**Agenda**

* Use of GitHUB
* review results of F-2-F

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

**GitHub** – Sean

In the past patches have been posted to the mail list, but GitHub provides a mechanism allowing a user to subscribe to the issues list. We prefer to use this facility rather than using the mail list as a means for exchanging patches. Doug Ledford volunteers to write a wiki article detailing how to use the GitHub subscription service. The plan is to use the GitHub issues list as a roadmap for changes that need to be made to the APIs. We have no GitHub experts, so this will be a group learning experience.

**Cont’d’ Review output from the MPI point-to-point pseudo-code group**

Flow control (reviewing from last week’s discussion) – e.g. Reuse of receive buffer resources. What happens when the receiver is overrun with inbound requests. Are these simply reflected upwards to the MPI layer, or are they handled at the API layer? Break these out as two separate discussions:

* reporting overrun conditions upwards, and
* handling overruns autonomously by the provider.

Suspect we need to handle both cases. IB, for example has RNR NAK. Two possibilities are that API has control of a pool of buffers, the other is that the provider provides a mean for backing off the sender (requester).

Some providers may provide a back-off mechanism, but others may not, so the minimum requirement is to provide a mechanism for the application to understand the endpoint’s capabilities. Could be as simple as being “do you support flow control or not?” Hopefully it is not necessary to expose ‘how’ flow control is provided.

But regardless the provider needs to report that overflow has occurred, and the endpoint needs to specify what its behavior is in the case of overrun. There appear to be three options:

* if you have built-in flow control, this never happens ,
* turn the endpoint into error state and flush the queues,
* drop packets but report to the application.

What about flow control from the application layer allowing the application to throttle activity at the provider layer. In InfiniBand terms, the throttling mechanism is the number of work requests posted to a work queue.

To some extent, this is a function of the endpoint type. a reliable connected endpoint will have different behaviors than an unconnected or unreliable endpoint.

Summary: To fully understand this issue, we need one table per endpoint type specifying: 1. what end-to-end flow control mechanisms the provider supports at the transport layer, if any, and 2. the provider’s behavior in the presence of an overflow situation – are packets dropped silently, or dropped with notice or does the provider go to an error state?

This should be on a per-endpoint basis, since different types of endpoints will have different flow control mechanisms and may report overruns differently.

**Agenda for next meeting**

Continue working through the list of issues identified by the pseudo-code working groups at the face-to-face.

**Next regular telecom**

Next meeting: Tuesday, 9/16/14

9am-10am Pacific daylight time