Portals Programming on the XT3

Ron Brightwell Rolf Riesen
Sandia National Laboratories
Scalable Computing Systems Department

Barney Maccabe
University of New Mexico
Scalable System Lab

Cray Users' Group
Annual Technical Conference
May 8, 2006

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

Sandia National Laboratories

Outline

- Overview
- Portals objects
- Portals API
- Portals semantics
- XT3 implementation
- Upcoming work

Sandia National Laboratories

Portals Timeline

- Portals 0.0 - 1991
  - SUNMOS (Sandia/UNM OS)
  - nCUBE, Intel Paragon
  - Direct access to network FIFOs
  - Message co-processor
- Portals 1.0 - 1993
  - Message passing data structures in user-space
  - Kernel-managed and user-managed memory descriptors
  - Published but never implemented
- Portals 2.0 - 1994
  - Puma/Cougar lightweight operating system
  - Message selection capability (match lists)
  - Four types of memory descriptors (three implemented)
- Portals 3.0 - 1998
  - Cplant Linux clusters
  - Functional API
  - Target intelligent/programmable network interfaces (Myrinet)

Sandia National Laboratories

Portals 3.3 Features

- Best effort, in-order delivery
- One-sided operations
  - Put, Get, Atomic swap
- Supports zero-copy
- Supports OS-bypass
- Supports application offload
  - No polling or threads to move data
  - No host CPU overhead
- Well-defined transport failure semantics
- Unexpected operations are discarded
- Receive-side access control
- Runtime-system independent

Sandia National Laboratories

What Makes Portals Different?

- Connectionless RDMA with matching
- Provides elementary building blocks for supporting higher-level protocols well
  - MPI, RPC, Lustre, etc.
- Allows structures to be placed in user-space, kernel-space, or NIC-space
- Receiver-managed offset allows for efficient and scalable buffering of MPI “unexpected” messages
- Supports multiple protocols within a process
  - Needed for compute nodes where everything is a message

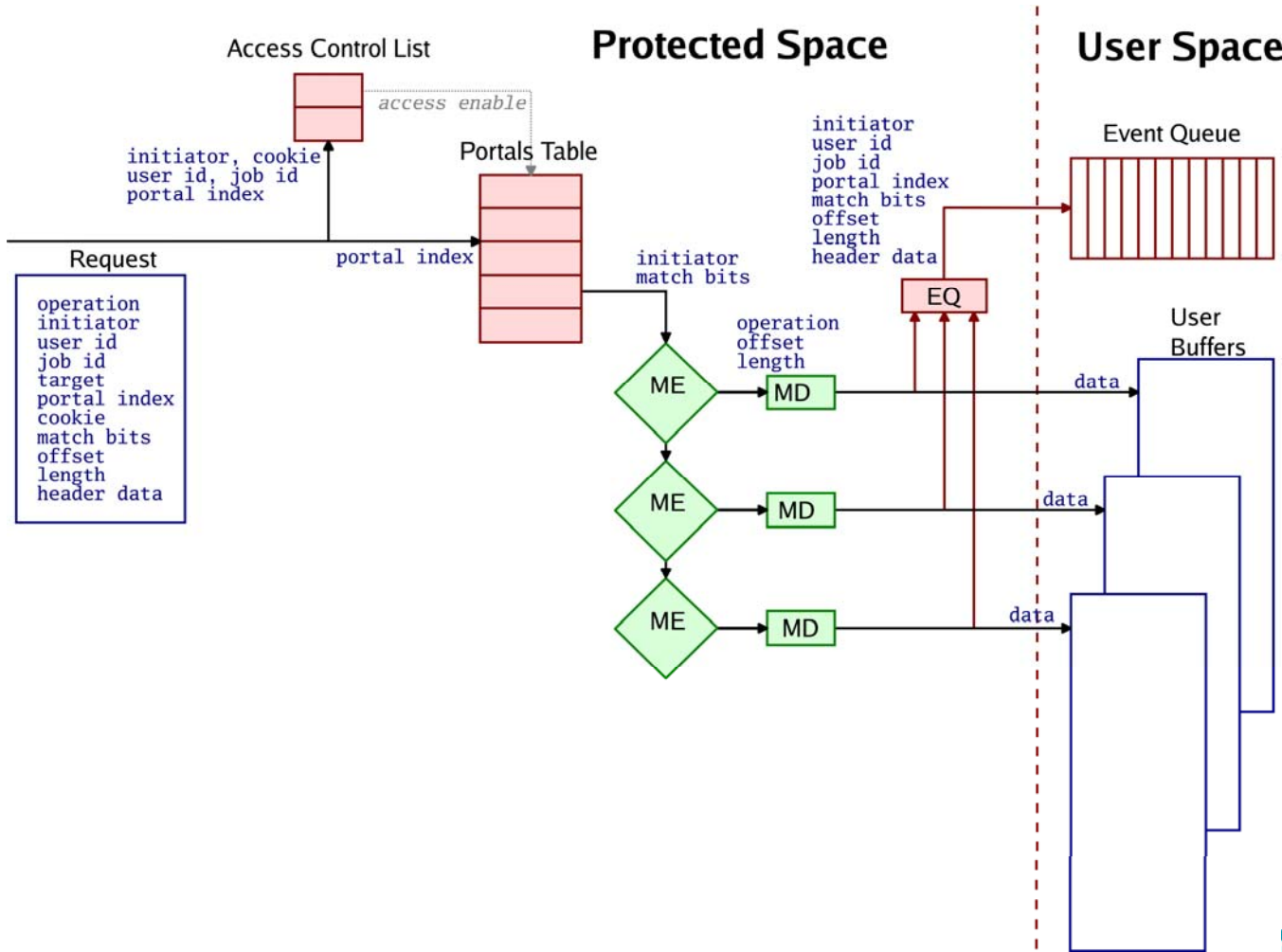
Sandia National Laboratories

Basic Objects: Memory Descriptors and Addresses

- Memory descriptors
  - Logically contiguous region of memory
  - Explicit creation
    - Pin pages (for Linux)
    - Pass address map to network interface
  - Scatter/gather semantics
- Remote addresses include:
  - Destination id (node id, process id)
  - Portal table index (protocol switch)
  - Message selection bits (MPI style matching)
  - Offset – sender-managed memory descriptors
  - Access control cookie

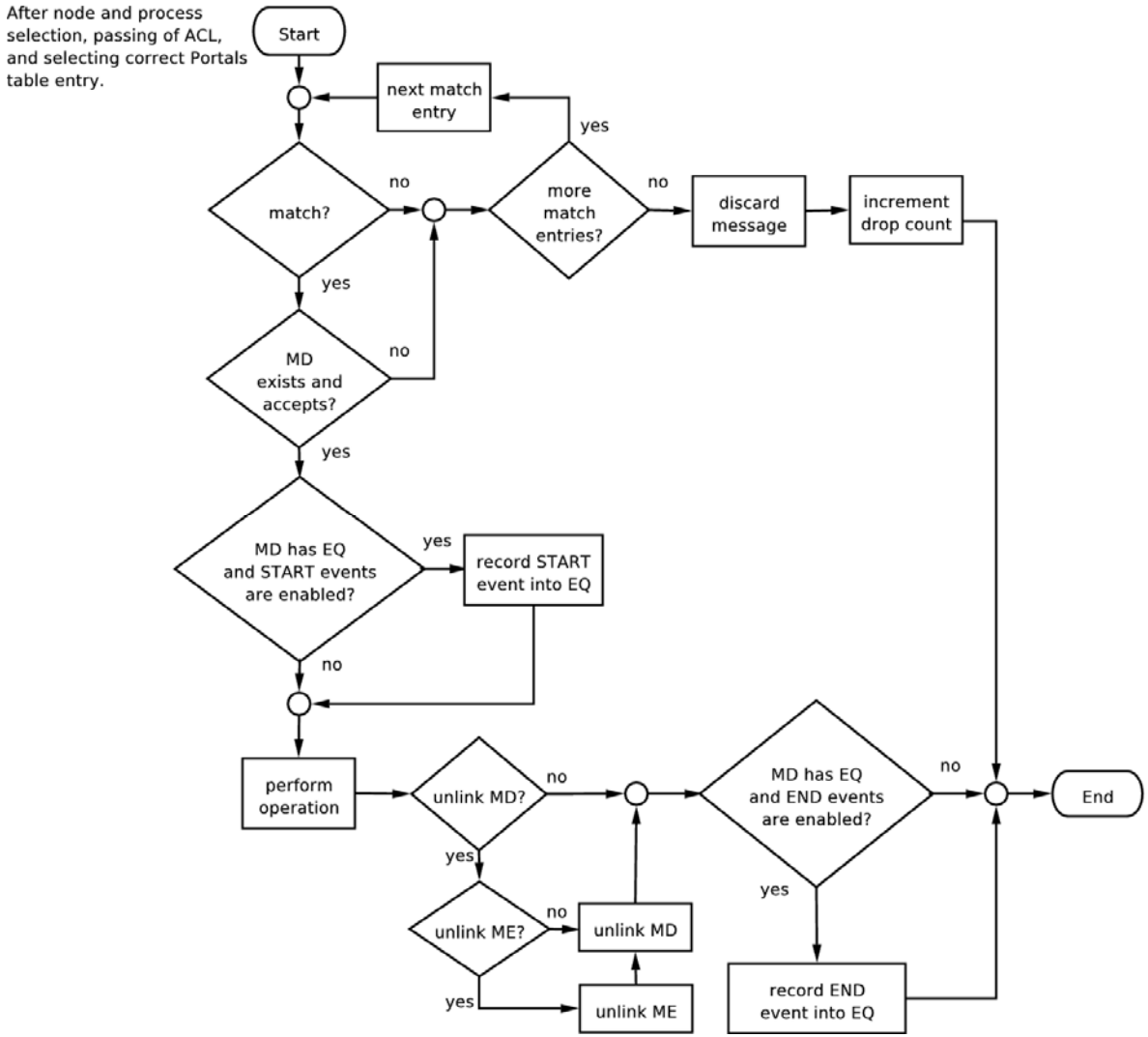
Sandia National Laboratories

Addressing



Sandia National Laboratories

Steps for Address Translation



Sandia National Laboratories

Library Initialization/Shutdown

- `int PtlInit( int *max_interfaces );`
- `void PtlFini( void );`

Sandia National Laboratories

Network Interfaces

- Supports the use of multiple network interfaces
- Each interface is independent
- Each interface provides
  - A portal table
    - With at least 8 entries
  - An access control table
  - Status registers
- Every portals object is associated with a specific interface

Sandia National Laboratories

Network Interface Creation & Destruction Functions

- int PtlNIInit(
    ptl_interface_t iface,
    ptl_pid_t pid,
    ptl_ni_limits_t *desired,
    ptl_ni_limits_t *actual,
    ptl_handle_ni_t *ni_handle
  );

- int PtlNIFini(
    ptl_handle_ni_t ni_handle
  );

Sandia National Laboratories

Network Interface Utility Functions

- int PtlNIStatus(
  ptl_handle_ni_t ni_handle,
  ptl_sr_index_t status_register,
  ptl_sr_value_t *status
  );

- int PtlNIDist(
  ptl_handle_ni_t ni_handle,
  ptl_process_id_t pid,
  unsigned long *distance
  );

Sandia National Laboratories

Identification

- User Id
  - Every process runs on behalf of a user
  - Mandates trusted header information
- Process Id
  - Uniquely identifies a process on a portals network
- Job Id
  - Allows for processes to be aggregated
  - Useful for parallel jobs

Sandia National Laboratories

Identification Functions

- ```
  int PtlGetUid (
       ptl_handle_ni_t      ni_handle,
       ptl_uid_t            *uid
   );
  ```

- ```
  int PtlGetId(
       ptl_handle_ni_t      ni_handle,
       ptl_process_id_t     *pid
   );
  ```

- ```
  int PtlGetJid(
       ptl_handle_ni_t      ni_handle,
       ptl_job_id_t         *jid
    );
  ```

Sandia National Laboratories
  ```

Match Entry (ME) and Match Lists

- Each match entry contains
  - Source node id
  - Source process id
  - 64 match bits
  - 64 ignore bits
- A match entry is attached to a portal table entry
- Match entries can be linked to create a match list
- Can be automatically unlinked and freed from list when used up

Sandia National Laboratories

Match Entry Creation Functions

```
• int PtlMEAttach(
        ptl_handle_ni_t     ni_handle,
        ptl_pt_index_t      pt_index,
        ptl_process_id_t    match_id,
        ptl_match_bits_t    match_bits,
        ptl_match_bits_t    ignore_bits,
        ptl_unlink_t        unlink_op,
        ptl_ins_pos_t       position,
        ptl_handle_me_t     *me_handle
   );

• int PtlMEAttachAny(
        ptl_handle_ni_t     ni_handle,
        ptl_pt_index_t      *pt_index,
        ptl_process_id_t    match_id,
        ptl_match_bits_t    match_bits,
        ptl_match_bits_t    ignore_bits,
        ptl_unlink_t        unlink_op,
        ptl_ins_pos_t       position,
        ptl_handle_me_t     *me_handle
   );
```

Sandia National Laboratories

Match Entry Insertion Function

```
int PtlMEInsert(
    ptl_handle_me_t   base,
    ptl_process_id_t  match_id,
    ptl_match_bits_t  match_bits,
    ptl_match_bits_t  ignore_bits,
    ptl_unlink_t      unlink_op,
    ptl_ins_pos_t     position,
    ptl_handle_me_t   *me_handle
);
```

Sandia National Laboratories

Match Entry Destruction Function

- `int PtlMEUnlink( ptl_handle_me_t me_handle );`

Sandia National Laboratories

Memory Descriptor (MD)

- Describes regions of memory for data movement operations
- Defines what operations can be performed on the memory
- Can be automatically freed when used up

Sandia National Laboratories

Memory Descriptor Structure

- start
  - Starting address of memory region
- length
  - Length in bytes of memory region
- threshold
  - Maximum number of operations
- max_size
  - Largest incoming request
- options
  - See next slide
- user_ptr
  - User-specific value returned in events
- eq_handle
  - Event queue where operations are recorded

Sandia National Laboratories

Memory Descriptor Options

- **`PTL_MD_OP_PUT`**
  - Respond to put operations
- **`PTL_MD_OP_GET`**
  - Respond to get operations
- **`PTL_MD_MANAGE_REMOTE`**
  - Offset is specified in the request
- **`PTL_MD_TRUNCATE`**
  - Truncate incoming request to size of MD
- **`PTL_MD_ACK_DISABLE`**
  - Do not allow put acknowledgements
- **`PTL_MD_IOVEC`**
  - Start and length refer to a `ptl_md_iovec_t`
- **`PTL_MD_MAX_SIZE`**
  - Recognize the `max_size` value in the MD
- **`PTL_MD_EVENT_START_DISABLE`**
  - Turn off start events
- **`PTL_MD_EVENT_END_DISABLE`**
  - Turn off end events

Sandia National Laboratories

Scatter/Gather MD

- typedef struct {
    void   *iov_base;
    ptl_size_t   iov_len;
  } ptl_md_iovec_t;

Sandia National Laboratories

Memory Descriptor Creation Functions

- int PtlMDAttach(
  ptl_handle_me_t me_handle,
  ptl_md_t md,
  ptl_unlink_t unlink_op,
  ptl_handle_md_t *md_handle
  );

- int PtlMDBind(
  ptl_handle_ni_t ni_handle,
  ptl_md_t md,
  ptl_unlink_t unlink_op,
  ptl_handle_md_t *md_handle
  );

Sandia National Laboratories

Memory Descriptor Destruction Function

- `int PtlMDUnlink( ptl_handle_md_t md_handle );`

Sandia National Laboratories

Function to Change an Existing MD

```
int PtlMDUpdate(
    ptl_handle_md_t     md_handle,
    ptl_md_t            *old_md,
    ptl_md_t            *new_md,
    ptl_handle_eq_t     eq_handle
  );
```

- Primarily needed to allow for the atomic search-and-post function needed by MPI

Sandia National Laboratories

Event Queue (EQ)

- Circular queue that records operations on MDs
- Signal the start and end of data transmission into or out of an MD
- Finite number of entries
- Overflowing an event queue will overwrite events

Sandia National Laboratories

Event Entry Contents

- Event type
- Initiator of event (nid,pid)
- Portal table index
- Match bits
- Requested length
- Manipulated length
- Offset
- 64 bits of out-of-band data

Sandia National Laboratories

Type of Events

- PTL_EVENT_GET_START
- PTL_EVENT_GET_END
- PTL_EVENT_PUT_START
- PTL_EVENT_PUT_END
- PTL_EVENT_GETPUT_START
- PTL_EVENT_GETPUT_END
- PTL_EVENT_REPLY_START
- PTL_EVENT_REPLY_END
- PTL_EVENT_SEND_START
- PTL_EVENT_SEND_END
- PTL_EVENT_ACK
- PTL_EVENT_UNLINK

Sandia National Laboratories

Event Structure

```
• typedef struct {
        ptl_event_kind_t            type;
        ptl_process_id_t            initiator;
        ptl_uid_t                   uid;
        ptl_jid_t                   jid;
        ptl_pt_index_t              pt_index;
        ptl_match_bits_t            match_bits;
        ptl_size_t                  rlength;
        ptl_size_t                  mlength;
        ptl_handle_md_t             md_handle;
        ptl_md_t                    md;
        ptl_hdr_data_t              hdr_data;
        ptl_seq_t                   link;
        ptl_ni_fail_t               ni_fail_type;
        volatile ptl_seq_t          sequence;
} ptl_even_t;
```

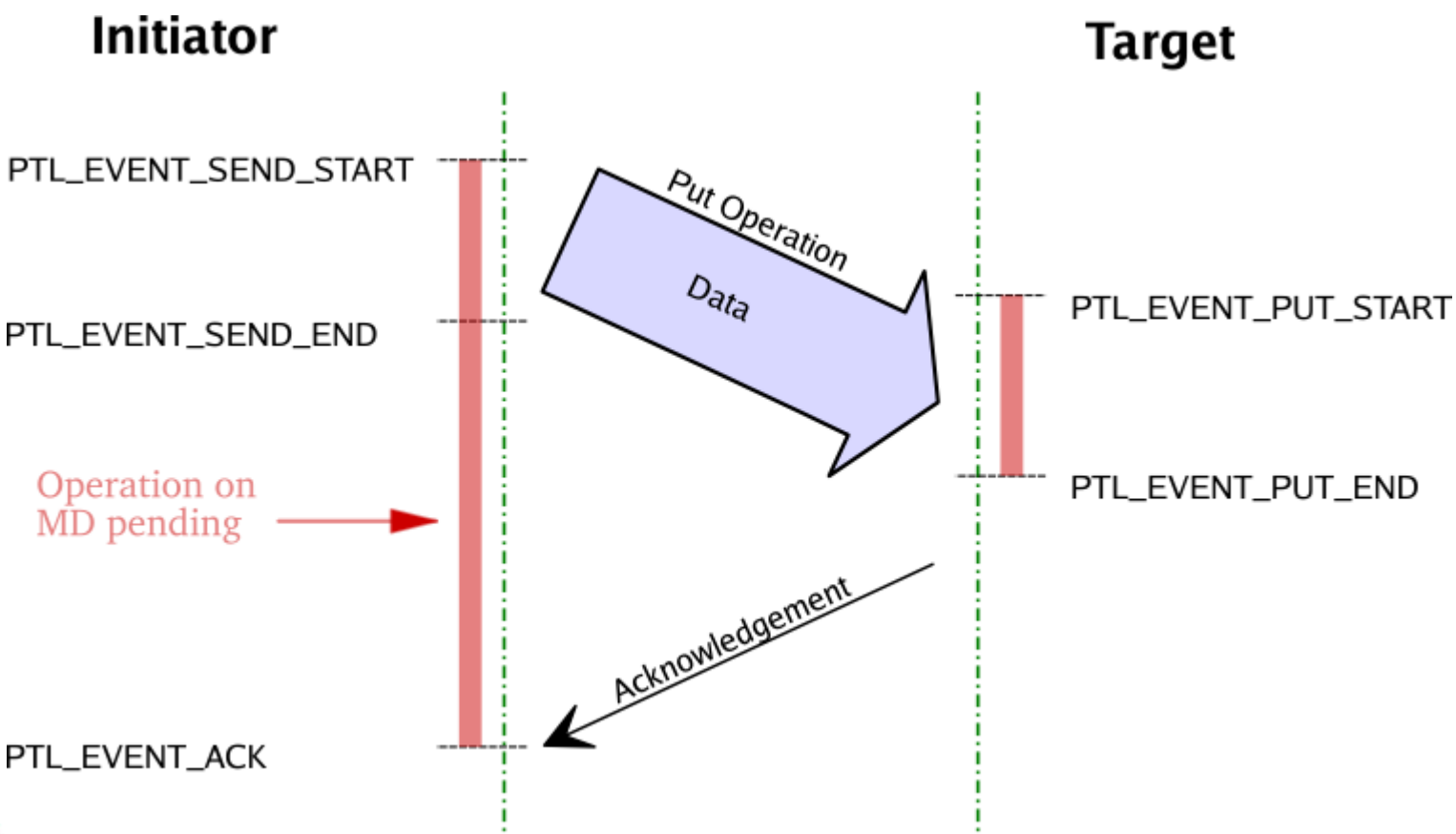
Sandia National Laboratories

EQ Handlers

- **Not covered in this tutorial**

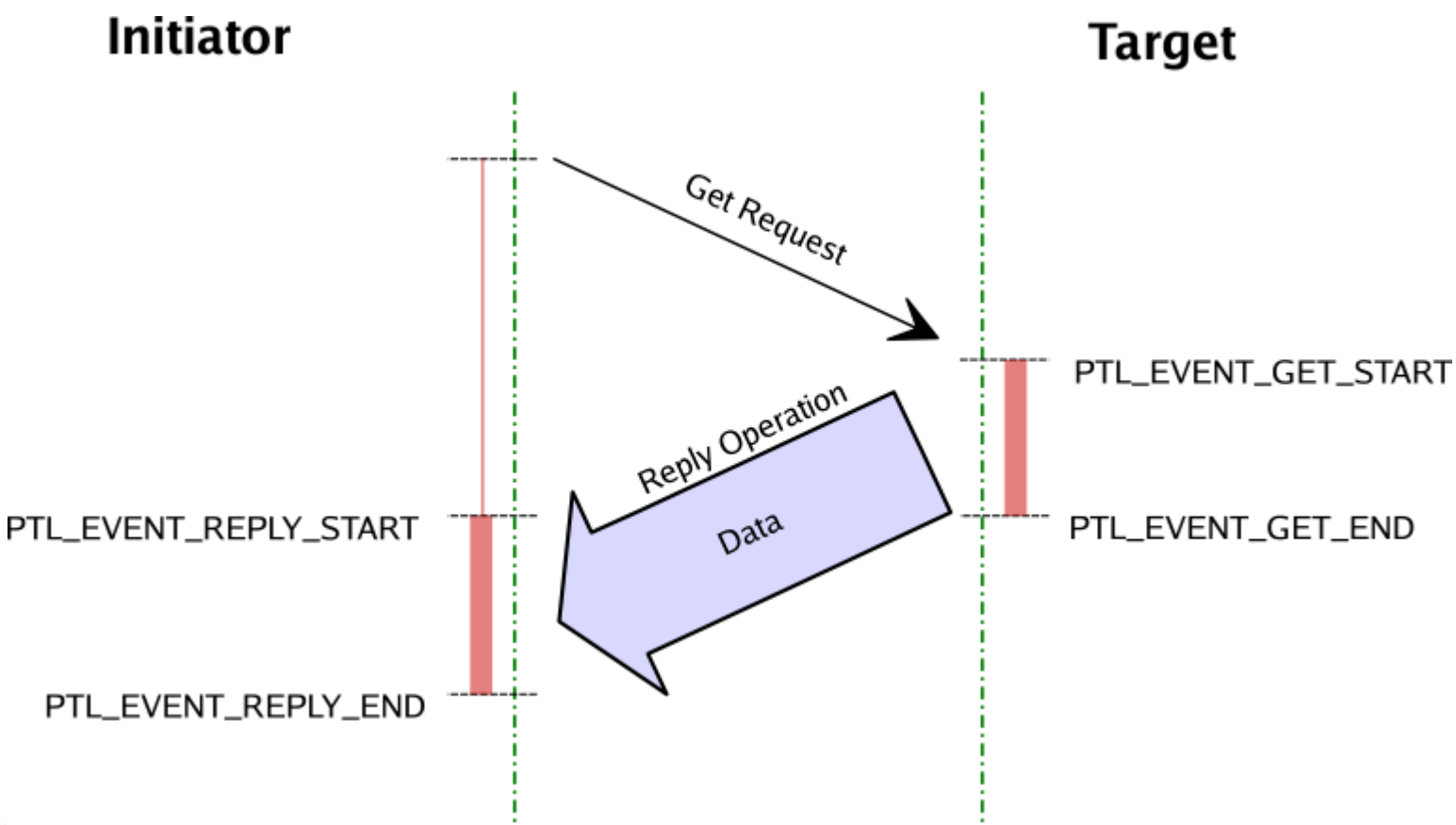
Sandia National Laboratories

Events for a Put

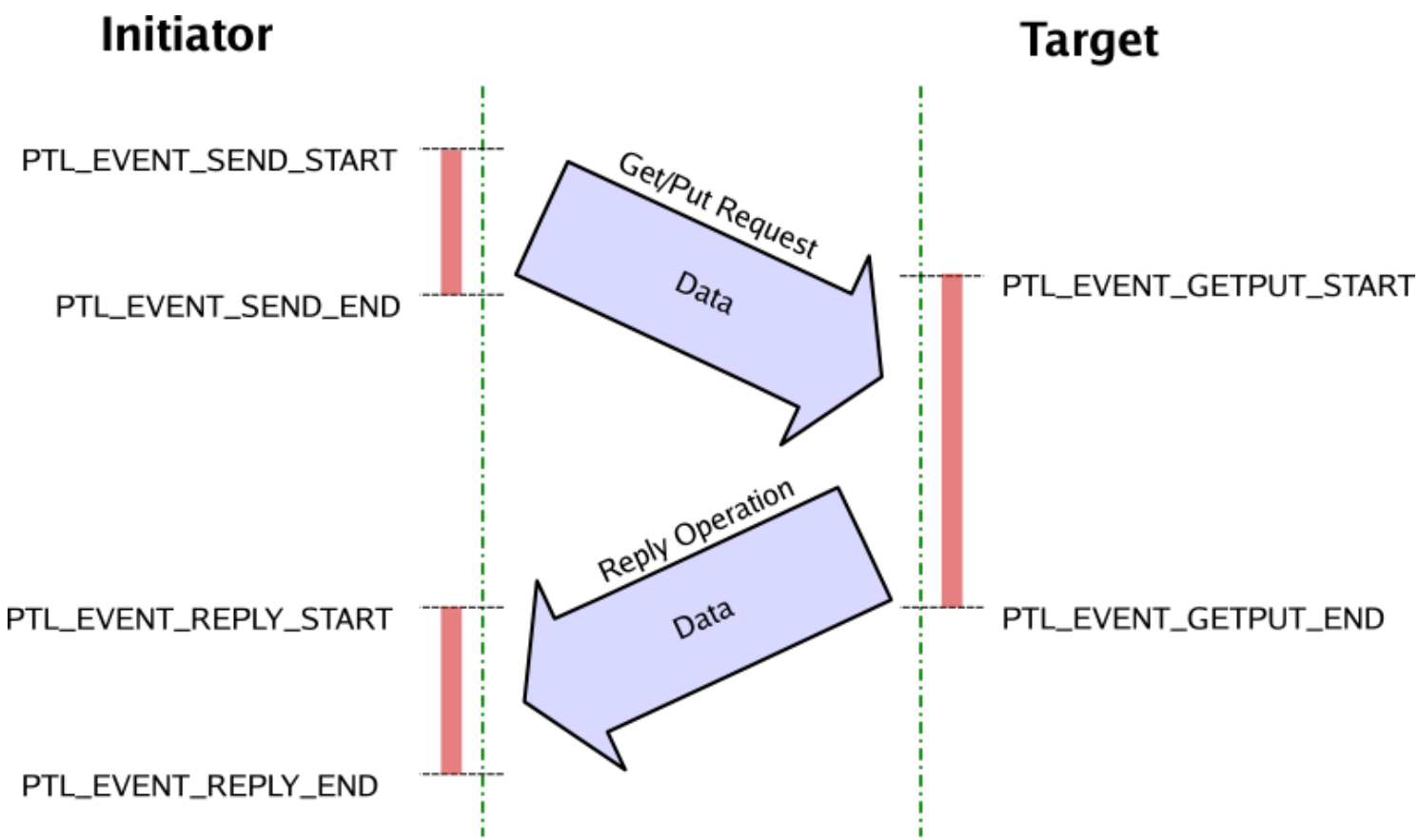


Sandia National Laboratories

Events for a Get



Events for a GetPut



EQ Creation and Destruction Functions

```
• int PtlEQAlloc(
      ptl_handle_ni_t          ni_handle,
      ptl_size_t               count,
      ptl_eq_handler_t         eq_handler,
      ptl_handle_eq_t          *eq_handle
  );

• int PtlEQFree(
      ptl_handle_eq_t          *eq_handle
  );
```

Sandia National Laboratories

Functions for Obtaining Events

- int PtlEQGet(
    ptl_handle_eq_t eq_handle,
    ptl_event_t *event
  );

- int PtlEQWait(
    ptl_handle_eq_t eq_handle,
    ptl_event_t *event
  );

- int PtlEQPoll(
    ptl_handle_eq_t *eq_handles,
    int size,
    ptl_time_t timeout,
    ptl_event_t *event,
    int *which
  );

Sandia National Laboratories

Events, Packets, and Failure

- Portals ensures best effort delivery
- Underlying network may break messages into packets
- Two consequences
  - Out of order completion
    - Consequence of zero copy and packets
  - May trash application's memory
    - Consequence of failure, zero copy, and packets
- Event sequences
  - Start, followed by end (success) or fail
  - e.g., PUT_START, PUT_END
  - Failure is absolute
  - Nothing happens after end or fail

Sandia National Laboratories

Access Control

- Controls which processes are allowed to perform operations
- Default entry of 0 allows for all processes with the same user ID to communicate
- Operations that fail due to access control are not user-visible

Sandia National Laboratories

Access Control Table Manipulation Function

```
int PtlACEntry(
    ptl_handle_ni_t          ni_handle,
    ptl_ac_index_t           ac_index,
    ptl_process_id_t         match_id,
    ptl_uid_t                uid,
    ptl_jid_t                jid,
    ptl_pt_index_t           pt_index
  );
```

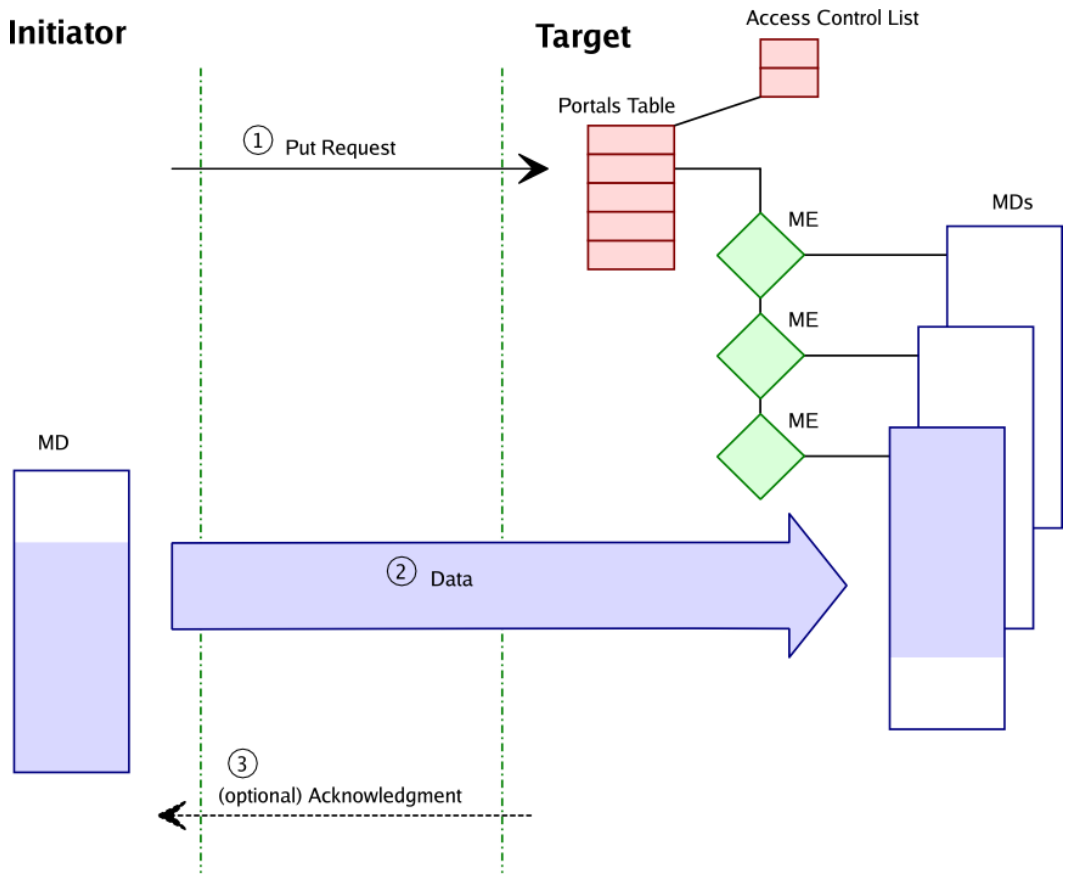
Sandia National Laboratories

Data Movement Operations

- **Put**
  - **Initiator sends data to a remote portal**
  - **Can receive an optional acknowledgment**
- **Get**
  - **Initiator sends request for remote data from a portal**
  - **Data is delivered to local memory descriptor**
- **GetPut**
  - **Atomic swap**

Sandia National Laboratories

Put Operation



Sandia National Laboratories

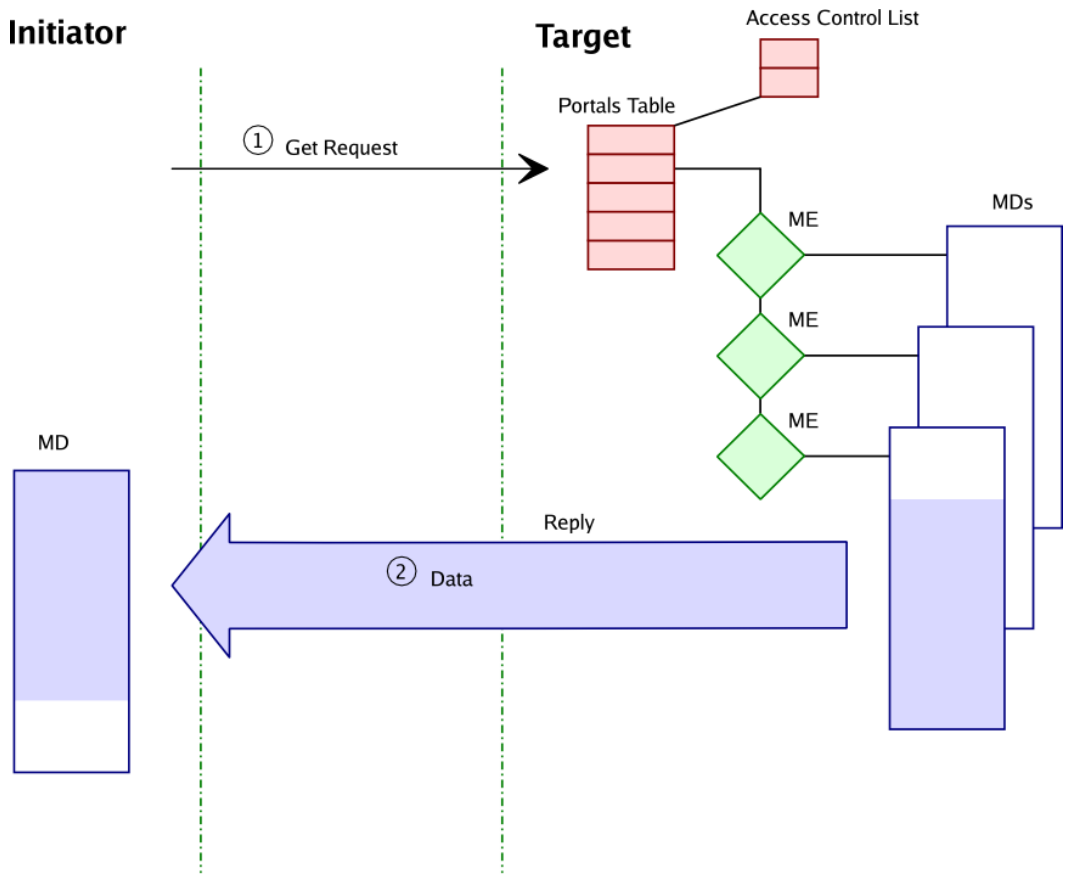
Put Functions

- int PtlPut(
  ptl_handle_md_t md_handle,
  ptl_ack_req_t ack_req,
  ptl_process_id_t target_id,
  ptl_pt_index_t pt_index,
  ptl_ac_index_t ac_index,
  ptl_match_bits_t match_bits,
  ptl_size_t remote_offset,
  ptl_hdr_data_t hdr_data
  );

- int PtlPutRegion(
  ptl_handle_md_t md_handle,
  ptl_size_t local_offset,
  ptl_size_t length,
  ptl_ack_req_t ack_req,
  ptl_process_id_t target_id,
  ptl_pt_index_t pt_index,
  ptl_ac_index_t ac_index,
  ptl_match_bits_t match_bits,
  ptl_size_t remote_offset,
  ptl_hdr_data_t hdr_data
  );

Sandia National Laboratories

Get Operation

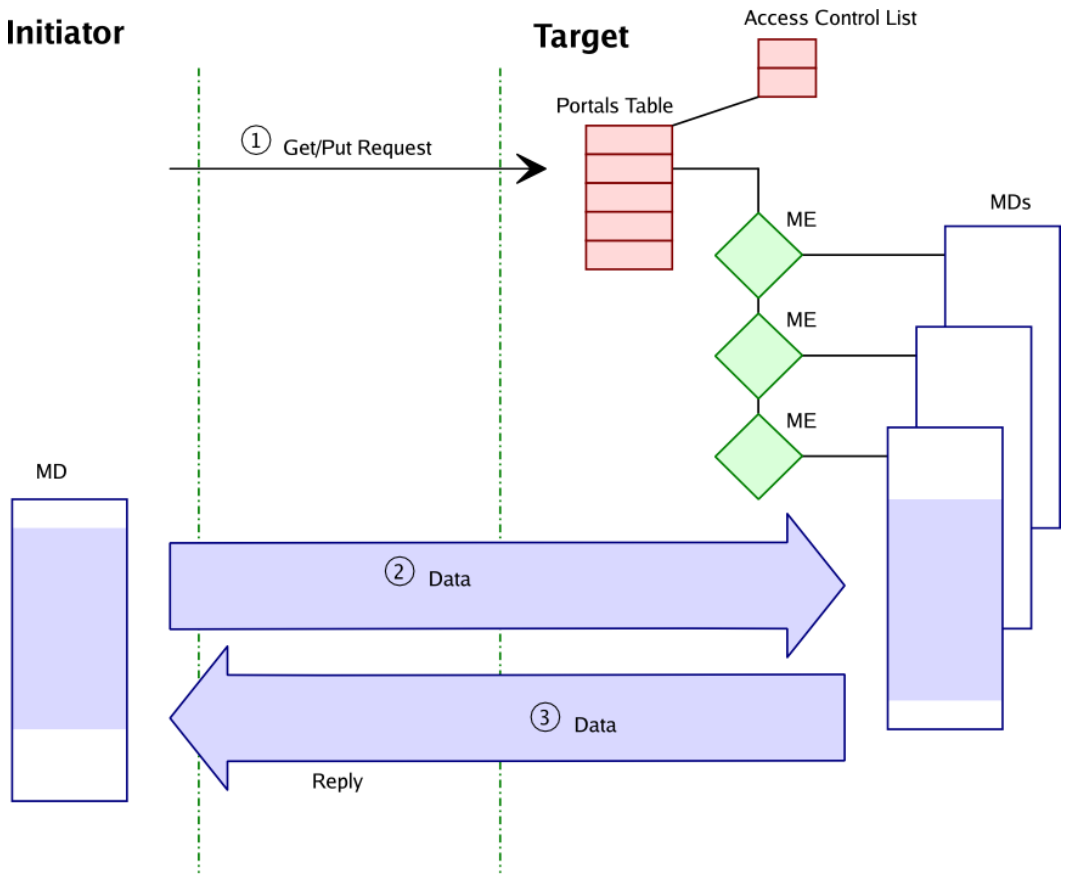


Get Functions

- int PtlGet(
  - ptl_handle_md_t md_handle,
  - ptl_process_id_t target_id,
  - ptl_pt_index_t pt_index,
  - ptl_ac_index_t ac_index,
  - ptl_match_bits_t match_bits,
  - ptl_size_t remote_offset
  -);
- int PtlGetRegion(
  - ptl_handle_md_t md_handle,
  - ptl_size_t local_offset,
  - ptl_size_t length,
  - ptl_process_id_t target_id,
  - ptl_pt_index_t pt_index,
  - ptl_ac_index_t ac_index,
  - ptl_match_bits_t match_bits,
  - ptl_size_t remote_offset,
  -);

Sandia National Laboratories

GetPut Operation



Sandia National Laboratories

GetPut Function

```
• int PtlGetPut(
      ptl_handle_md_t          get_md_handle,
      ptl_handle_md_t          put_md_handle,
      ptl_process_id_t         target_id,
      ptl_pt_index_t           pt_index,
      ptl_ac_index_t           ac_index,
      ptl_match_bits_t         match_bits,
      ptl_size_t               remote_offset
      ptl_hdr_data_t           hdr_data
    );
```

Sandia National Laboratories

Summary

- Minimal library space
  - Nothing depends on message size
  - All objects can be confirmed when created
- Designed for library writers
  - Not for application developers
  - Low-level API
    - We're happy to drop requests
    - Structures are complicated
    - Some functions (`PtlMDUpdate()`) are not obvious
- Designed to reflect underlying hardware
  - NICs
  - Packets and failure
- Provide the right amount of protection

Sandia National Laboratories

What Portals Does

- Separates communication space from computation space
  - Moderately dynamic
    - During descriptor construction
    - Any part of an application's memory can be used for communication
  - Simplifies coherence issues
    - Important for PCI implementations as well
- Handles important protocol processing
  - MPI long message strategy
    - Force rendezvous at receiver
    - Post and forget
  - Supports parallel servers

Sandia National Laboratories

What Portals Doesn't Do

- Dynamic integration of computation and communication space
  - May be needed for things like UPC
  - Race conditions
  - Memory consistency models
- Poor support for collectives
  - Each process must actively participate in collective operation
  - Would prefer to have a “contribute and forget” capability
  - Reduce variance in time for collective operations

Sandia National Laboratories

Example: Implementing MPI on Portals

Sandia National Laboratories

MPI on Portals

- Fundamental problem:
  - Rendezvous between sender and receiver
- Receiver can wildcard sender, sender cannot wildcard receiver
  - Destination is known
  - Origin may not be known
- Rendezvous must occur at the receiver
  - Easy if receiver is ready when send starts: use eager send
  - MPI standard recommends pre-posted receives

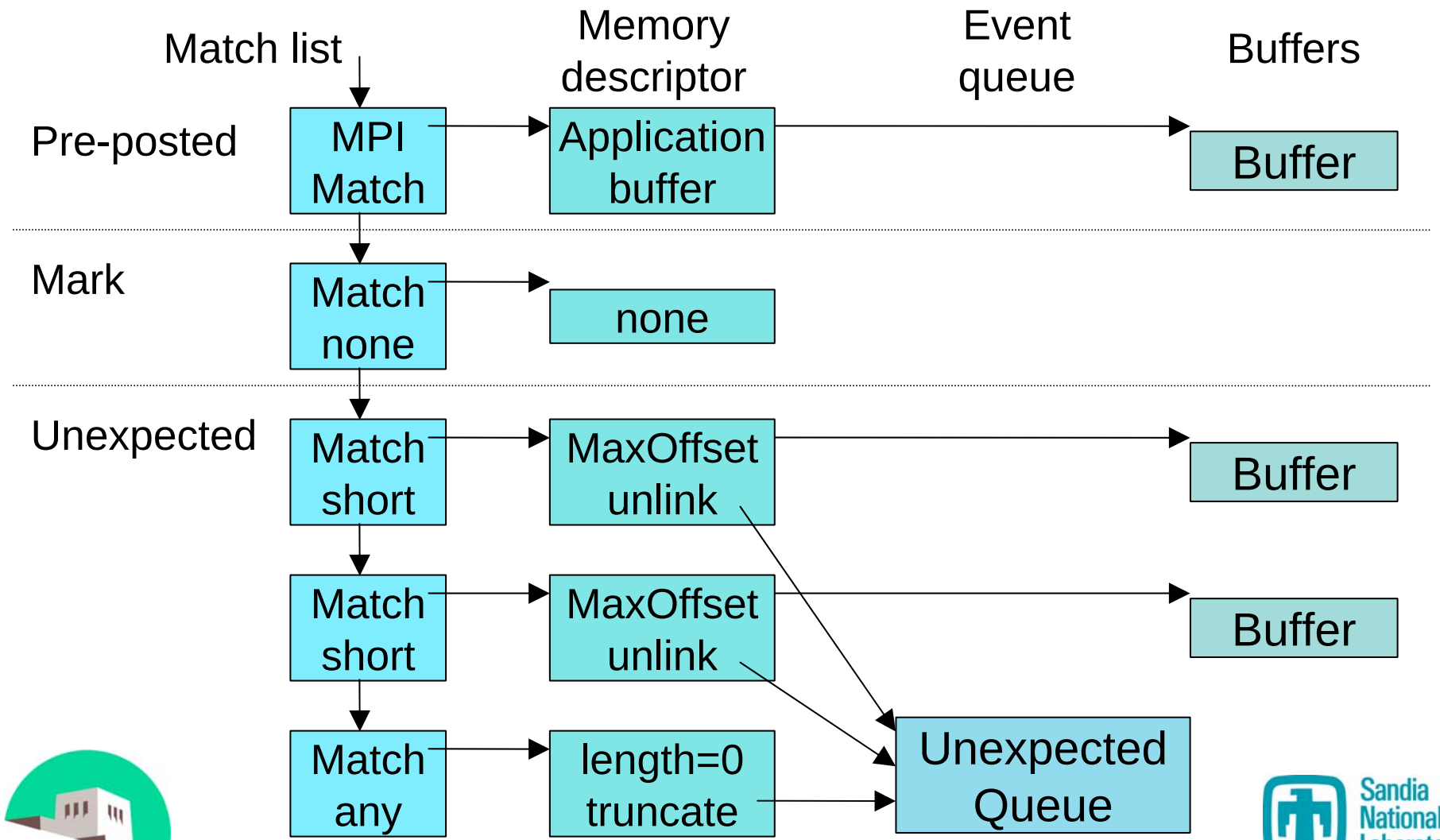
Sandia National Laboratories

How to do MPI

- Two portal entries
  - One for receiving messages
  - One for unexpected long messages
- Match list for receive portal
  - Expected, pre-posted, receives (MPI matching)
  - Unexpected short messages
  - Unexpected long messages
- Long MPI send – eager
  - Build an MD and add to unexpected long message portal
  - Perform the put
  - Unexpected messages are dropped and later read
- Short MPI send is eager
  - Build a free-floating MD
  - Perform the put
  - Unexpected messages held for matching receive
- Very short MPI send is eager
  - Copy user data into pre-allocated MD
  - Perform the put

Sandia National Laboratories

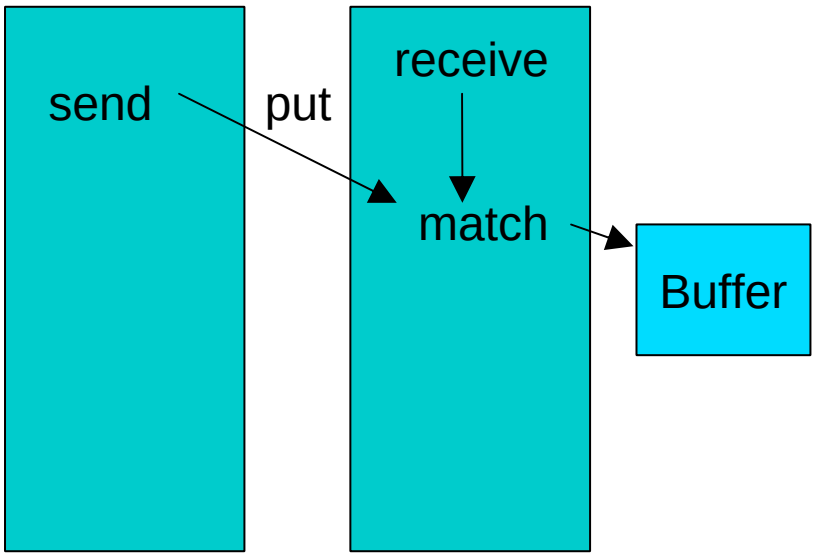
MPI Receive in Portals



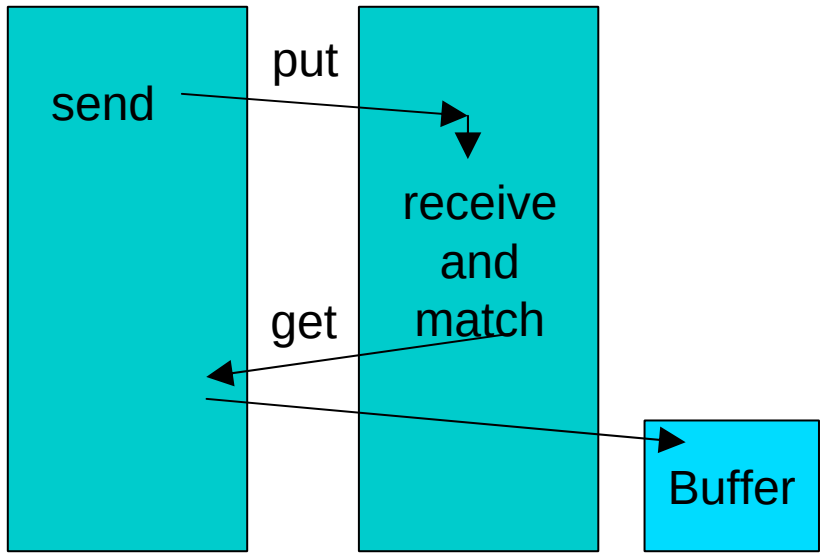
Sandia National Laboratories

MPI Long Message Protocol

Pre-posted Receive



Late Receive



Sandia National Laboratories

Good Things to do with Portals

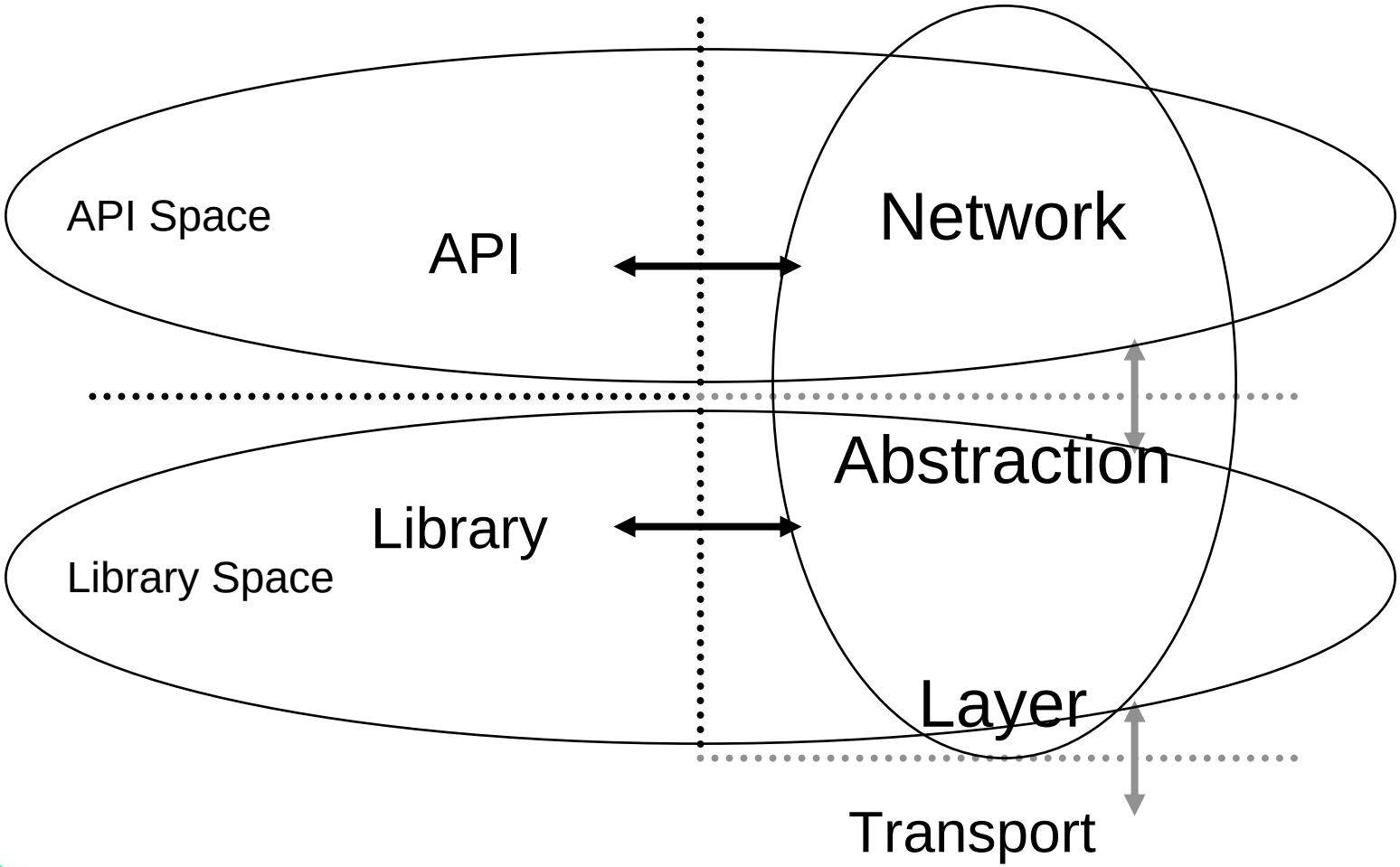
- Use catchalls at the end of every match list
  - Never have to worry about “losing” messages
  - Can be used to generate “negative” acknowledgments
- Use persistent MDs as much as possible
  - Catamount supports creating an MD over entire memory regions (data, stack, heap)
  - Can eliminate overhead of creating an MD each time

Sandia National Laboratories

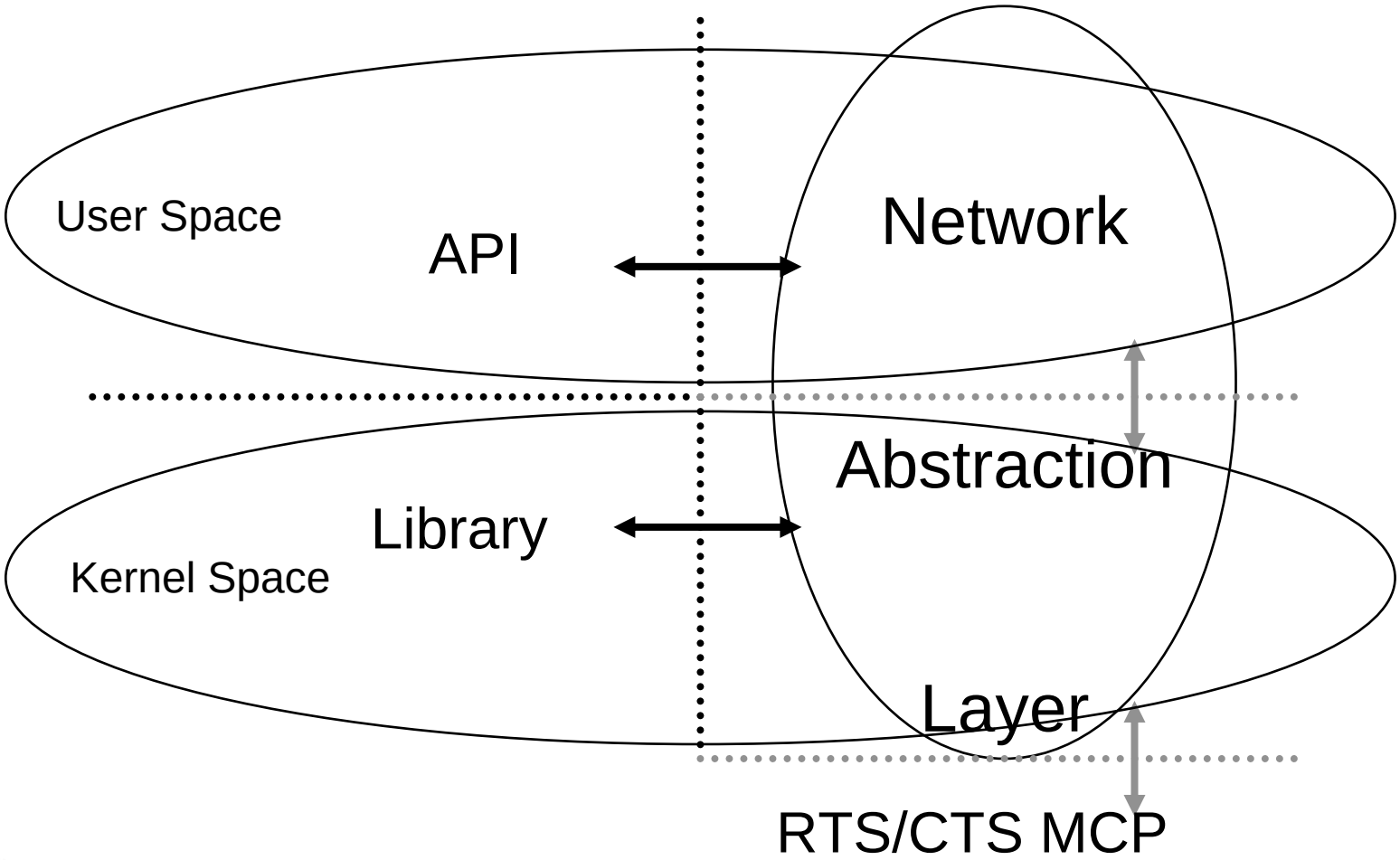
XT3 Implementation

Sandia National Laboratories

Portals Reference Implementation Design

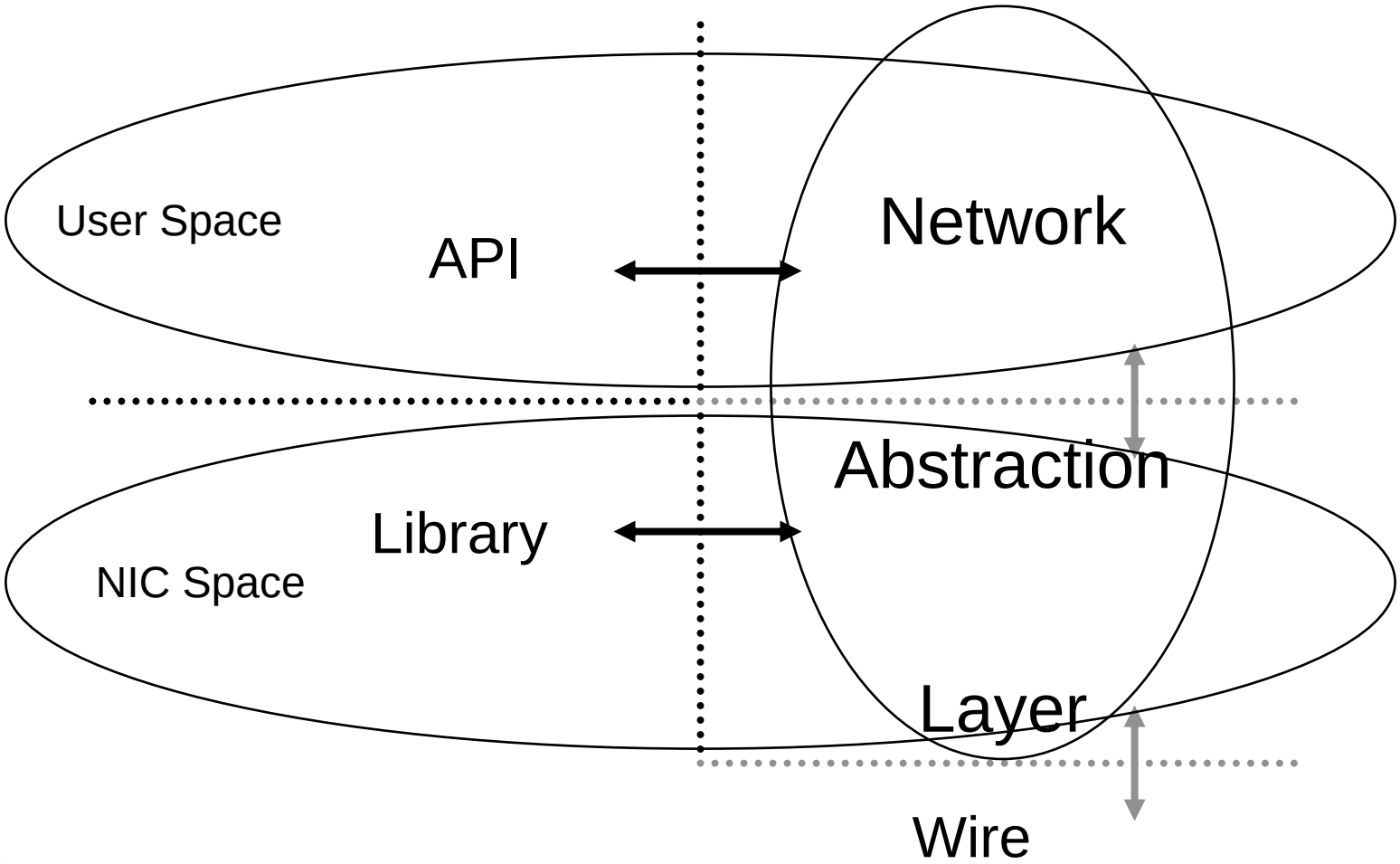


Myrinet Kernel Implementation



Sandia National Laboratories

Myrinet MCP Implementation



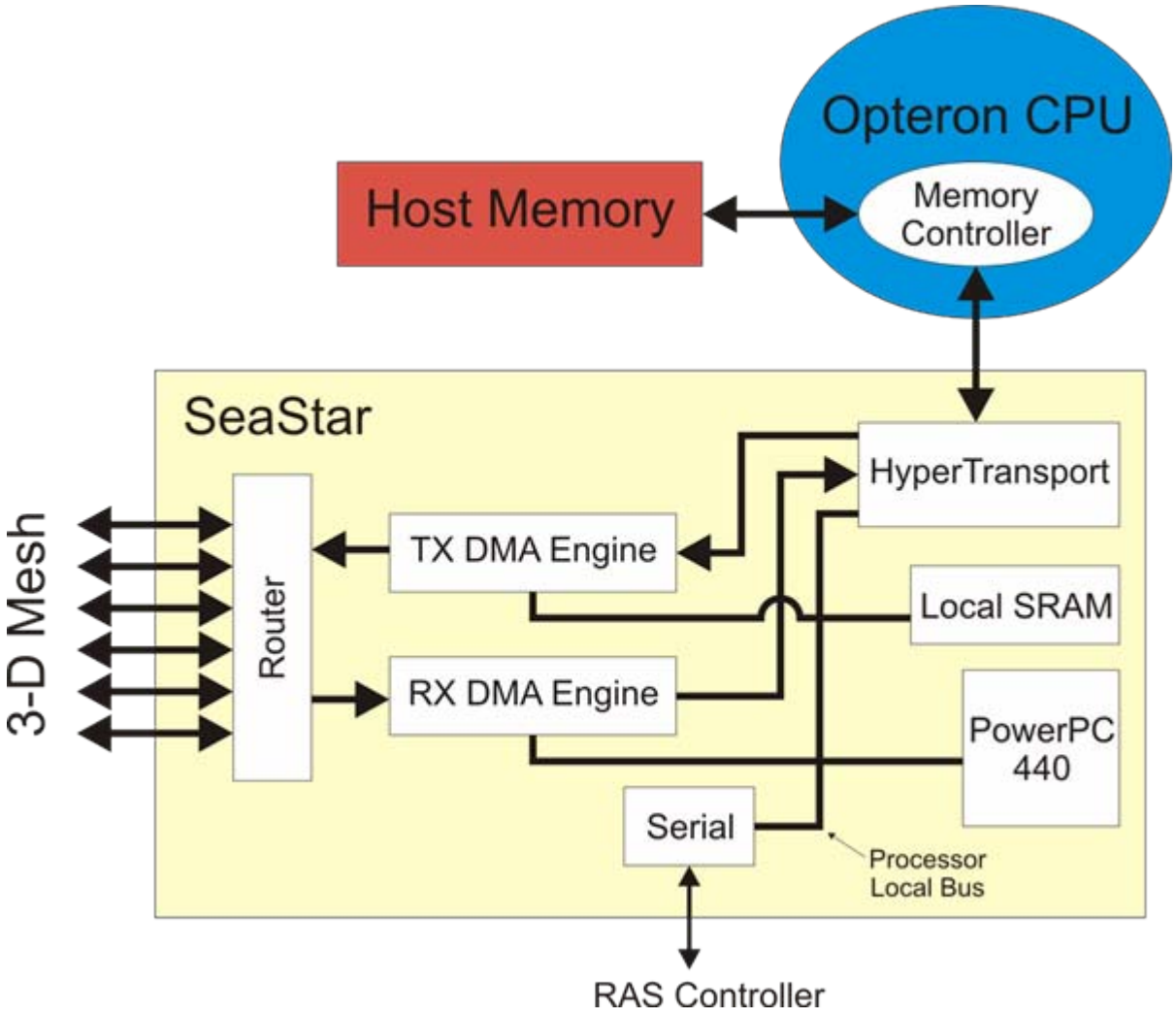
Sandia National Laboratories

Cray SeaStar NIC/Router

- 16 1.6 Gb/s HyperTransport to Opteron
- 500 MHz embedded PowerPC 440
- 384 KB on-board scratch RAM
- Seven-port router
- Six 12-channel 3.2 Gb/s high-speed serial links

Sandia National Laboratories

SeaStar Block Diagram



Portals 3.3 for SeaStar

- Cray started with Sandia reference implementation
- Needed single version of NIC firmware that supports all combinations of
  - User-level and kernel-level API
  - NIC-space and kernel-space library
- Cray added bridge layer to reference implementation to allow NAL to interface multiple API NALs and multiple library NALs
  - qkbridge for Catamount applications
  - ukbridge for Linux user-level applications
  - kbridge for Linux kernel-level applications

Sandia National Laboratories

SeaStar NAL

- Portals processing in kernel-space
  - Interrupt-driven
  - “generic” mode
- Portals processing in NIC-space
  - No interrupts
  - “accelerated” mode

Sandia National Laboratories

Upcoming

- "Accelerated" Portals – see talk on Thursday
- Portals collective library
  - Collective operations built on top of Portals
- Non-blocking collective functions
  - Collective operations integrated into Portals
  - SeaStar can support offloading collective operations
  - Barrier proof-of-concept is done and working
- Portals 4.0
  - Laundry list of issues with Portals 3.3 is too big
  - Unnecessary symmetry (`PTL_EVENT_SEND_START`)
  - Unneeded operations (arbitrary list insertion)

Sandia National Laboratories

XT3 Specifics

- Differences from the current specification
  - Send events are generated on a get
  - AC table is not implemented
- Keep N at 256 for N-to-1
- Use copy block for short messages to avoid wire-level acknowledgements
- No `PTL_EVENT_UNLINK`
- PtlNIDist() is not implemented
- Not all fields returned in an event structure are valid

Sandia National Laboratories

XT3 Portals Limits

- Max MEs: 2048
- Max MDs: 2048
- Max EQs: 512
- Max PT index: 128
- Max IOVECs: unlimited
- Max ME list: 2048
- Max GetPut size: 8
- Max outstanding messages: 2048

Sandia National Laboratories

Compiling for XT3

- #include”portals/portals3.h”
- module add PrgEnv-gcc or PrgEnv-pgi
- Must be compiled with qk-gcc
  - PGI and GNU don’t align consistently

Sandia National Laboratories

Compute Node OS Library

- #include”catamount/cnos_mpi_os.h”
- int cnos_get_rank()
- int cnos_get_size()
- cnos_nidpid_map_t *nidpid;
- nidpid_size = cnos_get_nidpid_map(&nidpid);
- int cnos_barrier()

Sandia National Laboratories

Questions?

Sandia National Laboratories