**OFI Data Storage / Data Access Subteam Weekly telecom – 12/01/2015**

**DS/DA Shared Documents:** <http://downloads.openfabrics.org/WorkGroups/ofiwg/>

**Agenda**

* roll call, agenda bashing
* Complete NVM Use Cases discussion
* Begin discussing how these map onto APIs
* Kfabric vs sockets, how are they differentiated?
* Back to the Kernel Maintainer slide deck

**Final Review of NVM Access Methods slides**

* Re byte access vs block access – need to emphasize that a block is a defined size.
* Slide 6 – use of the expression NVDIMM; should this be genericized to NVM? Agreed not to pursue use of the expression PM

**Kfabric vs Sockets – how are they differentiated?**

* Sockets is about a linear stream of bytes (a pipe). Biggest problem is that it assumes a copy on send and receive. Kernel has the latitude to re-package things.
* Storage, on the other hand, is block based, which doesn’t map well onto sock stream semantics
* Sockets require a choice between either markers or reliability – if you want to send a defined number of bytes, you have to insert a header, and a payload. On the recv side, you have to mark where the boundaries are. The alternative is to use UDP (datagrams) which are unreliable. The closet available today is either RDS or SCTP. But RDS is a pretty clumsy way to do RMA operations.
* Bernard – kfabric supports one-sided operations without involvement of the responder side, which sockets cannot do. RMA includes protection mechanisms which sockets does not.
* Atomics don’t exist in sockets.
* Sockets are basically synchronous. When you do a SEND, the local buffer can be re-used immediately, you cannot have outstanding sends (on the requester side) or outstanding receives (on the responder side). A process can be sending to multiple sockets, but on any given socket the consumer has to manage the synchronicity individually.
* A richer set of completion semantics. If a message is posted to a socket, and returns, there is no guarantee that the message was actually sent, in the case where the peer process goes away.
* Summary:
  + Sockets is stream-oriented vs kfabric which is block oriented. Blocks can be conveyed over a stream, but its clumsy and requires things like markets, or the use of an unreliable protocol e.g. UDP
  + Sockets does not include the notion of one-sided operations, and does not have the mechanisms (e.g. memory protection mechanisms) to support one-sided operations
  + Sockets tends to be synchronous – if a given process has more than one operation in flight, the process itself has to ensure synchronization of the operation
  + Kfabric contains a much richer set of completion semantics

**Kernel Maintainer Slide deck**

* Need to address two major questions –
  + What is the difference between kfabric and verbs? and
  + What is the difference between kfabric and sockets?
* Next week we will review this deck to see how far along we are with answering these questions.

**Webex Recording:** [**Play recording**](https://cisco.webex.com/ciscosales/ldr.php?RCID=d41b90773270c8f7abe167b4a2a8971c) (50 min)

**Next regular telecom:**

Next meeting: Tuesday, 12/8/15

8am-9am Pacific daylight time

**NOTE:** We have switched over to using Webex (courtesy of Cisco). The URL for joining meetings is:

[Join WebEx meeting](https://cisco.webex.com/ciscosales/j.php?MTID=m221d8a20185d84b30daa0096aca0f182)

**Join by phone**

+1-866-432-9903 Call-in toll-free number (US/Canada)

+1-408-525-6800 Call-in toll number (US/Canada)

Access code: 201 212 241