OPENFABRICS ALLIANCE

2025 OFA Webinar Series

FROM SLINGSHOT TO ULTRA ETHERNET: BUILDING A TRANSPORT STANDARD FOR LIBFABRIC

Keith D Underwood

Senior Distinguished Technologist

HPE

MEET THE PRESENTER: Keith D. Underwood, HPE

Keith Underwood is a Senior Distinguished Technologist in the HPE High Performance Networking Business Unit where he leads next generation NIC architecture definition. He is the editor of the UEC Transport Semantic specification and an active contributor to the UEC Transport definition. Prior to joining Cray, Keith was a founding member of the Omni-Path team at Intel and led the Omni-Path 2 NIC architecture. He was part of the team that created the Portals 4 API that seeded the Libfabric definition, and was a key contributor to developing the MPI-3 RMA extensions.

INTRODUCTION TO THE ULTRA ETHERNET CONSORTIUM (UEC)

A GROWING CONSORTIUM

*not all members listed
snapshot as of 2024-10

Mission:
Advance an Ethernet-Based Open, Interoperable, High-Performance Full-Stack architecture to meet the Growing Demands of AI and HPC at Scale

Ultra Ethernet Consortium

>100 member companies
>1300 active participants

Source:ultraethernet.org

BUILDING A STANDARD FOR A FULL NETWORK STACK

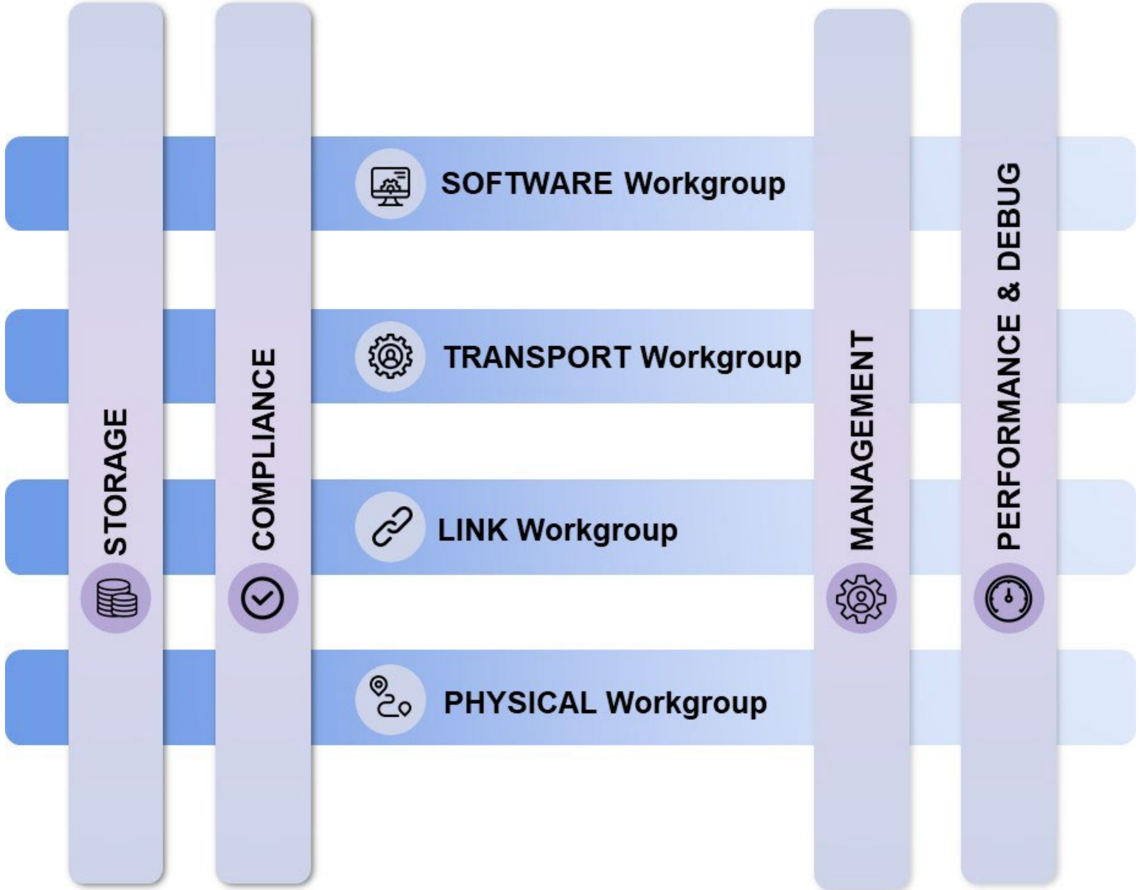
Enabling AI and HPC at the Largest Scale

- **The first new network standard in 25 years**
  - Released 1.0 June,2025
- **Many active working groups covering the entire stack**
  - Many layers
  - One specification

tiny compared to existing standards

- **The spec is ~~big~~** ^

UEC is a JDF project and an International Standards Organization



A LITTLE HISTORY

ULTRA ETHERNET: AN ORIGIN STORY

- **In the early 1990s…**
  - Nobody gave webinars….
  - Ethernet looked like this…
  - And supercomputers didn't use Ethernet
- **A supercomputer was shipped that used connections**
  - And didn't have enough memory to launch an application
  - People learned that connections were bad
    - Although not everyone got the memo
  - A new operation system was born (SUNMOS)
    - With a networking stack that did not use connections
    - They retro-named that networking stack Portals 0.0

CONNECTIONS AND NETWORKS: WHAT WENT WRONG?

When you have a hammer, everything looks like a nail

- **If you give an architect a connection…**
  - They will want to make that connection ordered (especially if it is 1999)
  - Just to keep the hardware simple
- **When they have an ordered connection, they will want to optimize around it….**
  - They will only put the addressing information on the first packet of the message
  - They will use the ordered connection to manage resource exhaustion
  - They will embed the ordered connection in the error model
- **Once they have used the connection for optimization, they will want to tangle it up with the semantics…**
  - They will use the connection to pick the destination buffer
  - And, the security of that buffer will depend on the connection
- **When they tangle the connection up with the semantics…**
  - They will want the user to set it up
- **When user software programs around the connection…**
  - It is hard to find your way back out

Image: AI Generated

SEMANTICS VS RELIABILITY: A NECESSARY SEPARATION

Reliability and Ordering

- **Did the packet get there?**
  - Loss detection
  - Retransmits
  - When do I need to give up on a packet getting there?
- **Did it need to be in order?**
  - If so, was it?
- **Was it a duplicate?**
- **Typical tools:**
  - Sequence numbers
  - Probe packets
  - Timeouts
  - Retransmit limits

Semantics

- **What do I do with that packet?**
  - Is it a part of a larger message?
  - How big is the message?
  - Where does it go in the message?
- **Where is that message going at the target node?**
  - Which job and which process in that job?
  - Which queue in that process?
- **What is that message supposed to do?**
  - Is it a write or a read?
  - A send? or a tagged send that needs matching?
  - Or, is it an atomic? If so, what is the operation and datatype?
- **What is the behavior of failure?**
  - What do I do if the target process wasn't there?
  - How do I handle resource exhaustion?
  - What happens if there was an address translation failure?

DELIVERING RELIABILITY AT MASSIVE SCALE

Semantic Sublayer sees a Reliable Network

- **Semantic layer makes packets, and the reliability layer makes sure they get there**
  - Many approaches available, and the choice is transparent to the upper layers
- **Not that long ago: "just build a reliable network" was viable**
  - Lasted from ASCI Red to Cray Aries
  - Careful design and link level retry meant that all the packets got through
  - 50 Gbps signaling (and beyond) is a more difficult regime
- **Isolating reliability at the network layer enables less state**
  - Reliability can be NIC-to-NIC instead of process to process

Architecting to Minimize Reliability State

- **Reliability state can be trivially small for ordered networks**
  - Per peer sequence number
  - Tracking of all outstanding packets
  - Examples:
    - Cray Seastar: Portals 3.3 and reliability in 384K of memory
    - OPA2-classic: 16 bytes of state per peer NIC
- **NIC-to-NIC reliability reduces state versus process to process connections**
  - Assume 10K NICs with 100 processes per NIC
    - NIC-to-NIC: each NIC has 10K connections
    - Process-to-Process: each NIC has 100 x 100 x 10,000 connections
- **Slingshot introduced the era of dynamic connections**
  - Removes scaling ceiling for large NIC counts
  - Eliminates need to wire up all of the NICs

INTRODUCTION TO SLINGSHOT: ETHERNET FOR EXASCALE

ETHERNET ALWAYS WINS

A Statement of Economic Realities

- **1995: Beowulf Clusters Emerge**
  - 10baseT and 10base2
- **2004: the management network (e.g., Cray XT3)**
  - No need to re-invent the wheel
- **2005-2021: the marketing era**
  - Gigabit Ethernet flooded the bottom half of the Top500 list
- **June, 2021: Ethernet hits #5**
  - NERSC Perlmutter with Slingshot network
- **June, 2022: Ethernet Reaches Exascale**

- **Modern high bandwidth links use:**
  - An Ethernet SerDes
  - Connectors made for Ethernet
  - Cable technology designed around Ethernet
  - Optical modules driven by Ethernet
  - And, because of that, Ethernet FEC
- **Ethernet has plenty of room to:**
  - Negotiate features on a port (LLDP)
  - Encode additional features alongside Ethernet on the same wire
- **Some Ethernet limitations are just about design choices**
  - FEC latency is inherent, but switch latency is a choice
  - Ethernet overheads can be negotiated down, and packet rate is a switch design choice

SLINGSHOT IS ETHERNET FOR HPC

True Ethernet *and* True HPC on Every Port – All the Time

- **Carry standard Ethernet frames**
  - Plug into the datacenter Ethernet backbone without gateways
  - Leverage Ethernet connected storage
- **Run standard IP stacks**
  - TCP and UDP on standard IP packets on standard Ethernet frames
  - Even RoCEv2
- **Directly connect third-party Ethernet NICs**
  - In fact, this was how Perlmutter was originally deployed
- **Concurrently carry HPC-enhanced frames**
  - Optimized headers for tiny payloads
  - Reduced interframe gap
  - Link level reliability
  - Credit based flow control
- **Implement a native transport for libfabric**
  - Provide efficient support for MPI, SHMEM, and CCLs
  - Tag matching offload
  - Rendezvous offload
  - Atomic operation offload
  - Runs alongside TCP and IP
- **Advanced traffic management for both TCP and HPC flows**
  - No endpoint modification required
  - Adaptive routing while preserving ordering
  - Advanced congestion management to deliver high bandwidth and low latency under load

FROM SLINGSHOT TO UEC

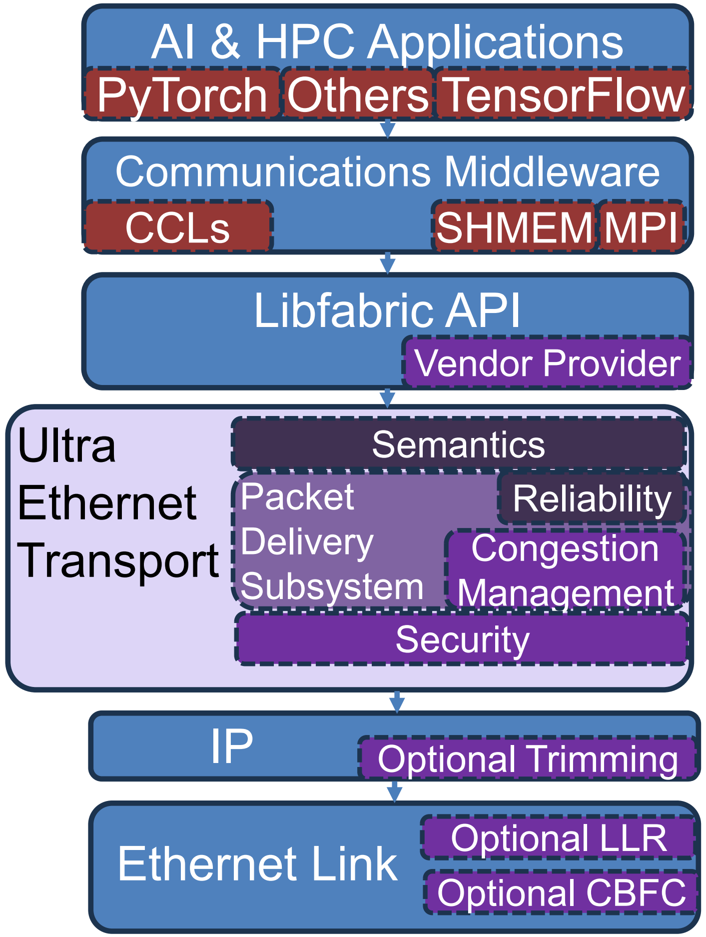
THE CONCEPTION OF ULTRA ETHERNET

- **Basic conversation structure (February, 2022):**
  - AI is booming, and it needs a **lot** of network
  - Ethernet has performance limiting characteristics and lacks some features traditionally found in HPC systems
  - InfiniBand is “different”: datacenters don’t like snowflakes
  - Slingshot is both Ethernet and Exascale – can we standardize those capabilities?
    - Link Level Retry
    - Credit base flow control
    - Optimized frame formats
  - Me: maybe we can even do a new semantic layer and network transport?
    - <eye roll you could hear over the phone> “you’re welcome to propose that…”
- **September, 2022: our first UEC face to face meeting**
  - The tone had shifted: there was clear interest in developing a modern RDMA definition
  - We also standardized enhancements to the link
- **We never expected it to be easy, and it never was**
  - But, worked with some great people along the way

ASSEMBLING THE ULTRA ETHERNET STACK

Successful Standards Build on Proven Technology

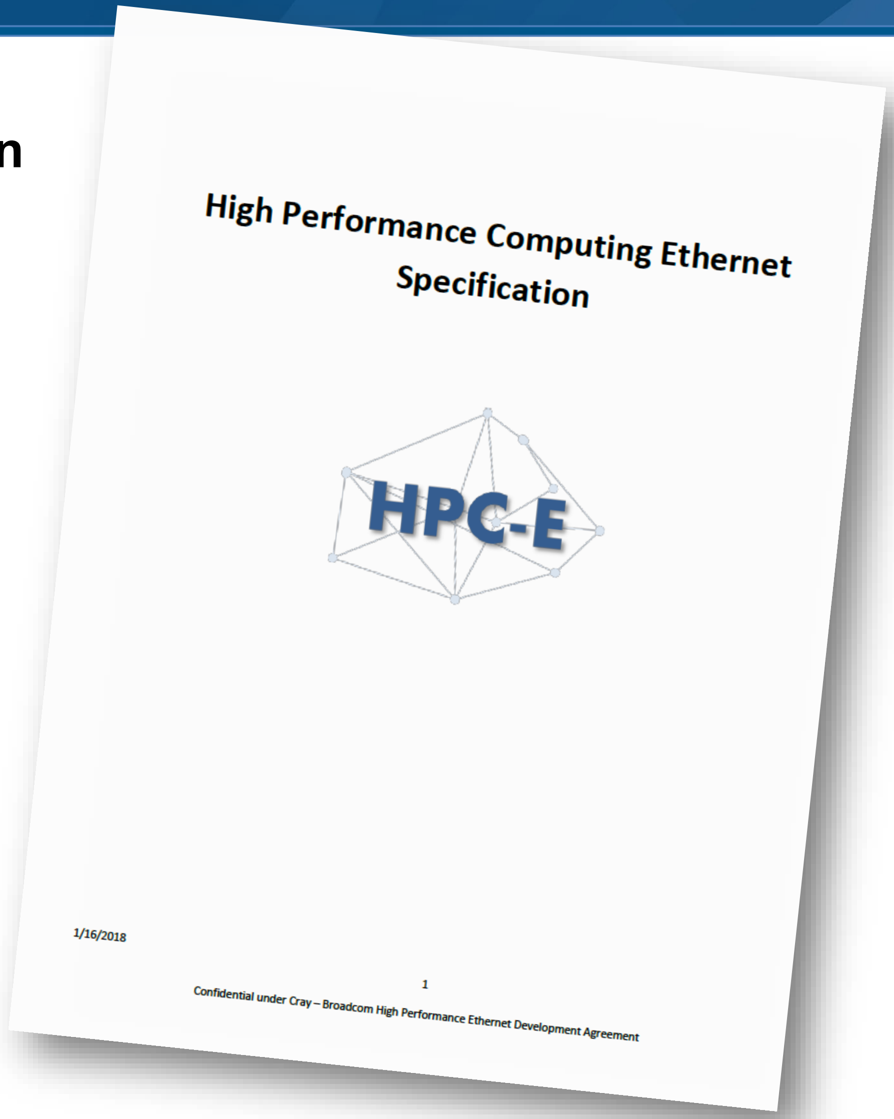
- **AI and HPC application stacks – unmodified**
- **Communication middleware – unmodified**
- **Libfabric – an open-source network API with unordered operations**
  - Proven in the Cloud and in HPC
- **Proven in HPC at Exascale:**
  - Semantic layer to support libfabric
  - "Connectionless" reliability architecture
- **Driven by AI in the datacenter:**
  - Congestion management solutions for Best-Effort Ethernet
  - Scalable shared key security
- **Ethernet and IP: no modification required**
  - But, some optional technology to help
    - Optional link level retry
    - Optional credit-based flow control
    - Optional packet trimming



ENHANCED EFFICIENCY AND RESILIENCY FOR ETHERNET LINKS

Slingshot implements standard Ethernet and negotiates enhanced features using LLDP

- **HPC-E: Co-designed by Cray and Broadcom, first released in Slingshot-200**
  - Optimized frames and headers for bandwidth efficiency
    - Removed Ethernet’s 64B minimum frame size and enabled 32B frames
    - Removed inter-packet gap
    - Reduced preamble
    - Optimized “native IP” headers without an L2 header
  - Enhanced link layer reliability features
    - Low-latency FEC (see 25Gbit Ethernet Consortium)
    - Link level retry (LLR) to tolerate transient errors (used in Cray systems for some time)
    - Lane degrade to tolerate hard failures (used in Cray systems for some time)
- **UEC 1.0 includes:**
  - Link level retry
  - Credit based flow control (more efficient than PAUSE)
  - LLDP negotiation of enhanced features
- **Now, UEC is exploring bandwidth efficiency improvements**



A SCALABLE RELIABILITY LAYER

STARTING FROM THE SLINGSHOT "CONNECTIONLESS" RELIABILITY LAYER

Leveraging the Connectionless Libfabric API

- **What does that mean?**
  - No persistent, process to process connection
  - No user visible connection establishment or connection state
- **What it does not mean:**
  - It does not mean that there are never any sequence numbers
  - It does not mean that we cannot implement ordering, reliability, etc.

Packet Delivery Contexts for Reliability

- **Established between a pair of NICs on demand when a NIC has data to send**
  - Pipelined establishment means they stand up "instantly"
  - Can be shared by multiple processes
  - More than one allowed
- **Removed when the pair of NICs is no longer communicating**
  - One round-trip time to remove them
  - Normally removed due to:
    - Idle
    - Connection pressure at the sender
    - Connection pressure at the target

FOUR MODES OF RELIABILITY

Reliability Modes Brought from Slingshot

- **Reliable, Ordered Delivery (ROD)**
  - Guaranteed delivery of every packet exactly once in the order it was sent
  - Great for MPI header matching
- **Reliable, Unordered Delivery for Idempotent operations (RUDI)**
  - Guaranteed delivery of every packet at least once
  - Great for scalability – no Packet Delivery Context state required
  - Limited semantics: no matching or target side completions

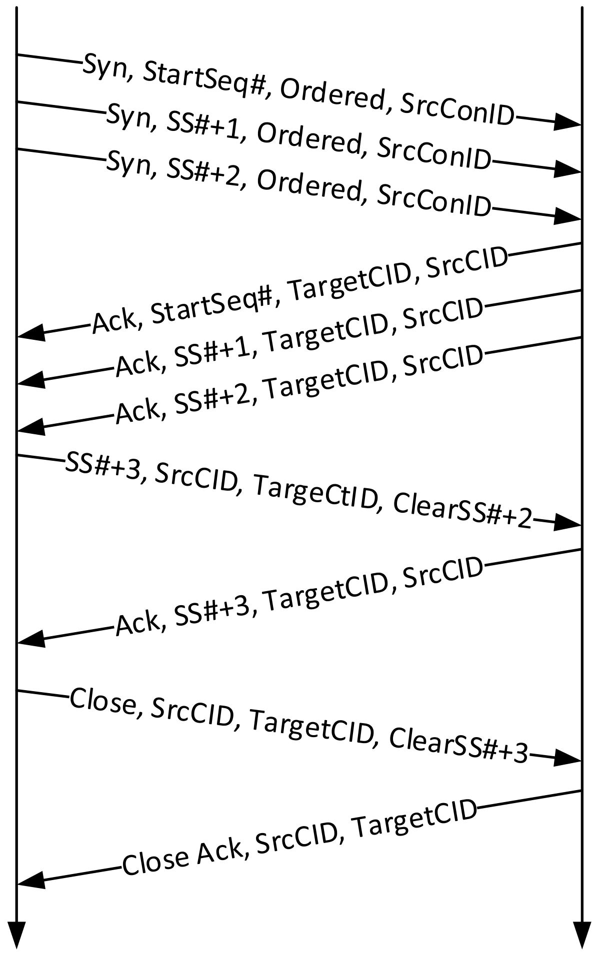
New Reliability Modes

- **Reliable, Unordered Delivery (RUD)**
  - Guaranteed delivery of every packet exactly once
  - Not great for MPI header matching
  - Required for Ultra Ethernet congestion management
  - Many programming models work fine with no ordering (more on this later)
- **Unreliable, Unordered Deliver (UUD)**
  - Enable user who need a UDP-like service to program the entire application in libfabric

RELIABILITY USING PACKET DELIVERY CONTEXTS (PDC)

How Connectionless Reliability Works

- **Connection starts when there is data ready to send**
  - Packets are marked with a SYN bit until the first response is received
  - Response contains a target connection ID to use in subsequent packets
  - This is a “0 RTT Startup”
- **All acknowledgements must be received before a connection can be closed**
  - This means that a Close can clear all state at the target
  - Close operations are acknowledged and will be retried if the acknowledgement is lost
- **Targets can request an initiator close a connection**
  - For example, if the target connections are overloaded

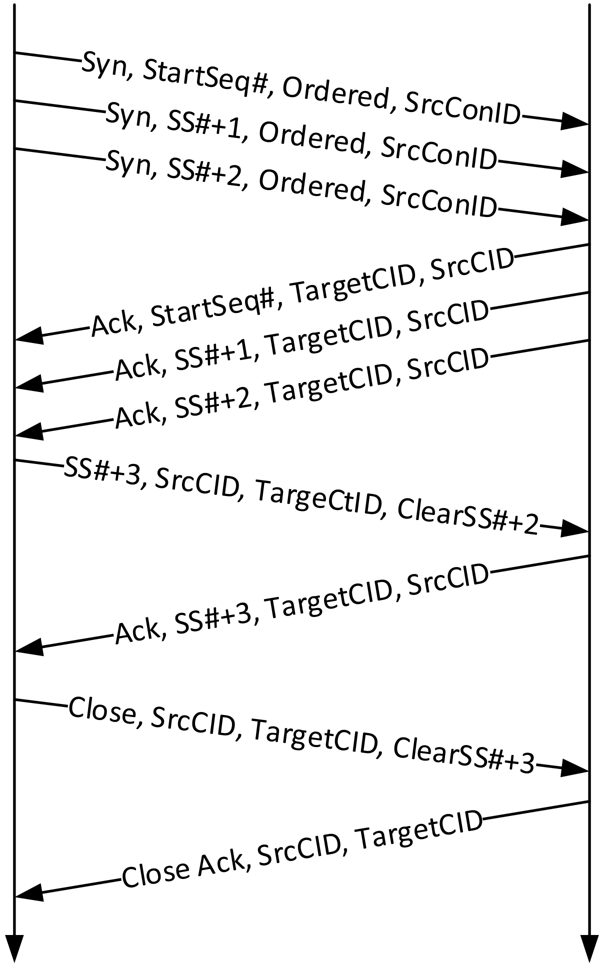


A LOOK AT THE RELIABILITY HEADER

- **What is Clear?**
  - A cumulative acknowledgement of acknowledgements
    - Some acknowledgements carry the result of the operation (Example: fetching atomic operations)
    - Must let the peer know those have been received
  - Significant bandwidth efficiency improvement for reliable responses
- **What are those PDC IDs?**
  - Identifiers of the source and destination PDC state
  - Destination PDC ID is not known until after a round-trip
    - DPDCID added to Ultra Ethernet to improve the scalability of implementations
    - Reuse that field to indicate the offset of packet from the starting PSN
- **Flags: Retransmit, Ack Request, and SYN**

PDS Request Header (ROD, RUD)

Header Start Byte	0				31
	byte 0		byte 1	byte 2	byte 3
0	type	next_hdr	flags	clear_psn_offset	
	psn				
8	spdcid			dpdcid or {pdc_info, psn_offset}	



A BRIEF DETOUR: CONGESTION MANAGEMENT

Slingshot Congestion Management

- **Flow-based congestion management in the switches**
  - Distinguishes congesting flows from victims
  - Distinguishes endpoint and mid-fabric congestion
  - Applies back-pressure to congesting flows
- **Acks use dedicated hardware resources**
  - Sub-microsecond reaction to changing load
- **Congesting flows are held at ingress**
  - They do not consume network buffer space
- **Interoperates with adaptive routing**
- **Integrates with third party NICs and switches**
  - Fine-grain Flow Control (FGFC) returns identifiers and credit for congesting flows

Ultra Ethernet Congestion Management

- **Designed for lossy network environments using:**
  - ECN Marking
  - Round-trip time
  - Packet Trimming
- **Avoids mid-fabric congestion with adaptive packet spraying**
  - Packets are unordered and randomly distributed using entropy in packet
  - "Bad paths" (e.g., ECN marked) are not reused "soon"
  - When many paths are bad, taper the injection rate
- **This is the motivation for RUD**
  - Depends on "most" traffic being unordered

A TRANSPORT TO SUPPORT LIBFABRIC

CROSSING THE RUBICON: A CONNECTIONLESS ARCHITECTURE

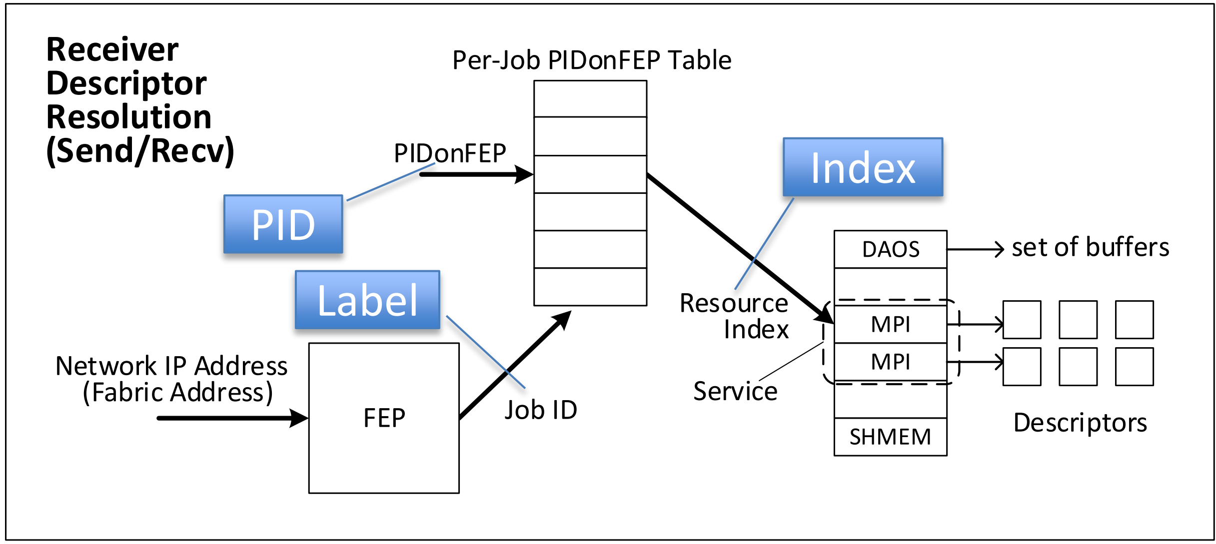
Focusing on a Scalability First approach to network transports

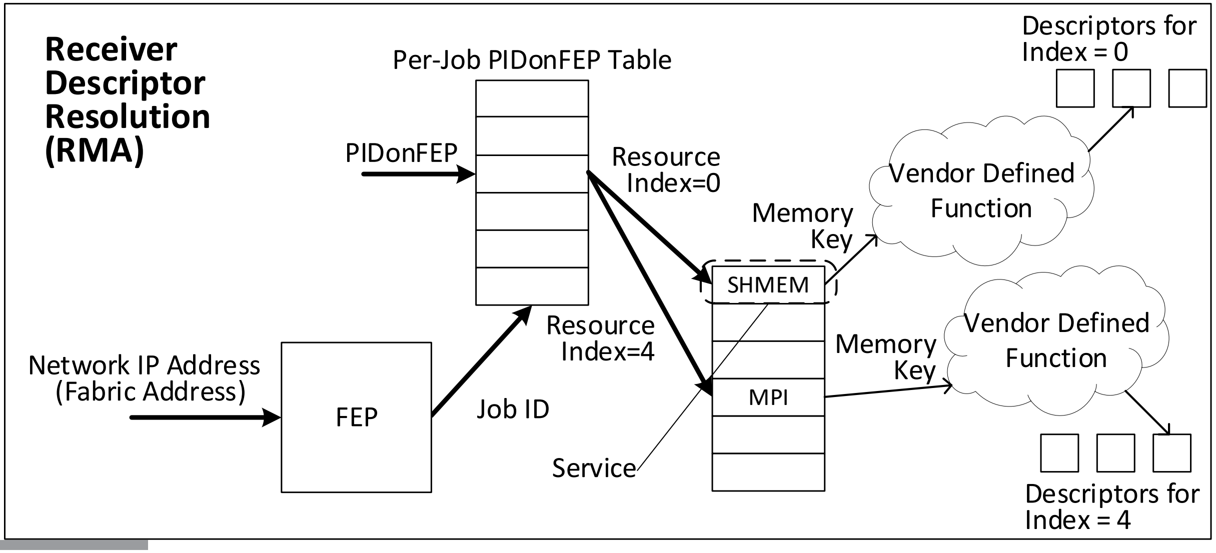
- **Must *not* use a connection to:**
  - Find a buffer
  - Authorize access to that buffer
  - Describe the failure model
  - Recover from buffer exhaustion (e.g., receiver not ready)
- **Must enable scalable addressing**
  - Eliminate startup costs associated with collecting and redistributing endpoint addresses
  - Minimize required per-peer address storage
- **Must enable interoperability between providers**
  - Define all the bits on the wire for all of the transaction types
- **Must enable multiple user level libraries concurrently**
  - CCL, SHMEM, MPI – all linked in the same application
- **Addresses are hierarchical:**
  - A fabric address (IP address)
  - A Job Identifier (JobID)
  - A process identifier (PIDonFEP) relative to that JobID
  - A resource identifier within that process (Resource Index) to select the endpoint
  - A memory key (RMA) or match bits (tagged messaging) within that resource
- **Authorization is based on the sender's identity**
  - Job ID
  - Encrypted and authenticated
- **Advanced transport features have wire level header definitions**
  - Tag matching
  - Rendezvous
  - Deferrable Send
  - Atomic operations

A SCALABLE ADDRESSING SOLUTION

Enabling the communication paradigms that parallel applications actually use

- **What does a user expect?**
  - Launch a parallel application: many processes on many nodes
  - Initialize a communications library (CCL, MPI, SHMEM)
  - Send a message to a peer – any peer – in the application
  - An application may use more than one communication library
- **An example Libfabric over UET flow:**
  - Launch a 256 rank application that is given JobID 42 on 16 nodes:
    - 16 fabric addresses: 10.0.0.16 to 10.0.0.31
    - 16 processes per node: PIDonFEP=0 to PIDonFEP=15 in JobID42
    - Map ranks to addresses:
      - Rank 0: 10.0.0.16, PIDonFEP=0
      - Rank 47: 10.0.0.18, PIDonFEP=15
      - Rank 48: 10.0.0.19, PIDonFEP=0
  - Initialize your favorite CCL, MPI, and SHMEM
    - Distribute rank mapping with job launch
    - Open a libfabric endpoint for each using a service string (returns well-known resource index set – e.g., Resource Index=MPI)





SEMANTIC LAYER FEATURES

Designed for an Out-of-Order Network

- **Includes common HPC functionality**
  - Send/recv (including tagged send/recv)
  - RMA (read/write)
  - Atomic operations (including fetching atomics)
  - Rendezvous operations

SOM=0

byte 0	byte 1	byte 2	byte 3
rsvd, opcode	ver, dc, ie, rel, hd, eom, som	message_id	
ri_generation	JobID		
rsvd, PIDonFEP		rsvd, resource_index	
buffer_offset			
buffer_offset			
initiator			
memory_key / match_bits			
memory_key / match_bits			
rsvd		payload_length	
message_offset			
request_length			

SOM=1

byte 0	byte 1	byte 2	byte 3
rsvd, opcode	ver, dc, ie, rel, hd, eom, som	message_id	
ri_generation	JobID		
rsvd, PIDonFEP		rsvd, resource_index	
buffer_offset			
buffer_offset			
initiator			
memory_key / match_bits			
memory_key / match_bits			
header_data			
header_data			
request_length			

- **Enables out-of-order processing**
  - Every packet includes full addressing information
  - Packet includes which packet within a message it is
  - Enables processing packets in any order with direct placement at the right location in user memory

BEYOND THE BASIC HEADER

Optimizations and Extensions in the HPC Profile

- **Includes two header size optimizations**
  - Smallest: designed for smallest RMA transfers
    - Limited to one packet
    - Limited addressing
    - RMA only
  - Medium: used for single packet Matching or RMA
    - Limited to one packet
    - Complete addressing semantics
    - Supports tag matching
- **Support for atomic operations**
  - Uses an extension header (bottom)
  - Can be supported with any other header type

<table>
<tr><td colspan="8">byte 0</td><td colspan="8">byte 1</td><td colspan="8">byte 2</td><td colspan="8">byte 3</td></tr>
<tr><td colspan="2">rsvd</td><td colspan="6">opcode</td><td colspan="2">ver</td><td>dc</td><td>ie</td><td>rel</td><td>rsvd</td><td>eom</td><td>som</td><td colspan="2">rsvd</td><td colspan="14">request_length</td></tr>
<tr><td colspan="8">ri_generation</td><td colspan="24">JobID</td></tr>
<tr><td colspan="4">rsvd</td><td colspan="12">PIDonFEP</td><td colspan="4">rsvd</td><td colspan="12">resource_index</td></tr>
<tr><td colspan="32">buffer_offset</td></tr>
<tr><td colspan="32">buffer_offset</td></tr>
</table>

<table>
<tr><td colspan="8">byte 0</td><td colspan="8">byte 1</td><td colspan="8">byte 2</td><td colspan="8">byte 3</td></tr>
<tr><td colspan="2">rsvd</td><td colspan="6">opcode</td><td colspan="2">ver</td><td>dc</td><td>ie</td><td>rel</td><td>hd</td><td>eom</td><td>som</td><td colspan="2">rsvd</td><td colspan="14">request_length</td></tr>
<tr><td colspan="8">ri_generation</td><td colspan="24">JobID</td></tr>
<tr><td colspan="4">rsvd</td><td colspan="12">PIDonFEP</td><td colspan="4">rsvd</td><td colspan="12">resource_index</td></tr>
<tr><td colspan="32">header_data / buffer_offset</td></tr>
<tr><td colspan="32">header_data / buffer_offset</td></tr>
<tr><td colspan="32">initiator</td></tr>
<tr><td colspan="32">match_bits / memory_key</td></tr>
<tr><td colspan="32">match_bits / memory_key</td></tr>
</table>

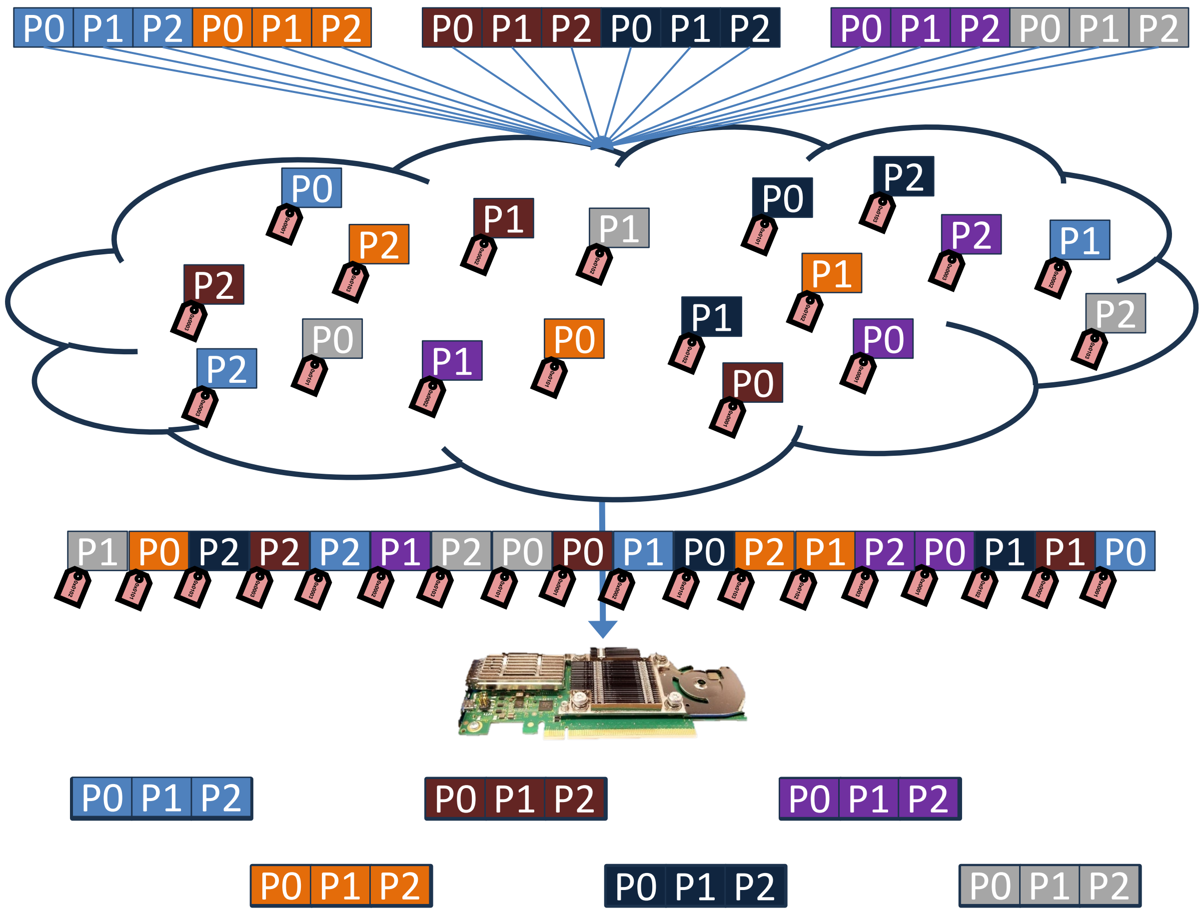
byte 0	byte 1	byte 2	byte 3
atomic_opcode	atomic_datatype	semantic_control	rsvd

MPI, CCL, AND UNORDERED NETWORKS

HARNESSING AN UNORDERED NETWORK

How does that… work?

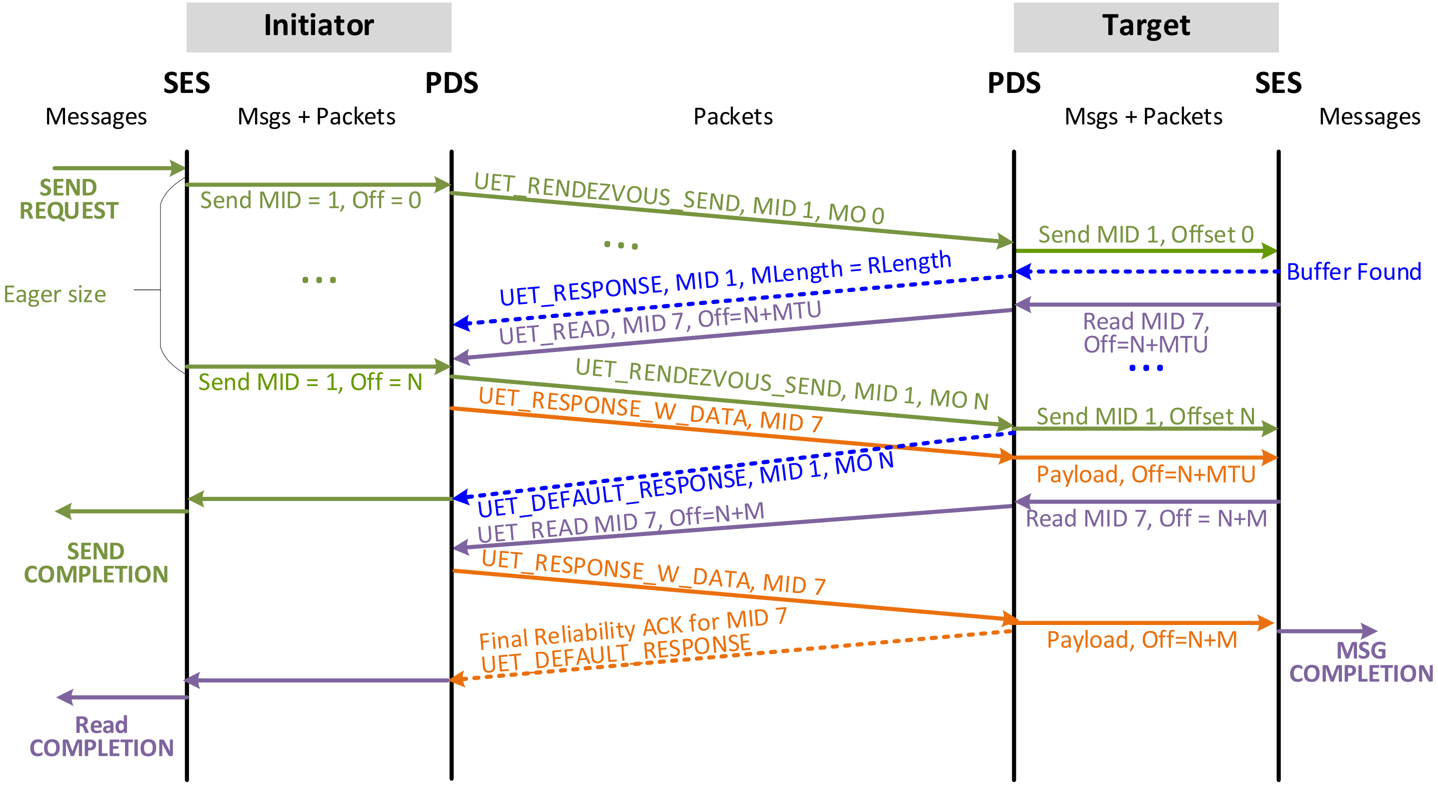
- **The protocol accepts out-of-order packets – and delivers them straight to user memory**
  - Separated semantics from reliability
    - Every message succeeds or fails on its own
    - Failures do not propagate from one message to the next
  - Works for RMA and messaging
- **fi_write(): the libfabric RMA Write**
  - Every packet has a destination address
- **fi_tsend(): tagged messaging**
  - A semantic solution from HPC with a simplified implementation for AI
  - Messaging library (NCCL/RCCL) can put a message number in the tag
  - Every packet has message number and location in message



WHAT ABOUT MPI?

MPI matching semantics are substantially harder

- **MPI messages must be matched "in order"**
  - Generally hard to avoid
  - Message "envelope" needs an ordered protocol (e.g., ROD)
  - But, that is not great for load balancing and congestion control
- **Introducing a standard wire protocol for rendezvous**
  - Rendezvous sequences are typical for MPI, but non-standard
  - Requests can use ordering
  - Most payload can use unordered PDCs



ENABLING RENDEZVOUS OFFLOAD

Standardizing the wire protocol for a Rendezvous Request

- **Requests extended with a rendezvous request header**
  - Includes all information needed to issue a read
  - Read offset points to the start of the message at the initiator
- **After matching, target issues read to "pull" the remaining data**
  - Length: may be as long as request_length
  - Offset: may start at read_offset or at any location up to read_offset + eager_length
  - Should use RUD to issue read

byte 0	byte 1	byte 2	byte 3	
rsvd opcode	ver dc ie rel hd eom som	message_id		Standard Request Header
ri_generation	JobID			
rsvd	PIDonFEP	rsvd	resource_index	
buffer_offset				
buffer_offset				
initiator				
memory_key / match_bits				
memory_key / match_bits				
header_data				
header_data				
request_length				
eager_length				Rendezvous Request Header
read_ri_generation	read_PIDonFEP		read_resource_index	
read_offset				
read_offset				
read_memory_key				
read_memory_key				

SUMMARY

- **Slingshot proved that Ethernet can meet the most challenging workloads in AI & HPC**
  - Proven at scale on all three of the US Exascale systems, HPE Cray supercomputers and now scale-out AI systems
- **The Ultra Ethernet 1.0 standard builds on that foundation to create a modern RDMA interface over Ethernet**
  - Extended with recent advances in congestion management from Cloud datacenter experts
- **Ultra Ethernet is the first standard transport designed to support libfabric**
  - Enabling native multi-vendor interoperability with libfabric for the first time
  - Breaks away from legacy APIs that inhibit scalability and innovation
- **And more: the next iteration of the UEC specification is under development**
  - HPC oriented link optimizations like LLR and CBFC create the right framework to support scale-up networking
  - Packet efficiency improvements are being considered to reduce overhead for small transfers

2025 OFA Webinar Series

THANK YOU

Keith D. Underwood

HPE