New Communication Standard Rationale

November 2018
Khronos Exploratory Group Process

1. Proposal to create a new industry standard
2. Exploratory Group determines industry interest and creates a detailed Statement of Work (SOW)
   - High-level discussions: industry need, requirements, design directions, identify possible future design contributions
3. Khronos Board votes to establish a Working Group to execute the SOW
   - Will there be Adoption of the proposed standard? Is Khronos the appropriate Standards Body to undertake and promote the work?
4. Working Group Meetings
   - Multiple design contributions welcome
5. Invite industry feedback and input. Open to multiple possible future design contributions

We Are Here!
What Problem do we need to Solve?

The Problem

The Solution
Why we need a New Standard?

To Facilitate Application Performance
• Very low overhead / low latency / deterministic communications
• Dynamic & Fault tolerant connections

To Reduce Software Complexity
• Very simple API - minimal learning curve
• One solution for heterogenous architectures

To Address Portability and Maintainability
• Interconnect agnostic
• Extendable to custom interconnects / new technology insertion

There doesn’t seem to be any open standard that addresses all three of these points.
Why Not MPI, MCAMI Or Something Else?

- Current standards do not fully address all the problems for heterogeneous architectures
- Technology has moved on considerably since many of these were established

<table>
<thead>
<tr>
<th>Adoption Cost</th>
<th>MPI</th>
<th>MCAPI</th>
<th>Sockets / ZeroMQ</th>
<th>OFI (libfabric / verbs / Network Direct)</th>
<th>New API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>Heavyweight standard. 300+ APIs, 800+ page spec Process based</td>
<td>Low level API, with several different communication concepts</td>
<td>-20 APIs but underlying complexity</td>
<td>Low level API, ~130 APIs, significant complexity</td>
<td>Simple API All localities, no implicit features</td>
</tr>
<tr>
<td>Performance</td>
<td>No one-way, zero-copy, two-sided transfers</td>
<td>No one-way, zero-copy, two-sided transfers</td>
<td>TCP/IP Stack, intermediate buffering</td>
<td>No OS involved. Control down to the HW level</td>
<td>Thin wrapper over Interconnect: e.g. RDMA</td>
</tr>
<tr>
<td>Fault Tolerant</td>
<td>Not designed for fault tolerance</td>
<td>Timeouts, disconnect detection, dynamic paths</td>
<td>Timeouts, disconnect detection, dynamic paths</td>
<td>Timeouts, disconnect detection, dynamic paths</td>
<td>Timeouts, disconnect detection, dynamic paths</td>
</tr>
<tr>
<td>Portability</td>
<td>Some complexities for lighter weight RTOS’s, not for distributed threads</td>
<td>Generally portable</td>
<td>Generally portable, not for distributed threads</td>
<td>Not easily ported</td>
<td>Easily portable</td>
</tr>
</tbody>
</table>

These other point-to-point APIs still have their place for certain applications / architectures
Why not Sockets or a Derivative?

- **Adoption Cost**
  - Not intuitive to newcomers
  - About 20 functions, with variations for all OSes
  - Hello world is about 100 LOC (in C) with full error checking and OS portability
  - Support two variations of sending and receiving

- **Performance**
  - Heavy use of OS services
  - Created before RDMA existed, so when wrapped on RDMA, only get 50% of RDMA performance
  - Not suited for low latency or determinism due to system calls

- **High Application Availability**
  - Well suite for HAA with it’s explicit timeouts and disconnect detection

- **Portability**
  - Each OS has slight but un-intuitive variations
  - BSD and Posix flavors
  - Lacks inter-thread communication; difficult to transition from processes to threads
Why not MPI?

• Adoption Cost
  - Significant learning curve, development time, and maintenance time
  - Over 300 functions and the specification is 800 pages
  - 5 variations of APIs to transfer data
  - Process based, does not support inter-thread communication
  - Inter-application communication is challenging; other methods simpler
  - Lacks un-reliable communication; not suitable for stream IO devices

• Performance
  - Some high performance implementations, but API design inhibits real-time determinism (MPI/RT standard never evolved)
  - Hides many details (buffering, synchronization, interconnects used, etc.); makes it difficult for experts to fine tune
  - Inherent need for all-to-all communication topology - resource heavy

• High Application Availability
  - No explicit user definable event driven timeouts
  - No explicit disconnect detection

• Portability
  - MPI is portable to most modern Oses. Does not fit well with some RTOSes and bare / close to metal paradigms
  - Lacks inter-thread communication; difficult to transition from processes to threads
  - Lacks unreliable datagram communication, requiring a different communication interface
Why not OFI libfabric/Verbs/Network Direct?

• Adoption cost
  - Significant learning curve, development time, and maintenance time
  - About 100 functions (more for OFI)
  - Hello world is about 1000 LOC (in C) with full error checking
  - Support multiple variations of sending and receiving: one sided, two sided, push, pull, etc.

• Performance
  - Top notch with latency, throughput, and determinism since little to no system calls involved

• High Application Availability
  - Well suited for HAA with it’s explicit timeouts and disconnect detection

• Portability
  - Super computer / HPC focused, not well suited for embedded heterogeneous
  - Lacks inter-thread communication; difficult to transition from processes to threads
Why not MCAPI?

• **Adoption Cost**
  - Similar learning curve to sockets
  - About 40 functions
  - Hello world is about 100 LOC (in C) with full error checking
  - Support multiple variations of sending and receiving: packet, scaler, message, connectionless

• **Performance**
  - Performance equal to underlying interconnect, which some caveats for RDMA
  - Not suitable to determinism with RDMA due to dynamic memory pinning and address coordination

• **High Application Availability**
  - Well suited for HAA with it’s explicit timeouts and disconnect detection

• **Portability**
  - Well suited for portability
Why not OpenMP?

• Not The Same Paradigm
  - Not meant to be a communication interface
  - Designed to do symmetric processing across cores of a single processor in a single process
  - Is a compiler technology instead of an API

• New HetComm Standard Won't Compete
  - OpenMP does not do inter-processor/process communication
  - OpenMP can sit side by side with inter-processor/process comms (e.g. MPI) instead of replacing them
  - OpenMP is not meant to do multi-threaded processing where each thread uses an independent algorithm
Why not HSA/HSail?

• Not The Same Paradigm
  - Is a unified memory architecture over a set of heterogeneous components (CPUs, GPUs, DSPs, etc.)
  - All components accessible via a single memory space
  - Does not contain an explicit communication interface

• New HetComm Standard Wont Compete
  - Customers using HSA don’t need comms
  - Customers not using HSA still need comms
What do we need in a New Standard?

- Simple API
- Built for performance
- Built for fault tolerance
- Easily extendable:
  - Wrappers for collective functions
  - Interconnect support
- Different language support (TBD)
  - C
  - C++ bindings
  - Python bindings
Who Will be the Potential Implementers?

- **Hardware vendors:**
  - COTS board vendors: Abaco, Mercury, Kontron, Curtiss-Wright, XE, etc.
  - Chip vendors: Intel, AMD, ARM, Xilinx, Qualcomm, Samsung, Ti, Analog Devices, etc.

- **Embedded software vendors:**
  - Tool suite vendors: National Instruments, MathWorks, Mentor graphics, etc.
  - OS vendors: Wind River, Red Hat, Green Hills, Microsoft, etc.

- **Research & Academia:** OSU, UNH, etc.

Who Will be the Potential Users?

- **Mil/Aero primes:** Boeing, Raytheon, Lockheed Martin, Northrop Grumman etc.
- **Autonomous vehicle / Robot developers:** BMW, Audi, GM, Continental, Bosch, NVIDIA, Waymo etc.
- **IoT, Industrial & Medical device vendors:** Siemens, GE, Philips, Hitachi, etc.
- **Research & Academia:** MIT Lincoln Lab, Oak Ridge Nation Laboratory, etc.

Note: The lists of potential implementors and users are examples only. They do not represent any current engagement or commitment.
Links For Further Discussion

- Exploratory Website
  
  https://www.khronos.org/exploratory/heterogeneous-communication

  - This website has links to:
    - A survey to gather feedback on your need for a new communication standard
    - Proposals for the new standard