**OFI WG telecon – 04/09/2019**

**Agenda:**

1. Opens, Agenda bashing
2. General timeline for v1.8 release
3. Continue discussion of smart networks (e.g. FPGS, collective offload)

**Opens**

**General Timeline for v1.8 Release**

Target v1.8 sometime second quarter.

Should pick up the EFA provider.

V1.9 sometime in Q3/Q4. Hopes to inject collective offloads, FPGA support, etc at that time.

**Discussion – Smart networks (notes from 3/26/19 meeting)**

Added the smart nic part to the existing presentation.

Proposal – define a new capability called FI\_NETWORK\_FUNC

To use it, bind a new n/w func to an endpoint (or possibly a memory region?).

Function is applied either as a function of a particular n/w operation, or opportunistically.

Currently, no mechanism to allocate or configure remote accelerators, including switch-based.

Allocating remote functions needs to be asynchronous (current proposal for setting up local network functions is synchronous.)

Reviewing the collective stuff from a couple of years ago.

One issue mentioned previously: Peers may not be in sync; there may be a need for some sort of a synchronization function.

**Important:** the description of the low level libfabric functions and how they map onto e.g. MPI collectives should not be looked at as the proposal, it’s simply an attempt to understand a possible mapping.

Discussed ideas about the correct level of abstraction that should be provided across the API. In some cases, the abstraction provided across the API should be such that the provider is able to intuit the intention of the consumer, in the case of offloaded functions.

**Continue the discussion of Smart Networks**

How to deal with multicast operations. Generally, for MPI, all peers are joined at the same time (as opposed to joining onesy-twosy). As inputs to the creation of the communicator, the communicator knows who its peers are. In MPI today, most of the group create/join calls are blocking.

How are collectives signaled across the API? Do we use existing function calls, or create new APIs which are functionally equivalent for readability?

“Gathered receive” – receiver identifies target buffer, sender specifies offset into the buffer. A weird amalgam of tagged and send operations. Possibly useful for MPI\_AlltoAll, (where each peer receives a piece of information from each other peer. Allows the sending peer to describe the offset into the buffer where his contribution is placed).

Continued discussing the correct level of abstraction for collectives across the libfabric API – if the abstraction is too primitive, the provider is unable to intuit the higher-level operation being requested. But if the abstraction is too high, it risks placing too much burden on the provider and/or turning libfabric into MPI.

**Workshop Feedback**

* A desire for better documentation for people developing providers
* A gratifying amount of support for libfabric
* A question about sockets support; we continue to have support for sockets, but looking at replacing the sockets providers, to be replaced by the TCP provider and utility providers. Should improve performance and easier to maintain.

What to call a ‘collective group’? Is it functionally distinct from a multicast group? Probably not, but for readability reasons might want to create a collective flag which is functionally equivalent to the multicast flag.

From a capabilities perspective, fi\_getinfo is focused on discovering local capabilities. How to discover what is available out in the fabric? fi\_getinfo is intended to be synchronous.

Synchonous vs asynchronous – local (endpoint) functions (e.g. memory registration) is normally done today synchronously by applications because it’s simpler. But for accessing remote capabilities, asynchronous may be required.

**Next meeting**

Tuesday, April 9, 2019

9:00 – 10:00AM PST

**Recording:**

**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)