer must fill in these fields before calling ibv_post_send()

Flags	Descriptions
next	pointer to next SWR in linked list
wr_id	user-defined identification of this SWR
sg_list	array of scatter-gather elements (SGE)
opcode	IBV_WR_SEND
num_sge	number of elements in sg_list array
send_flags	IBV_SEND_ SIGNALED

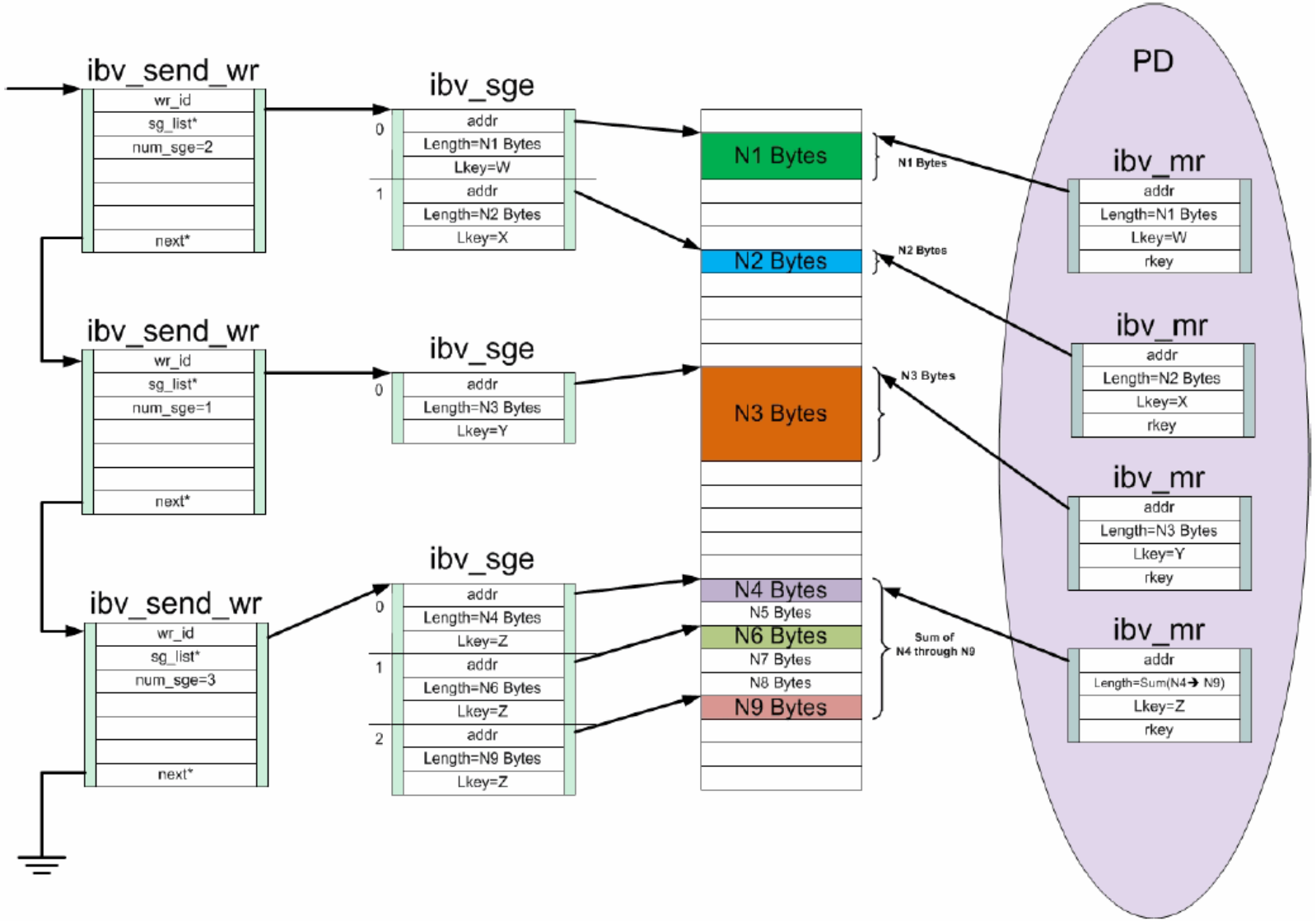
Posting to send data

- Verb: ibv_post_send()
- Parameters:
  - Queue Pair - QP
  - Pointer to linked list of Send Work Requests – SWR
  - Pointer to bad SWR in list in case of error
- Return value:
  - == 0 all SWRs successfully added to send queue (SQ)
  - != 0 error code

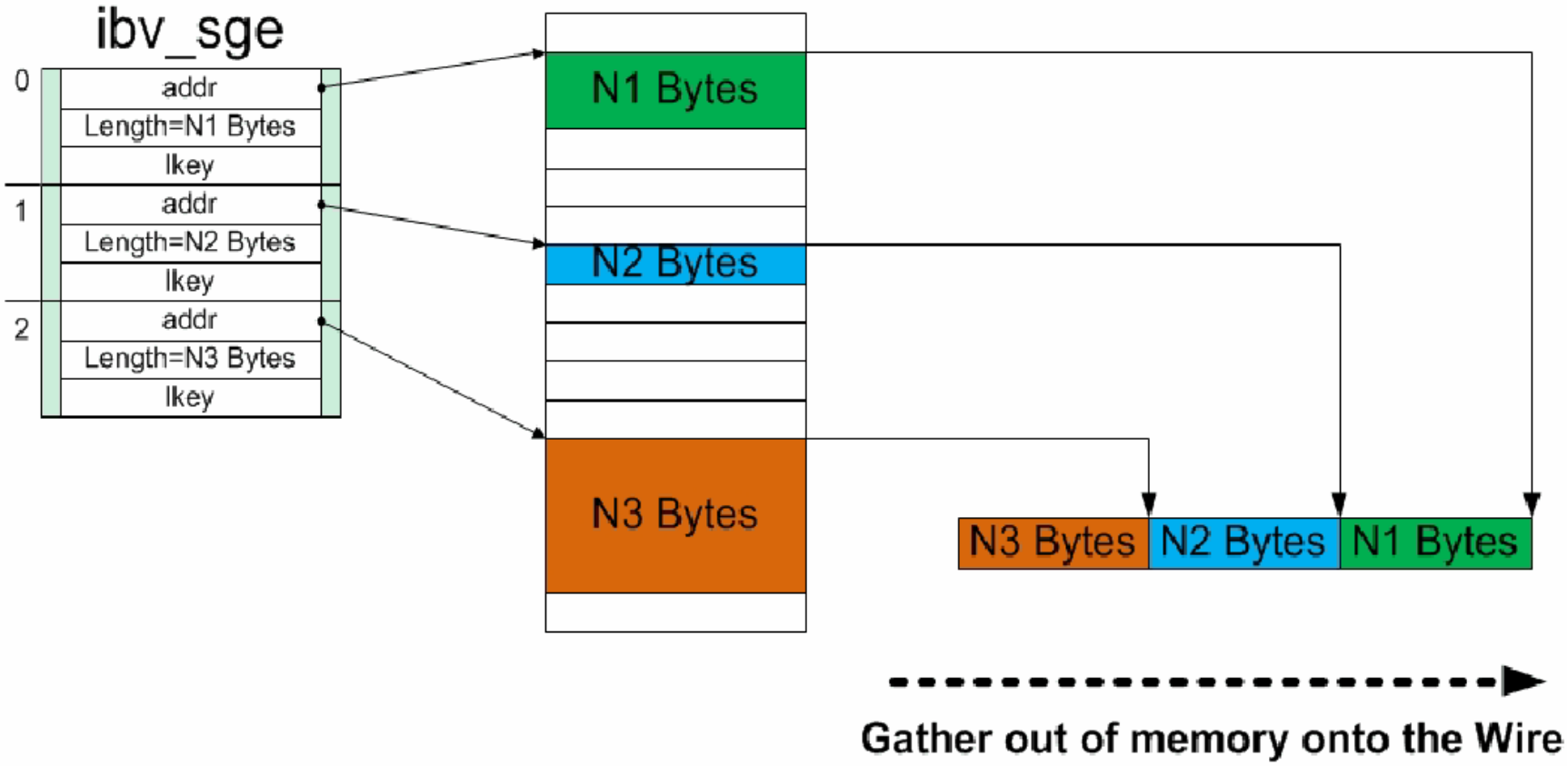
Scatter-gather



Protection domains



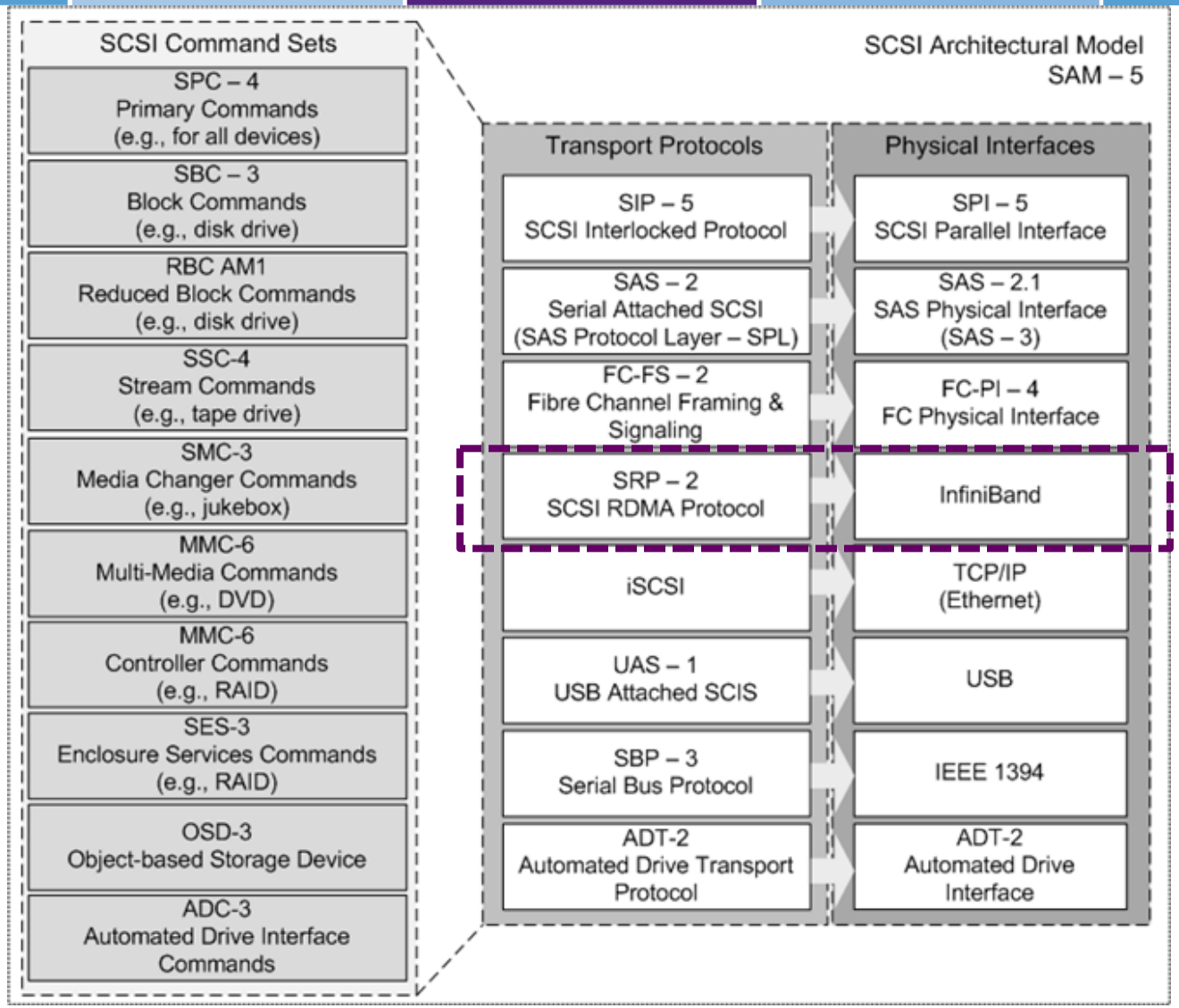
Gather up data during ibv_post_send()



SCSI Read Operation

IB transactions

SCSI RDMA Protocol (SRP-2)

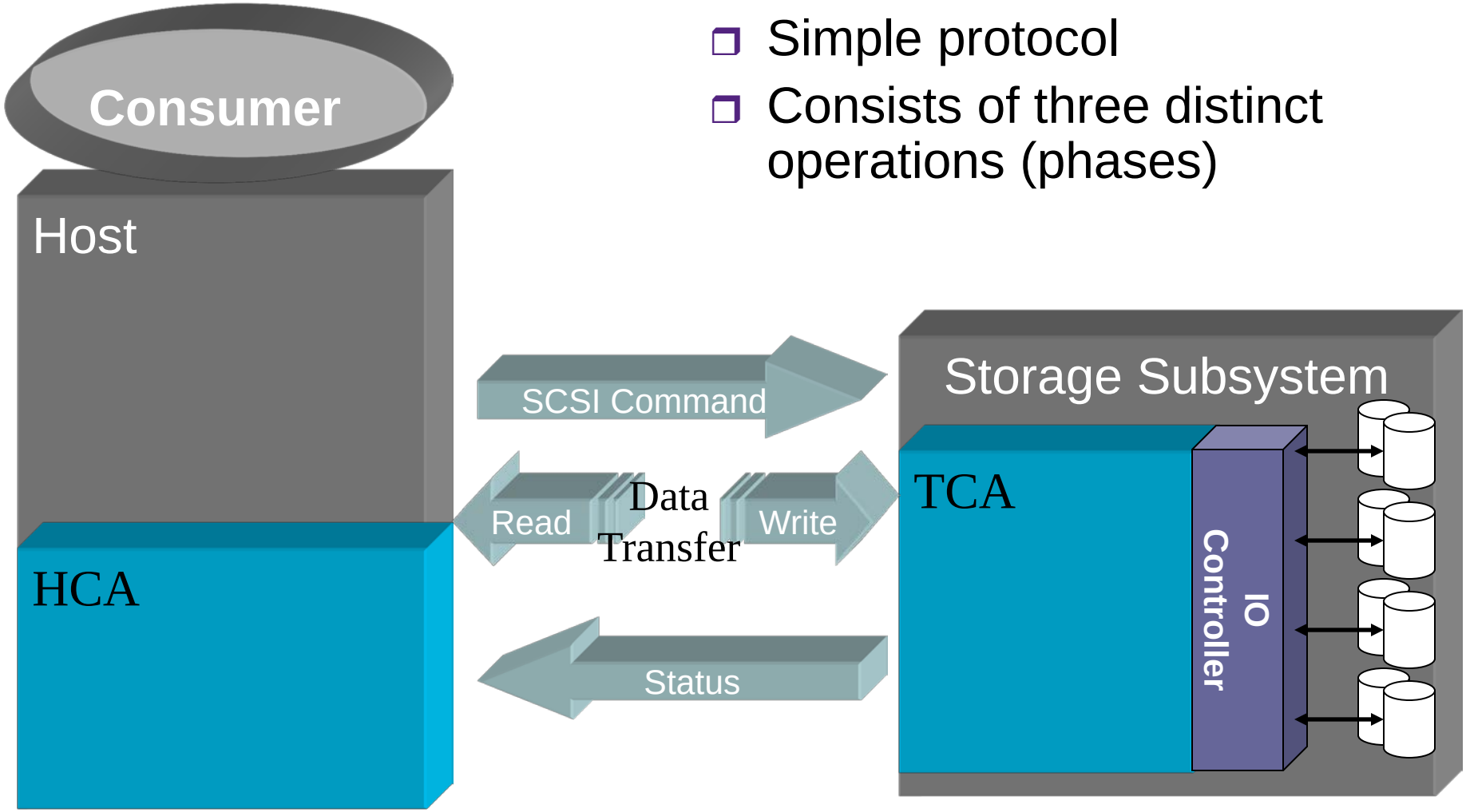


Typical I/O Transaction

- A typical I/O transaction might use a combination of channel and memory semantics.
  - a host process might initiate an I/O operation by using channel semantics to SEND a disk write command to an I/O device.
  - The I/O device examines the command and uses memory semantics to read the data buffer directly from the memory space of the processor node.
    - RDMA READ or WRITE
  - After the operation is completed, the I/O unit then uses channel semantics to push (SEND) an I/O completion message back to the processor node.

SCSI Protocol Basics

- Simple protocol
- Consists of three distinct operations (phases)

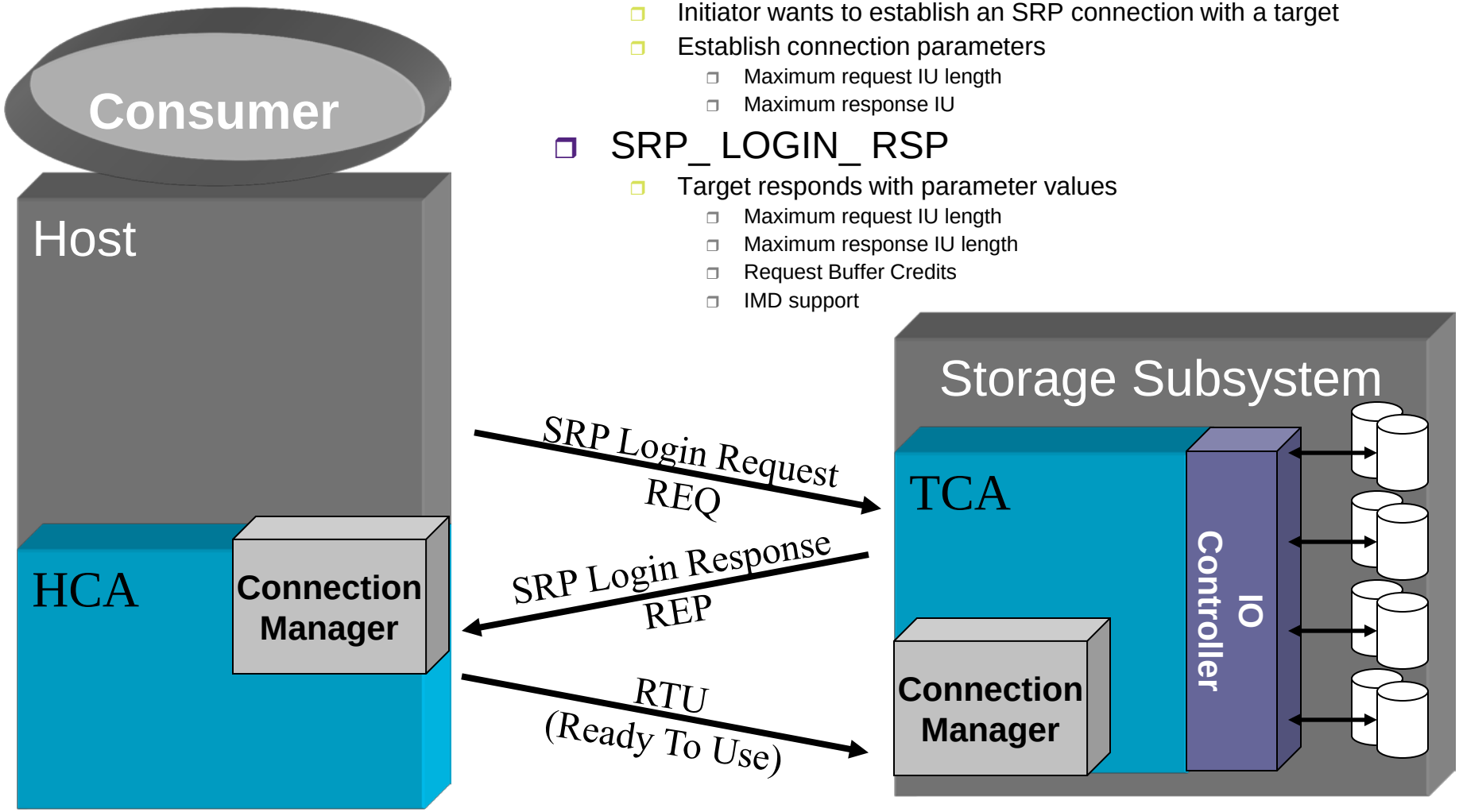


Connections

- IBA supports both:
  - connection oriented service
  - datagram service
  - For connected service, each QP is associated with exactly one remote consumer
- QP context is configured with the identity of the remote consumer's queue pair
- The remote consumer is identified by a port and a QP number.
  - The port is identified by a local ID (LID) and optionally a Global ID (GID)
  - During the communication establishment process, this and other information is exchanged between the two nodes
- For datagram service, a QP is not tied to a single remote consumer, but rather information in the WQE identifies the destination.
- A communication setup process similar to the connection setup process needs to occur with each destination to exchange any information.

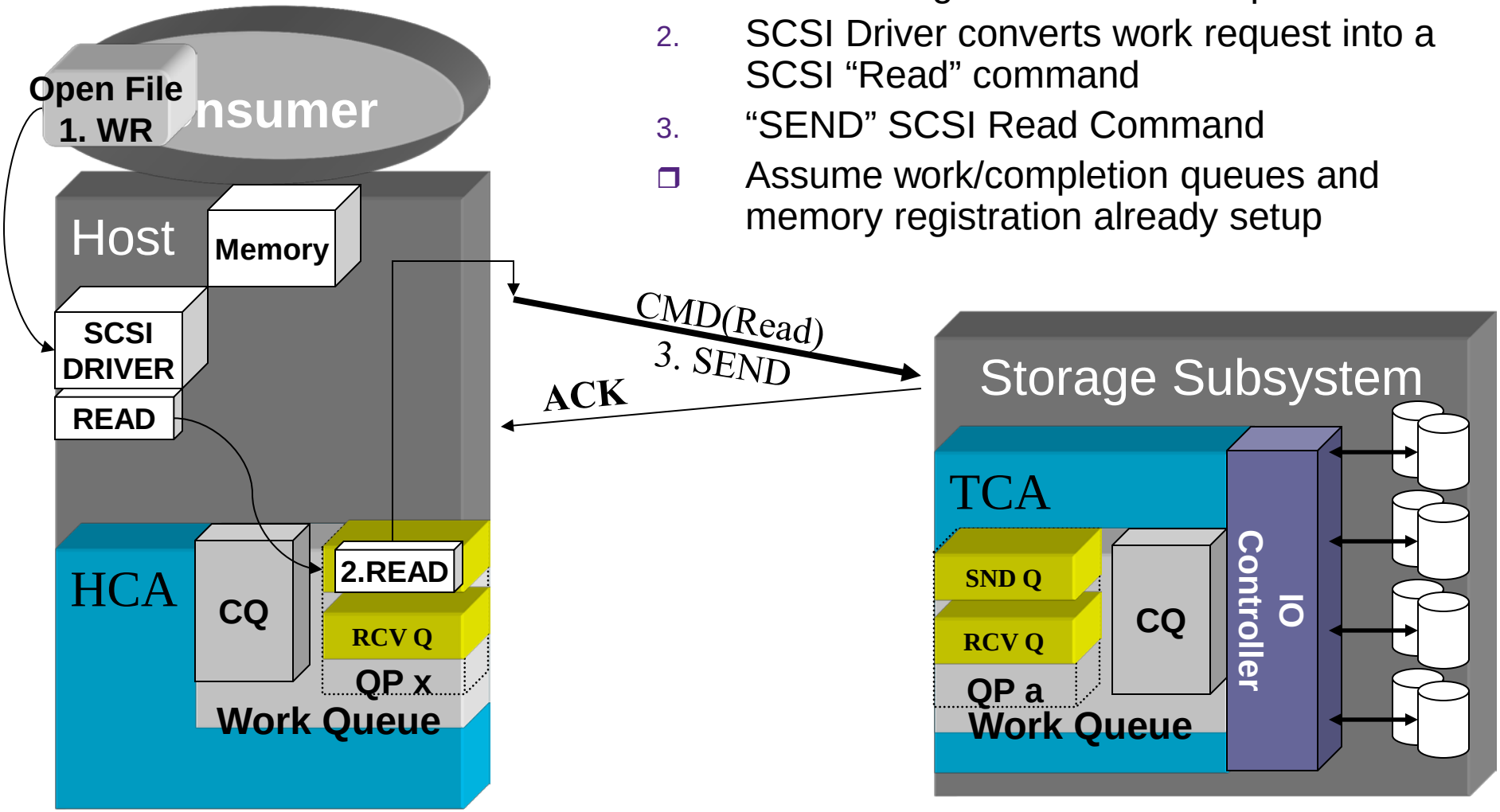
SRP Login – Connection

- SRP_ LOGIN_ REQuest
  - Initiator wants to establish an SRP connection with a target
  - Establish connection parameters
    - Maximum request IU length
    - Maximum response IU
- SRP_ LOGIN_ RSP
  - Target responds with parameter values
    - Maximum request IU length
    - Maximum response IU length
    - Request Buffer Credits
    - IMD support



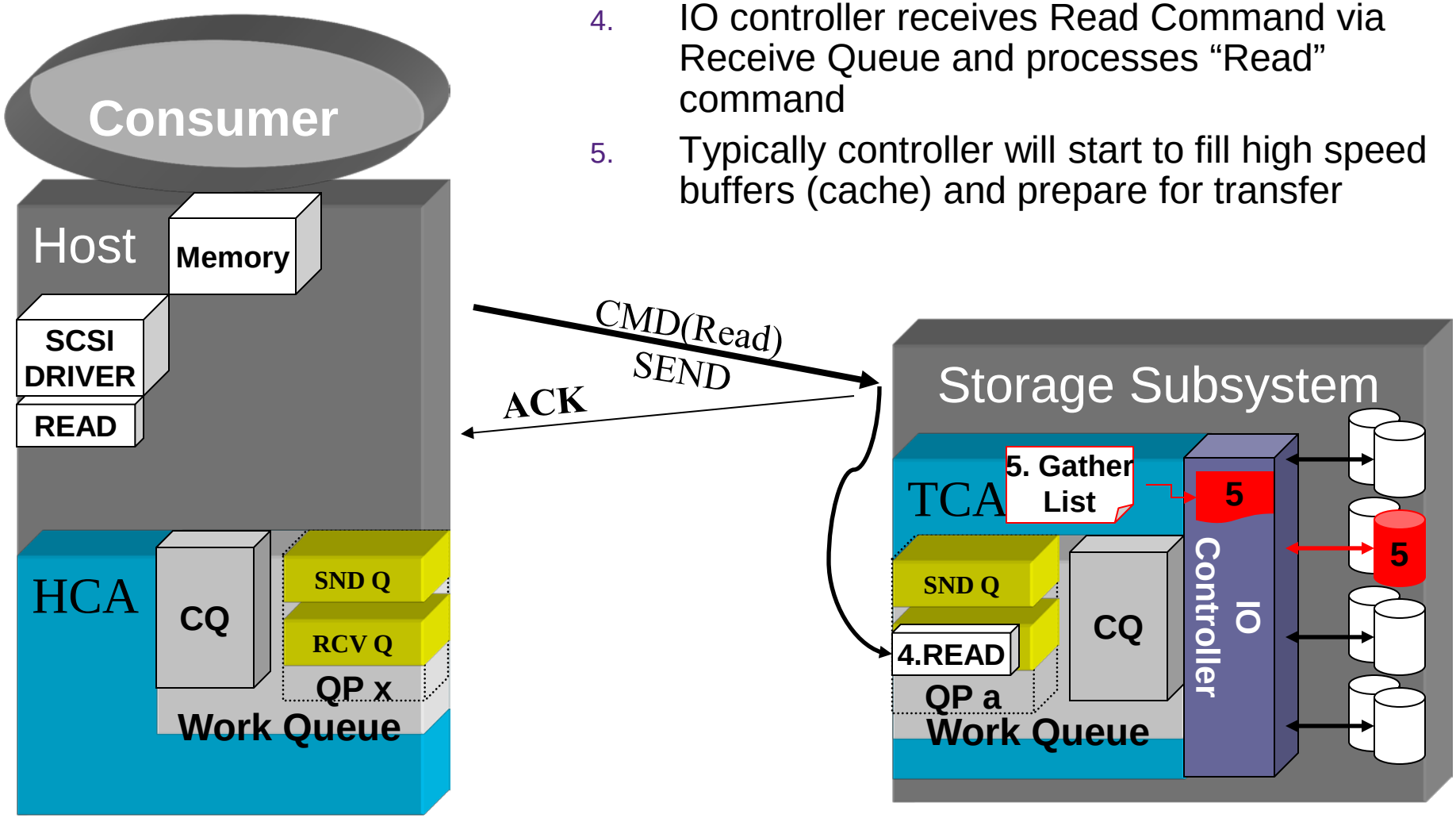
SRP Command Transfer

1. Consumer generates work request
2. SCSI Driver converts work request into a SCSI "Read" command
3. "SEND" SCSI Read Command
   - Assume work/completion queues and memory registration already setup



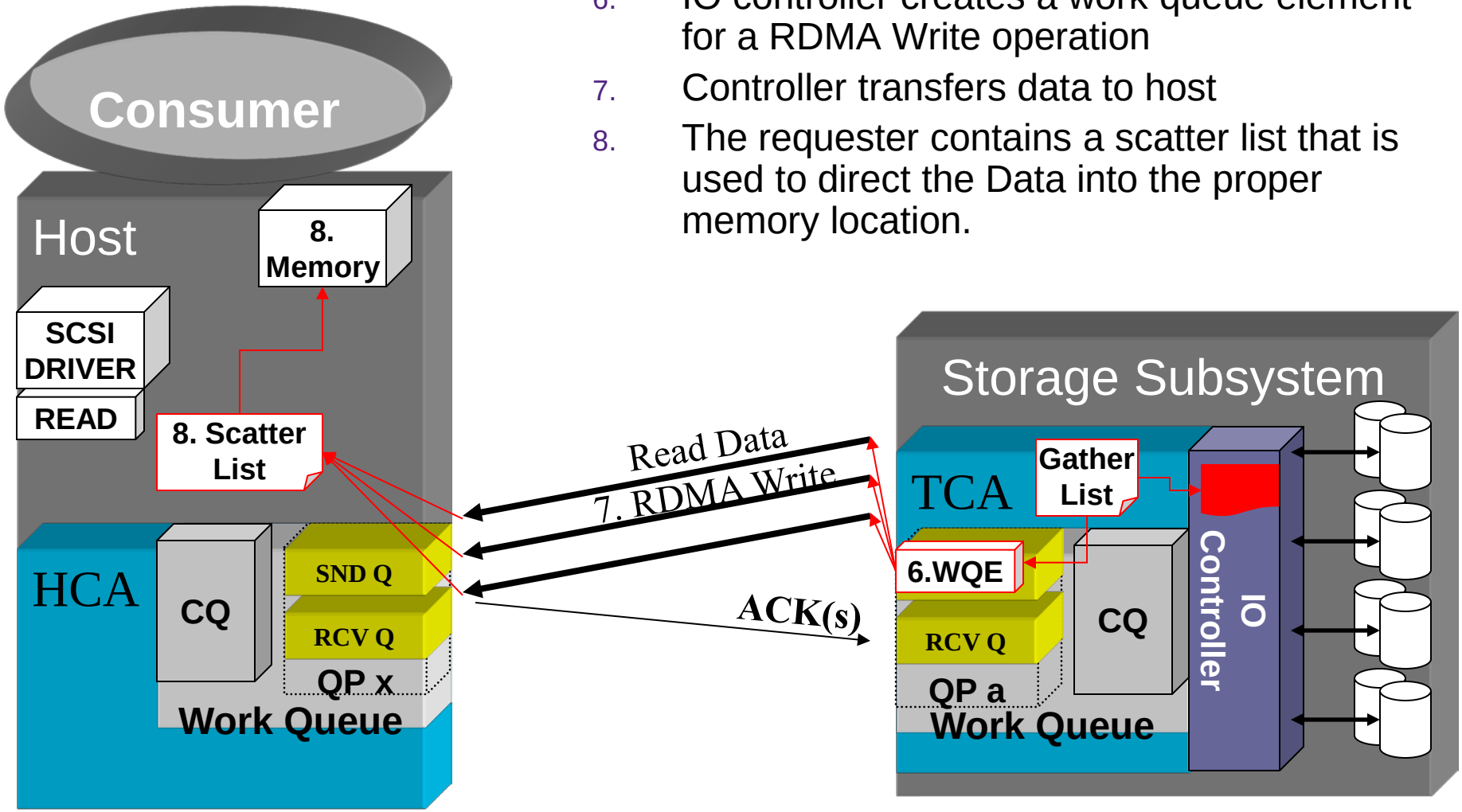
IO Controller Processes Operation

4. IO controller receives Read Command via Receive Queue and processes “Read” command
5. Typically controller will start to fill high speed buffers (cache) and prepare for transfer



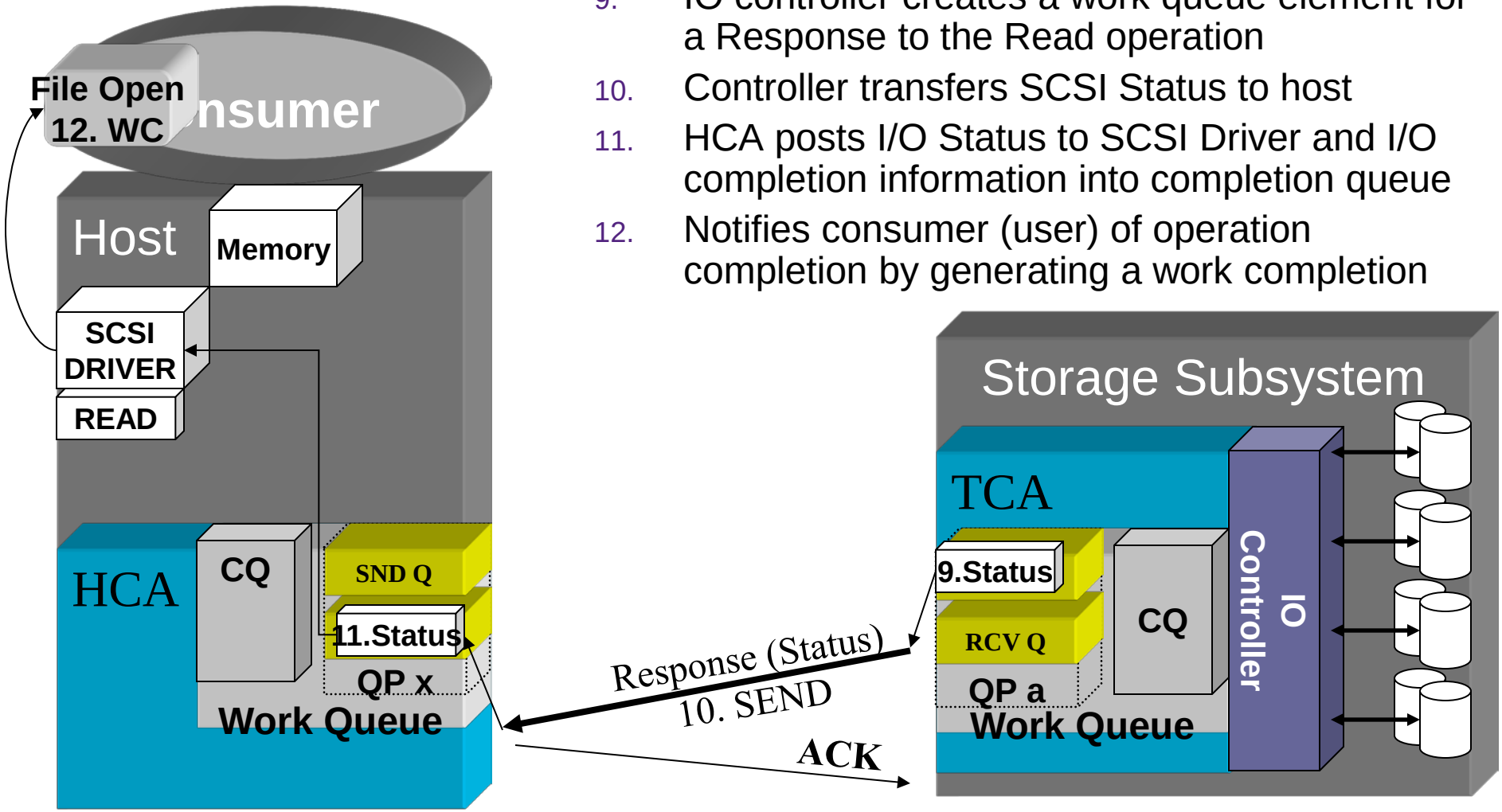
Transfer Data

6. IO controller creates a work queue element for a RDMA Write operation
7. Controller transfers data to host
8. The requester contains a scatter list that is used to direct the Data into the proper memory location.



SRP Status Transfer

9. IO controller creates a work queue element for a Response to the Read operation
10. Controller transfers SCSI Status to host
11. HCA posts I/O Status to SCSI Driver and I/O completion information into completion queue
12. Notifies consumer (user) of operation completion by generating a work completion



Memory management

Memory Registration

- The Resource Manager is responsible for Memory Registration.
  - Memory Registration allows the application to specify virtual addresses and protects the application.
  - Each time a QP is created an associated memory registration must take place.
  - This registration is used to reserve a virtually contiguous block of memory.
- The application controls:
  - Which QPs can access the memory
  - How the QP accesses the memory
  - The conditions of the access

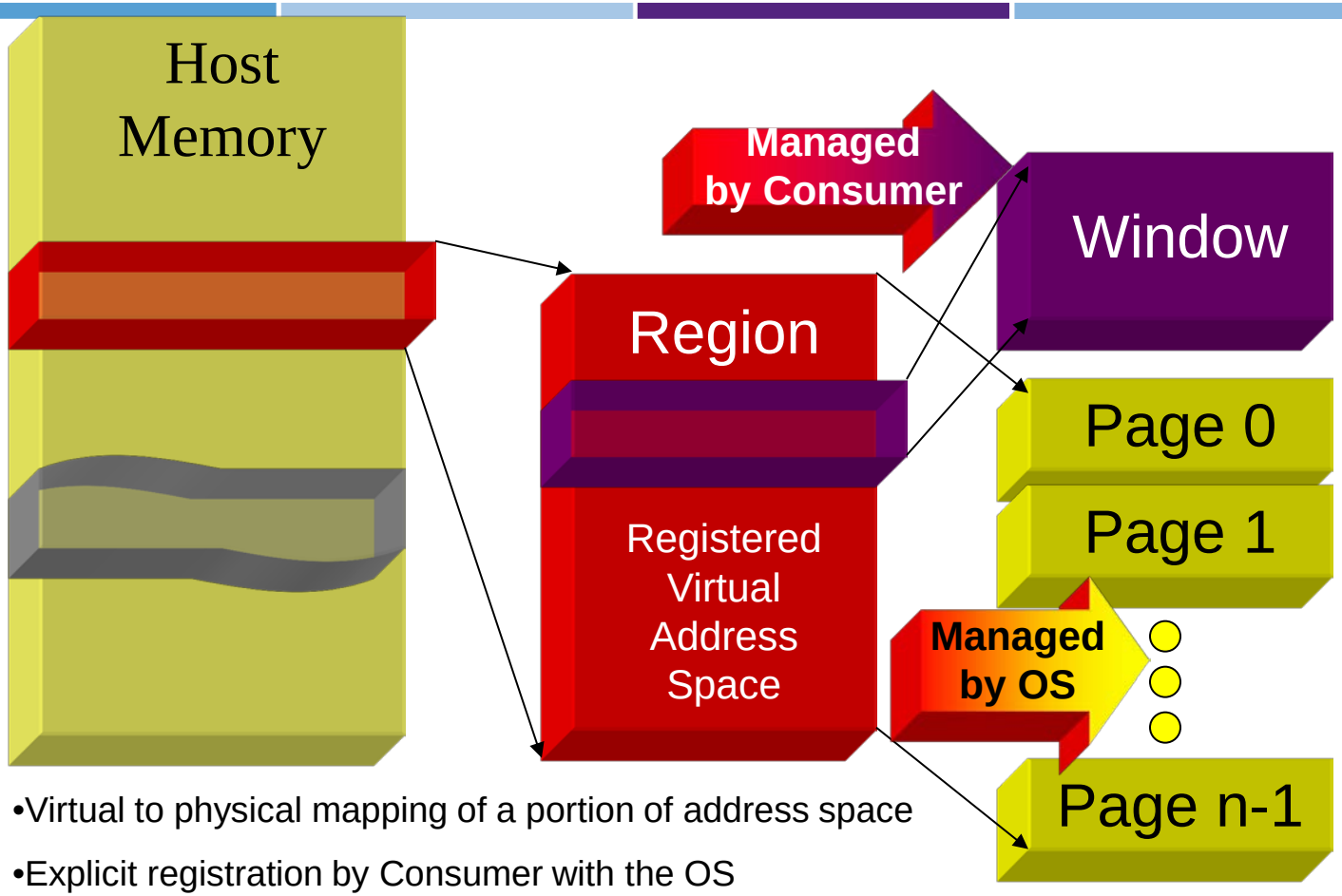
Two levels of memory registration

1. Windows
   - Objects that an application may allocate to dynamically control remote access to various portions of registered memory regions.
   - Provides byte-level granularity for remote access where Regions do no typically provide byte level protection.
2. Regions
   - Limits the number and size of registered memory
   - Identifies a virtually contiguous memory range
   - Specifies access rights
   - Produces an L_Key and an R_Key that application uses when it accesses that memory Region. The R_Key allows remote applications to access the local memory.

Memory Registration Characteristics

- The HCA is responsible for maintaining memory tables that map the memory region to actual physical pages.
- Memory registration may also involve pinning memory (or binding) where physical memory is not paged out.
  - The memory Bind is an operation the application performs via a WR that is posted on the QP.
- Once a memory region is registered, a memory window can be bound to it.
  - Binding can be quickly changed to another portion of the same region or to a portion of another memory region.
  - Memory binding and windows allows an application to quickly and easily allocate or reallocate a portion of a memory region.

Memory Model



-Windows are architected to enable flexible and efficient dynamic RDMA access control to underlying Memory Regions
-Consumer "binds" a pre-allocated Window to a specified portion of an existing Region by posting a request to a Send Queue

- Virtual to physical mapping of a portion of address space
- Explicit registration by Consumer with the OS
- QP access to Regions managed through Protection Domains
- Consumers use virtual addresses, HCA performs VtoP mapping
- Similar to VI architecture

Channel & Memory Semantics

- IBA communications provide the user with both channel semantics and memory semantics since both are useful for I/O and IPC.
- Channel semantics
  - sometimes called Send/Receive,
  - refers to the communication style used in a classic I/O channel – one party pushes the data and the destination party determines the final destination of the data.
  - The message transmitted on the wire only names the destination's QP, the message does not describe where in the destination consumer's memory space the message content will be written.
- With memory semantics
  - The initiating party directly reads or writes the virtual address space of a remote node.
  - The remote party needs only communicate the location of the buffer; it is not involved with the actual transfer of the data.

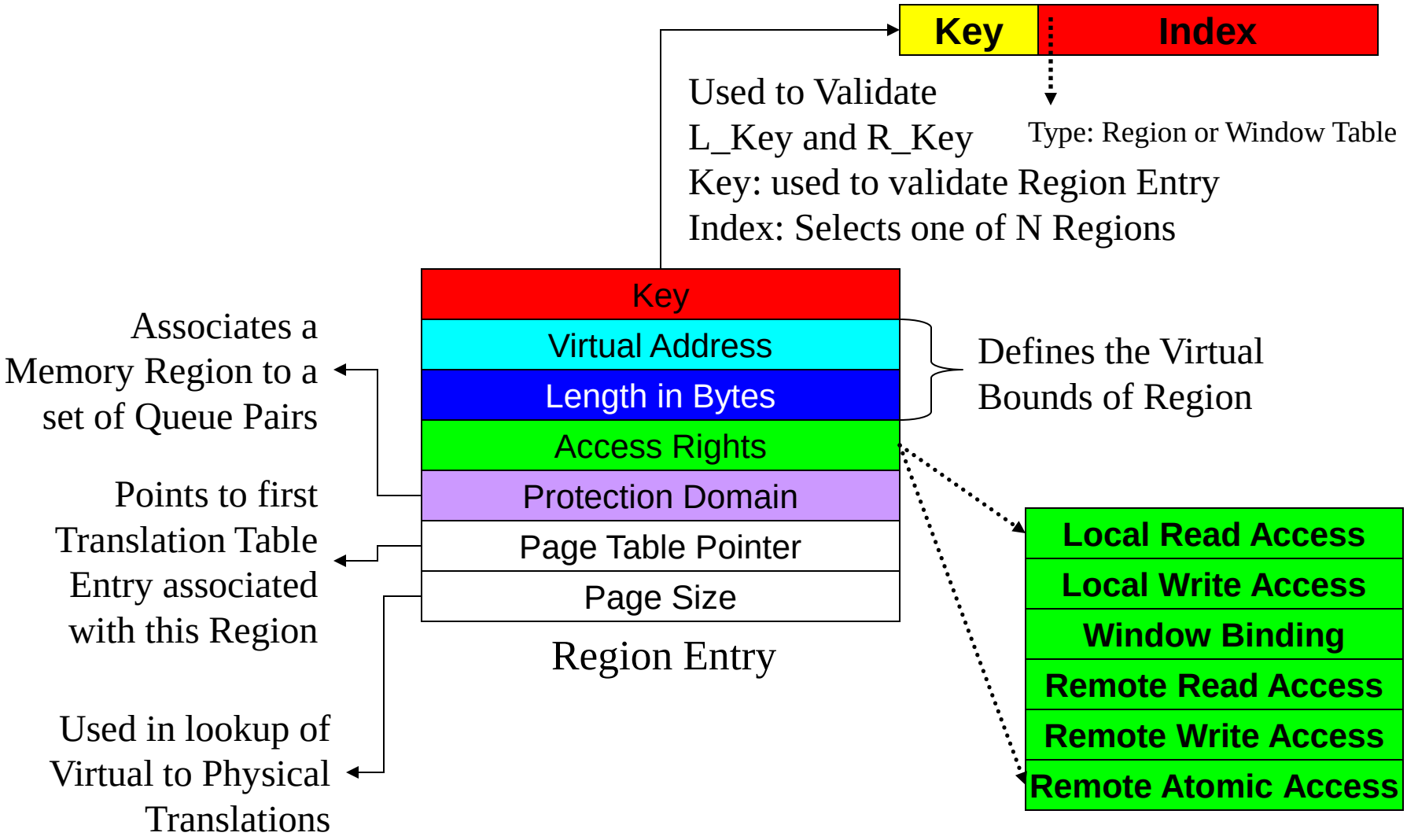
Virtual Memory Addressing

- IBA is optimized for virtual addressing.
  - an IBA consumer uses virtual addresses in work requests
  - the channel adapter is able to convert the virtual address to physical address as necessary
- Each consumer
  - registers regions of virtual memory with the channel adapter
  - the channel adapter returns 2 memory handles to the consumer
    - L_Key
    - R_Key
  - The consumer uses the L_key in each work request that requires a memory access to that region.

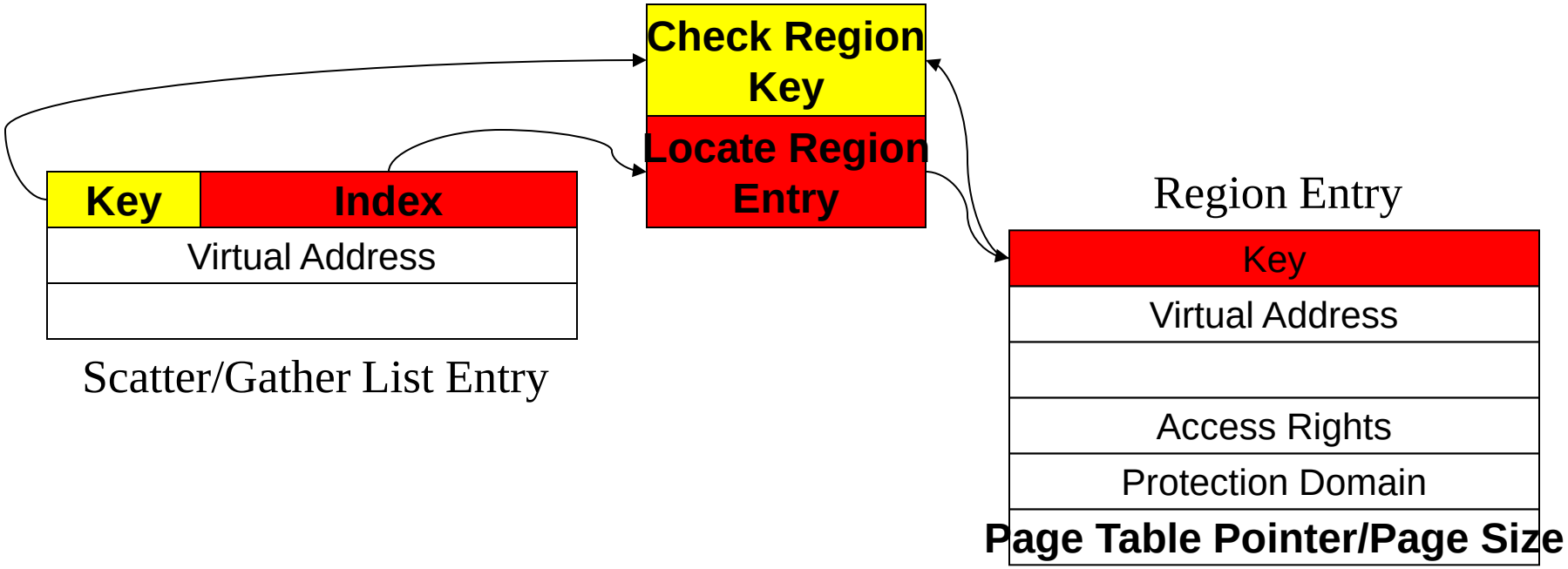
Memory Regions

- System memory is registered to allow access to local and remote channel adapters
  - Registration returns R_key and L_key tokens
  - Permission specified at registration time:
    - Read-only verses read-write
    - Local only verses local/remote
- R_key token is provided to remote nodes, granting them the ability to perform RDMA against the memory in the region
- L_key token is used for local access, i.e. scatter/gather list
- Verbs: Register, Register Physical, Query, Deregister, Reregister, Reregister Physical, Register Shared

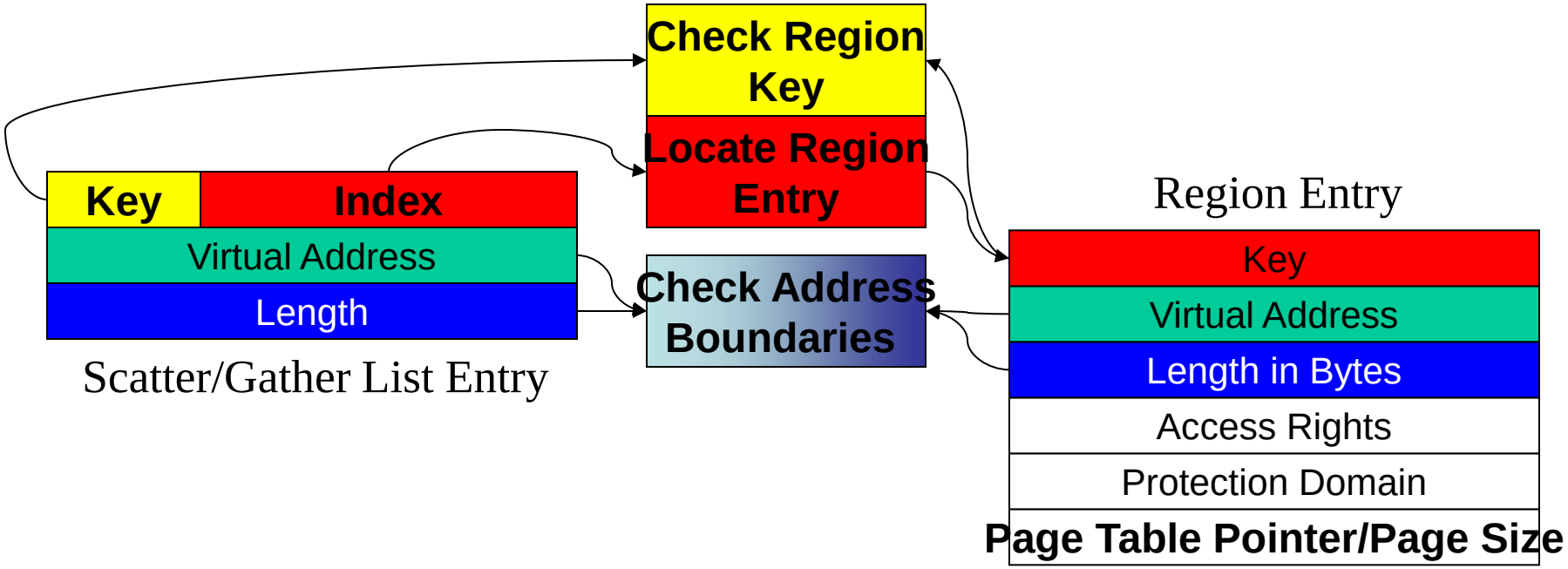
Memory Regions



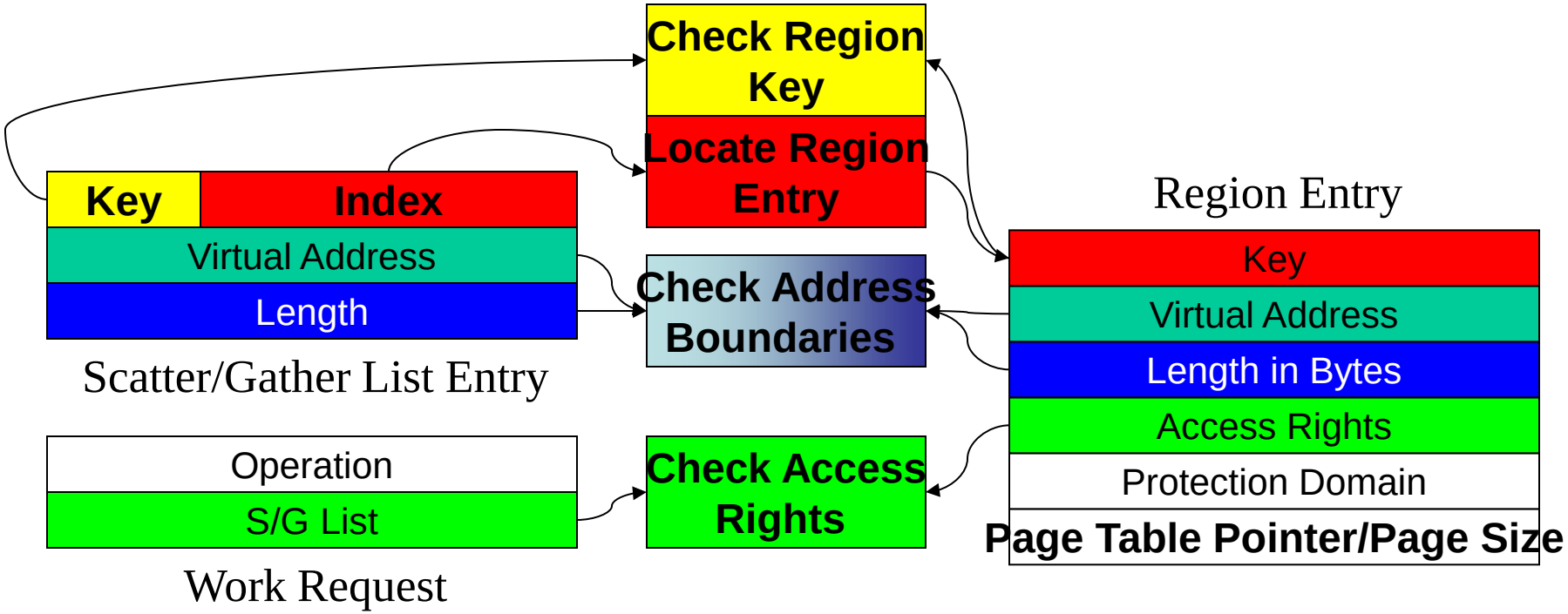
Memory Region Access Example A



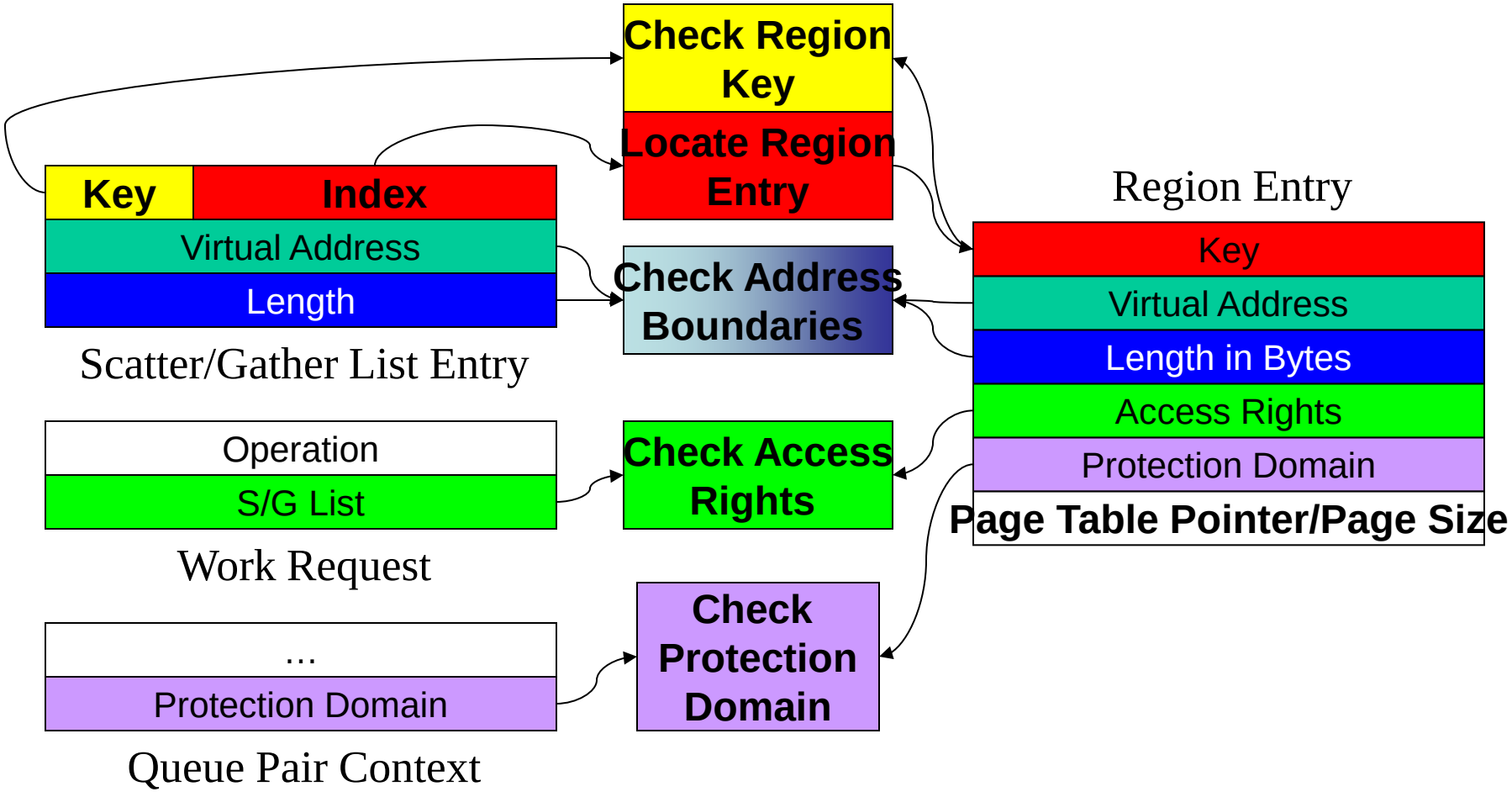
Memory Region Access Example B



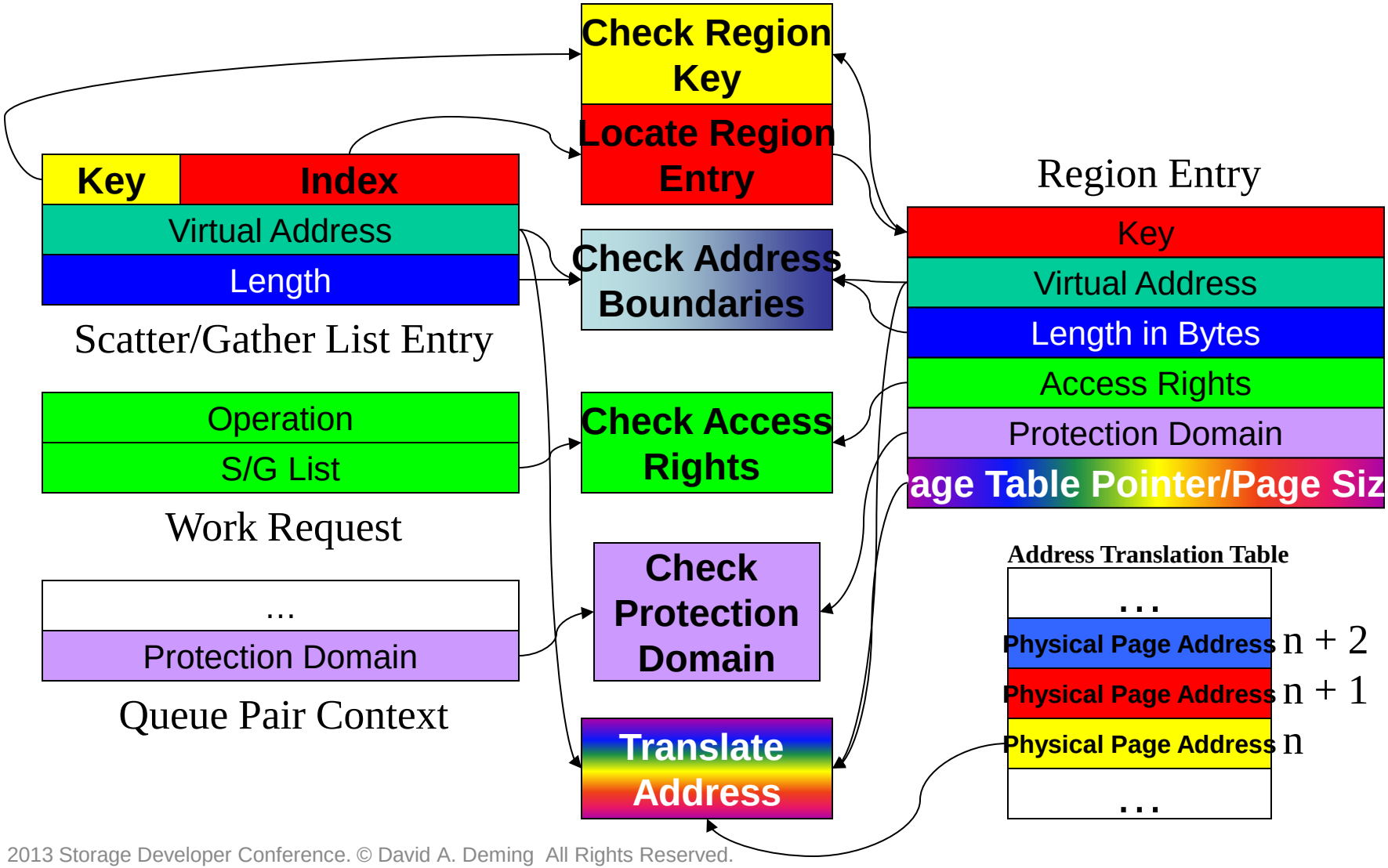
Memory Region Access Example C



Memory Region Access Example D



Memory Region Access Example E



InfiniBand Software Architecture
End