**Agenda**

* **F-2-F agenda**
* Fabric interfaces information structure: struct fi\_info / fi\_getinto()

**OFIWG Download Site:** [www.openfabrics.org](http://www.openfabrics.org) 🡪OFED/OFA Resources 🡪 OpenFabrics Interfaces WG

Requested details for face to face – location and dates.

**F-2-F**

**Date 8/19, 20**

**Location – Hillsboro, OR**

* Cisco: San Jose – happy to provide meeting room, no word yet on catering possibility
* Intel: Hillsboro – provide an offsight meeting facility including lunches, snacks, etc.

Bay area is preferred by some (because they are local to the Bay area)

Who do we lose if we go to Portland?

* Possibly Jeff Becker, depending on travel budgets

Using the Webex online polling tool, the majority select Portland (Hillsboro).

Web meeting, etc will be available at the facility – Rock Creek Country Club. There will be no registration fees, and Intel will provide lunches and snacks; continental breakfast (fruit etc) is a possibility.

AR: Intel to send travel logistics

**Fi\_info details**

* A low-level detailed description of the struct fi\_info was given. This structure, along with the fi\_getinfo() call, is the basis for querying providers for capabilities and specifying the operations needed by the application.
* Chris L : A question was asked regarding how multicast worked. Multicast join/leave operations have been defined, but further details have not been defined or worked out. The libfabric proposal should be considered a framework that is still in development at this point. It is not complete.
* Alan J: Asked about resolving multiple nodes (e.g. hostnames) with fi\_getinfo. The address vector is intended to handle resolving multiple remote addresses and hostnames. Fi\_getinfo is focused on identifying the local domain (NIC) needed to reach a specific address. Fi\_getinfo allows an application to specify a source address. More discussion is needed to determine if fi\_getinfo should expand to support multiple nodes, if a new call should be added, or if the proposed solution is acceptable.
* Chris L: Need to add raw ethernet as a protocol option. This is a trivial addition to the protocol list, but we need to understand if it impacts any other areas and how.
* Chris L: Concerned that complex data structures may limit performance. Fi\_info is not intended for use within the data path. The data transfer interfaces (send, write, recv, etc.) mostly avoid the use of data structures, except for ‘msg’ operations (sendmsg, readmsg, etc.). Initialization does involve more complex structures, as it defines the mechanism by which the provider is informed of the application’s usage model.
* Liran: Questioned why size (fi\_info) and mask fields (other structs) were used for structures, rather than just selecting one option. The mask concept was pulled from the extended verbs format. The use of the size in fi\_info should be re-examined. And we should determine if the mask field is ideal, given that backward compatibility is not a requirement. Structure versioning may be enough.

Looked at fi-info structure

Struct fi\_ep\_attr:

Basic idea: application gets to describe what it wants.

Data\_flow\_cnt – maximum number of flows

Inject\_size –

max\_order\_xxx\_size (three attributes for various ordering combinations)

max\_tag\_value how many tag bits are available for use (?)

Message order:

Defined by a series of flags. These do not govern the placement of data in the buffer, but control the order in which headers/data are processed.

Defined that msg ordering is a requirement for data ordering, but not the inverse.

Ordering is defined per endpoint. If a different ordering is desired, a new EP can be created.

Threading:

FI\_THREAD\_PROGRESS – the application needs to serialize access to objects; e.g. reading an event queue would require serializing access to the endpoint object.

Q: why bind thread safety to progress?

Threading model for control operations is always FI\_THREAD\_SAFE

For future extensibility, how can we add new fields to e.g. fi\_info structure?

Suggestion: if you add new fields, add a new struct fi\_info\_v1.

Could get us into an explosion of versions and backwards compatibility…

There are mechanisms being used where the first struct turns out to be the size of the structure itself, which is given as an input from the user.

Q: have we considered having all data structures opaque to the user, and instead use a series of functions using get and set calls.

A: Concerned that that would explode with a function call for every field. Seems like overkill.

**Agenda for next meeting**

Start going through other interfaces in detail.