**OFI WG telecon – 05/21/2019**

**Agenda:**

1. Opens, Agenda bashing
2. Status for 1.7.2 release, preparation for 1.8.0 release
3. Complete discussion of Intel’s Collectives proposal (attached).
	1. At the last meeting we did not get to Reduce-Scatter and All-Gather.
4. Updated Cray proposal for Traffic Classification (attached).  See slides 10, 11

**Opens**

-none-

**Release Scheduling**

Start RC for 1.7.2 should be this week or next, don’t want to overlap with 1.8.

Plan to start 1.8.0 in early June. Pulls in EFA provider.

Gen-z is working on a provider, targeted at release 1.9, Q4. Driven by the Pathforward contract which requires collateral to be pushed back into the community.

There are two aspects to Gen-Z which it isn’t clear how they should be exposed. E.g. mapping of remote memory on nodes that are memory only, (no intelligence on the node) and mapping memory between processes across the fabric. This would likely ride on top of the existing key-based mechanisms.

**Discussion – Continue Collectives Discussion (5/21/19)**

Most collectives are mapping onto fetch atomics with new opcodes

Reduce-Scatter

All-Gather – may have similar restrictions to those being considered for all-to-all. Based on fi\_fetch\_atomic with an FI\_ALLGATHER op-code.

Completions – all are expected to be asynchronous w/ notifications via normal CQ events or counter increment.

Ordering – Completion ordering is unchanged. Execution ordering is based on the order of submission of collectives with the same atomic call, same operation, and same data type. Otherwise, there are no ordering guarantees between different collective operations.

Currently working through MAN page descriptions and header file updates. Hoping to complete within the next two weeks.

Clarified that coll\_addr is the name of the type used to refer to the results of the fi\_av\_set and fi\_join operations, which produce the 64 bit address (fi\_addr\_t)

**Discussion – Continue Collectives Discussion (notes from 5/7/19 meeting)**

Two parts:

1. Membership – two possibilities.
	1. proposal is that e.g. MPI would fully manage the collective group in a multicast style
	2. push it down below the libfabric level.

Two step process:

1. identify the peer members via new construct ‘AV Set’, very close to an MPI group.
2. join the collective

For some collective operations, the order of members may matter.

Very similar to a multicast group, but with a few differences such as ordering. There are also a few requirements related to data buffers, which are identified at the time of the operation.

Join operation may involve network traffic, depending on whether the provider implements the join or not and whether the join is local or remote.

1. Local vs remote flag – the join operation could go only to the local node, or it could be distributed across the fabric.
2. Expected to be an asynchronous operation (both local and remote, although local is negotiable).
3. Collective operations
	1. New FI\_COLLECTIVE capability bit
	2. Based on existing atomic interface
	3. Includes new static inline wrappers (for user convenience)
	4. Handles most of the popular collective calls
	5. Examples:
		1. Barrier – fi\_barrier static inline 🡪 fi\_atomic call
		2. Broadcast – fi\_broadcast static inline 🡪 fi\_atomic call mapped to FI\_WRITE/FI\_READ depending on which side the broadcast you are on.
		3. All reduce – maps to fi\_fetch\_atomic
		4. All to All – does not map well to any existing call. Can map to atomic fetch but with some restrictions, i.e. type is replaced by number of bytes and all peers are sending the same amount of data (number of bytes)?
		5. A few more to be discussed (left for next meeting

**Updated Proposal for Traffic Classification**

Modified the definitions and number of “HPC” traffic classes:

1. Focused ‘Bulk Data’ to explicitly be ‘Bulk I/O Data’
2. Added a new Bulk Data class dedicated to IPC, especially for e.g. MPI rendezvous operations. Tentatively named ‘Bulk IPC Data’
3. Added a new ‘Management’ class
4. Decided not to add a class specifically for checkpoint
5. Agreed that setting traffic classes will be an asynchronous operation.

No apparent need for a capabilities mechanism; a provider will need to map whatever class is requested onto its available supported classes.

No current plan to add policing to enforce class usage.

Authentication – how does a provider enforce authentication? By using the existing auth key built into the API today.

Next step - move onto the second part of the proposal, describing how traffic classification is controlled across the API.

Assumption that traffic class is set at the time the endpoint is created. There may be apps that would like to change traffic classification on the fly, but it does not seem worthwhile to incur the impact of providing that degree of control.

Suggestion is to use scalable endpoints.

Two options were described:

1. There exists a set of virtual channels, one per traffic class, with each message tagged for a given class and therefore slotted to the appropriate virtual channel.
2. Or a queue is bound to a set of virtual channels and you’d have multiple queues.

If the former, each individual message has to be tagged with its desired TC. To be able to support either method, you would want to add a capabilities mechanism.

Most restrictive method: every operation posted to a given endpoint is going to use the same TC.

Least restrictive method: Every transfer request could have a different traffic classification.

Consensus is that the transmit context should carry the QoS.

**Next meeting**

Tuesday, June 4, 2019

9:00 – 10:00AM PST

**Recording:**

|  |
| --- |
| **OFIWG every-two-week meeting (2019)-20190521 1603-1** |
| Tuesday, May 21, 2019 |
| 12:03 pm  |  Eastern Daylight Time (New York, GMT-04:00)

|  |
| --- |
| [**Play recording**](https://cisco.webex.com/cisco/lsr.php?RCID=a33b208da38d4c26b66b3e394f3dc13e) (50 min 44 sec) |
| Recording password: tRGzsBZ2 |  |

 |

|  |
| --- |
|   |

**Webex link:** See the OFA central calendar for meeting logistics. <https://openfabrics.org/index.php/ofa-calendar.html>

**OFIWG Download Site:** [www.openfabrics.org/downloads/OFIWG](http://www.openfabrics.org/downloads/OFIWG)

**Github:** <https://github.com/ofiwg/libfabric>

**OFI Software Download Site:** [www.openfabrics.org/downloads/OFI](http://www.openfabrics.org/downloads/OFIWG)