OPENFABRICS ALLIANCE

OpenFabrics Interface WG

A brief introduction

10TH ANNUAL INTERNATIONAL DEVELOPER WORKSHOP

Paul Grun – co chair OFI WG
Cray, Inc.

OFI WG – a brief overview and status report

1. Keep everybody on the same page, and
2. An example of a possible model for the OFA going forward (more on this later)

Agenda

1. OFI WG
2. A new framework
3. Guiding principles
4. Current status, process, participation
5. Key issues

OpenFabrics Interface WG

Last August, the OpenFabrics Alliance undertook an effort to review the current paradigm for high performance I/O.

The existing paradigm is the Verbs API running over an RDMA network.

The OFA chartered a new working group, the OpenFabrics Interface Working Group (OFI WG) to:

Develop, test, and distribute:

1. Extensible, open source interfaces aligned with application demands for high-performance fabric services.

2. An extensible, open source framework that provides access to high-performance fabric interfaces and services.

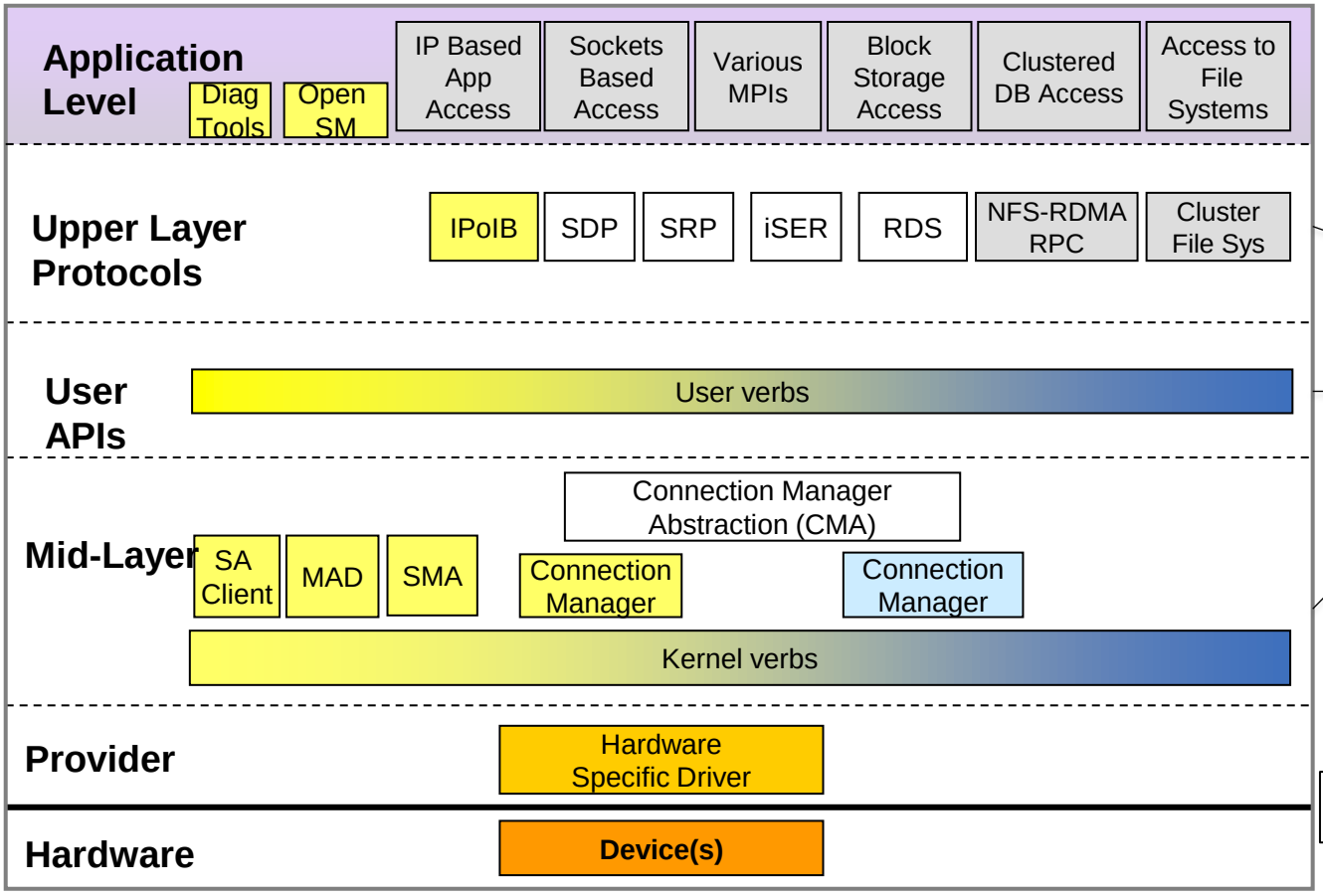
Put simply…

- A series of “API-lets”
  - vs “one API to rule them all”
- A framework to support them

OFI Objectives

- Maximize application I/O (aka network) effectiveness
- Excellent support for a wide range of (classes of) applications
- Minimize interface complexity and overhead
- Make the interface(s) extensible
- Not constrained to a particular wire, fabric or vendor

Verbs-based framework



Support for applications

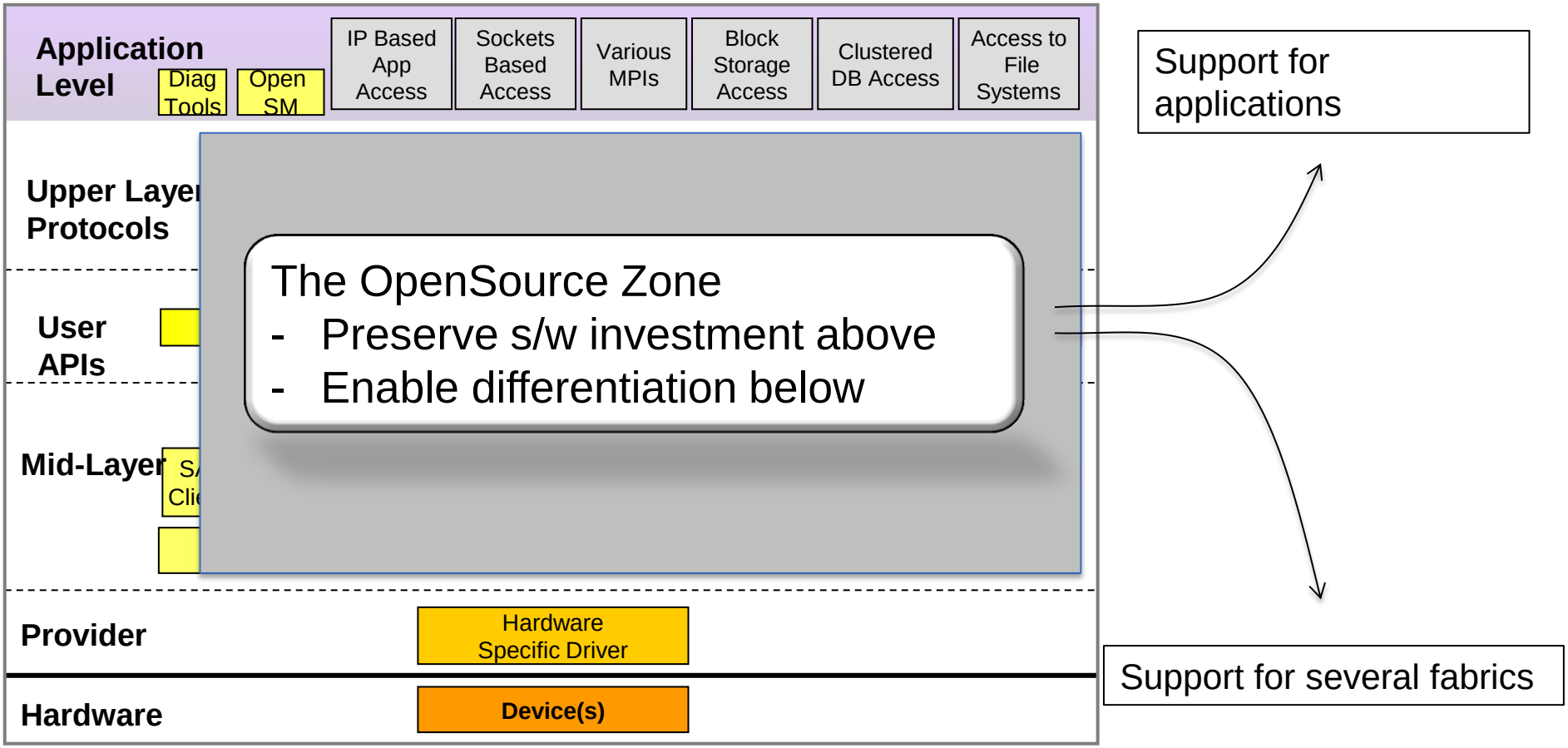
adaptation layer

libibverbs - 60 function calls

a series of kernel services

Support for several fabrics

Verbs-based framework

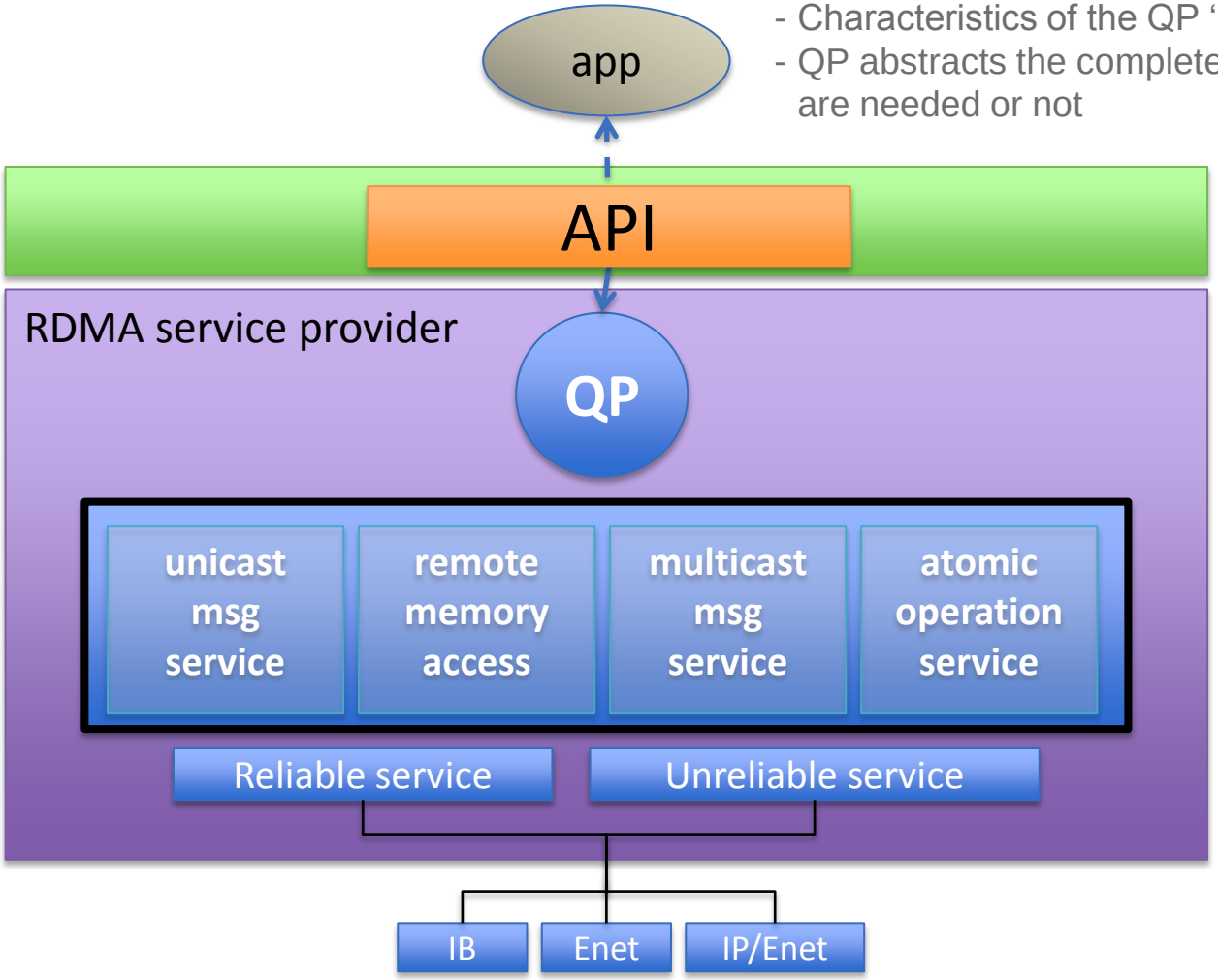


Verbs API

- The Verbs API closely parallels the Verbs semantics defined in the IB Architecture specs
- The IB spec defines a very specific set of I/O services – RC, RD, UC…
- Basic abstraction exported to an application is a queue pair
- A queue pair is configured to provide an operation (send/receive, write/read, atomics…) over one of a set of services (reliable, unreliable…)
- Low level details (e.g. connection management, memory management) are exposed to the application layer (which often doesn't care about such details)

Verbs model

- Characteristics of the QP 'bleed through' to the app
- QP abstracts the complete set of services, whether they are needed or not



one API (verbs)

QP is a h/w construct representing an I/O port

One service provider offering multiple services

three wire protocols

Observations

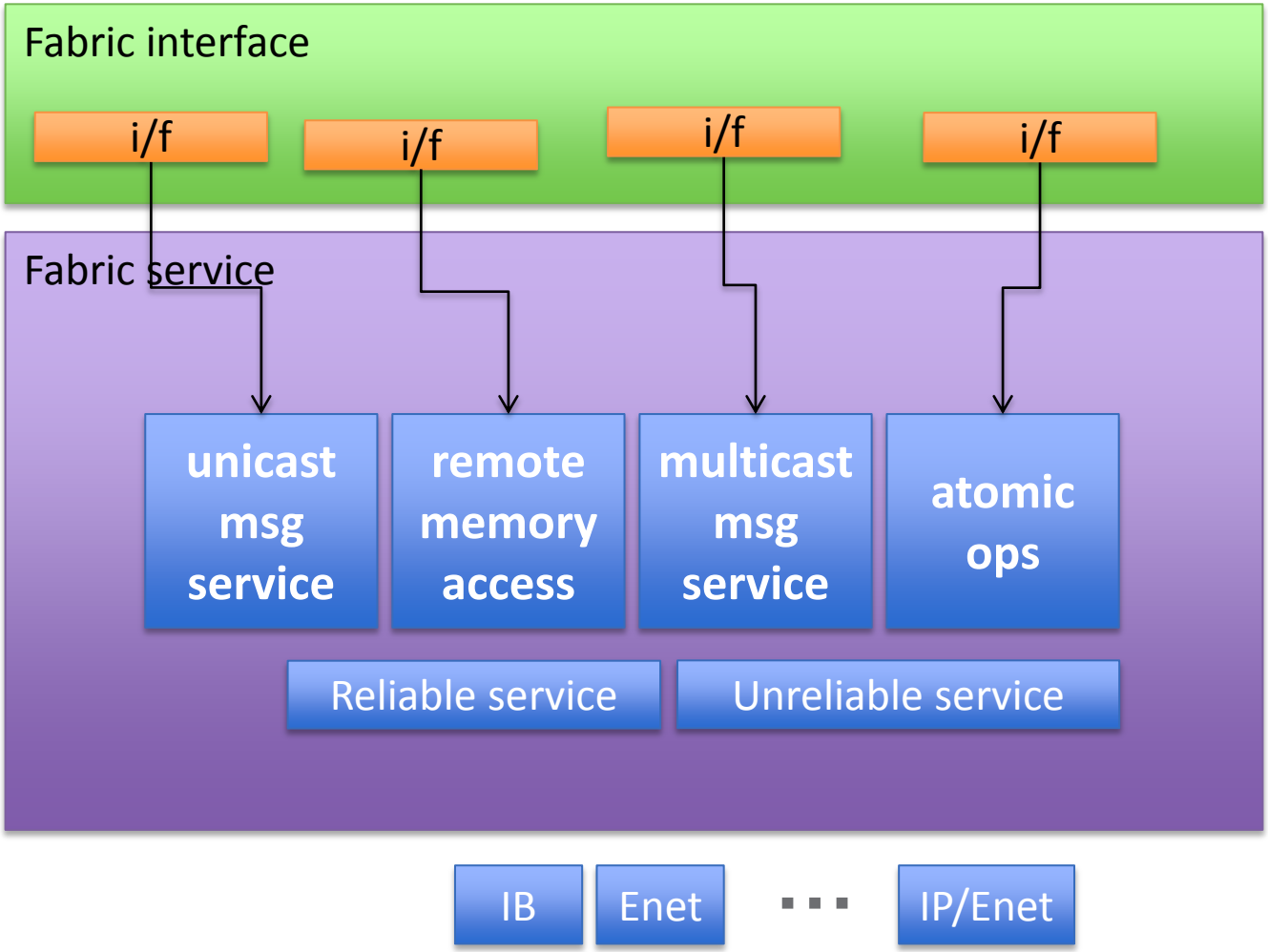
- A single API cannot meet all requirements and still be usable
- A single app would only need a subset of a single API
- Extensions will still be required
  - *There is no correct API!*
- We need more than an updated API – we need an updated *infrastructure*

From Sean Hefty's original proposal

Streamlining the API

- Provide a richer set of services, better tuned to application requirements
- Broaden the number of APIs (“API-lets”), but streamline each by reducing the functions associated with it.
- Each API represents a specific I/O service
- APIs are composable, and can be combined
- Abstract the low level fabric details visible to the application

OFI Model



APIs expose the underlying I/O service

Multiple service providers.

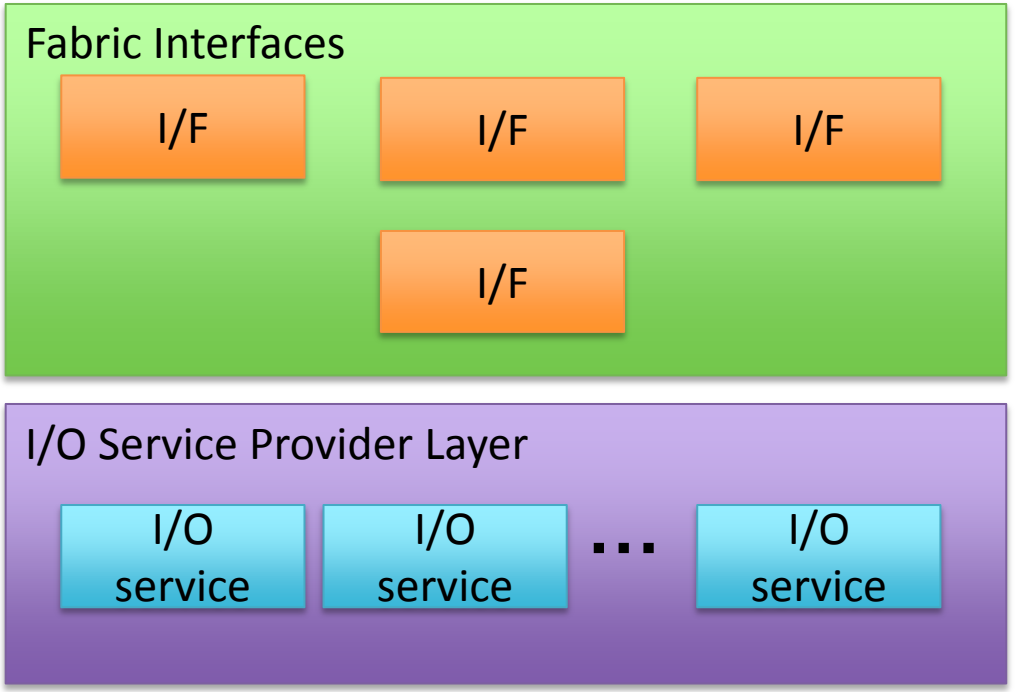
Innovation in I/O service optimization occurs here.

wire protocols

A framework

The framework exports a number of I/O services (e.g. message passing service, large block transfer service, collectives offload service, atomics service…) via a series of defined interfaces.

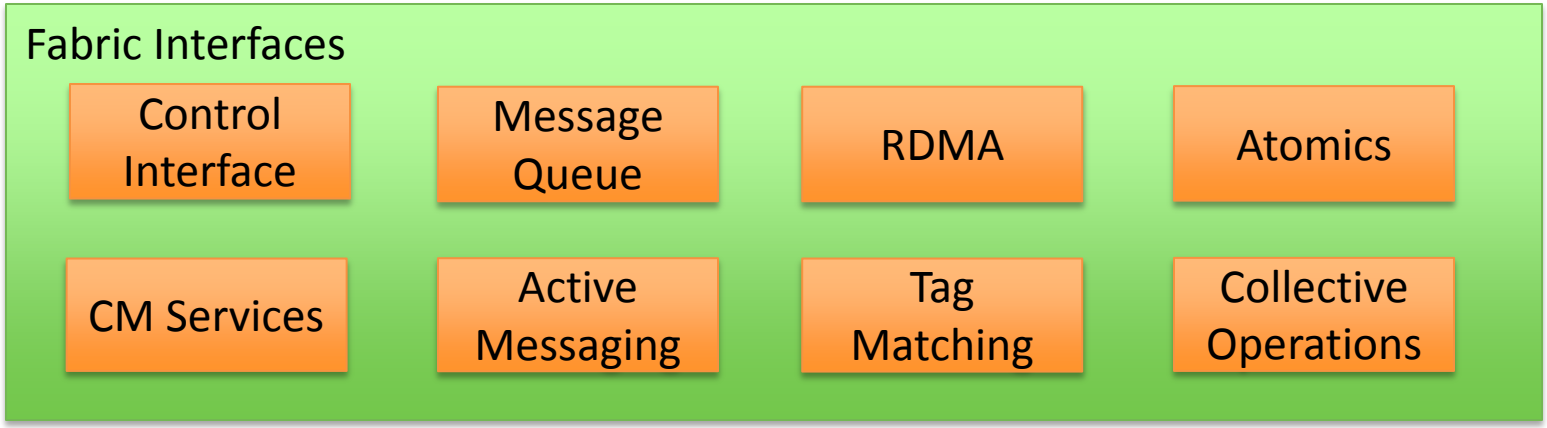
Framework defines multiple interfaces



Implementations are optimized at the provider layer

* Important point! The framework does not define the fabric.

(Scalable) Fabric Interfaces



Q: What is implied by incorporating interface sets under a single framework?

- Objects exist that are usable between the interfaces
  - Isolated interfaces turn the framework into a complex dlopen
- Interfaces are composable
  - May be used together

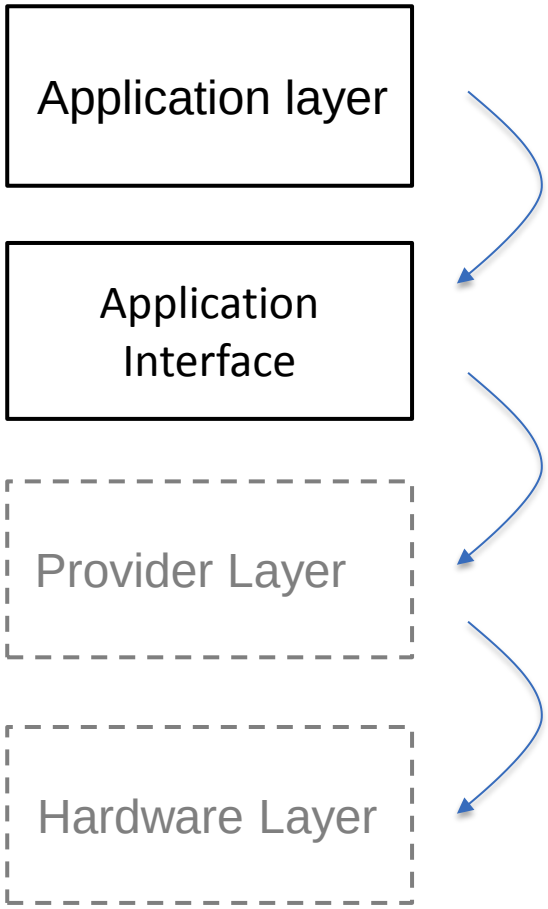
Guiding principles

There are really two –

1. Application-centric I/O

2. Fabric independence

Application as driver



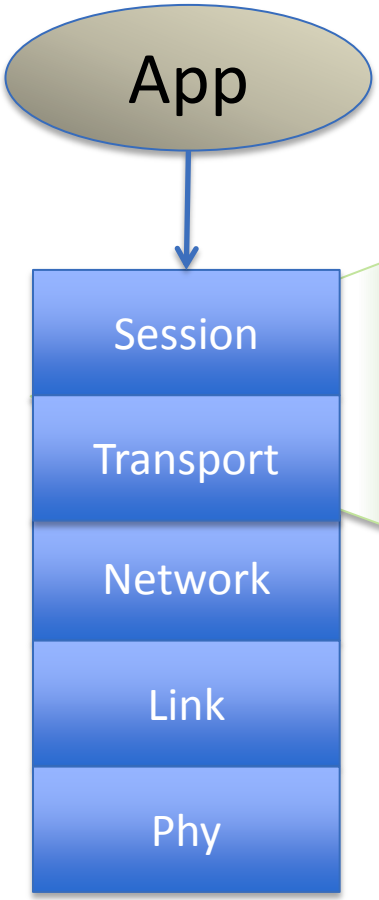
Examine the classes of applications that are important to target users of OFS.

Let the applications drive the appropriate interface definition.

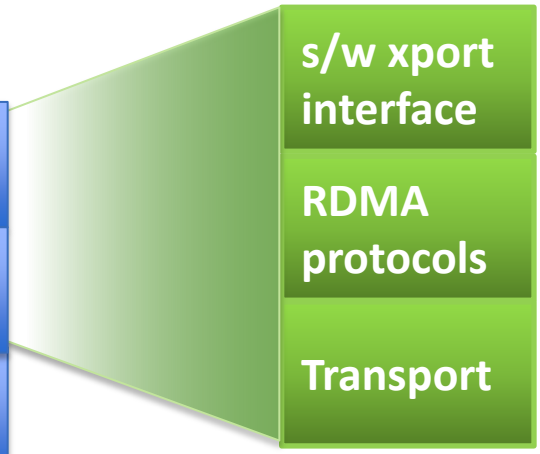
This, in turn, drives the necessary features that the fabric should support.

Different classes of applications may require different types of I/O services

A word about “applications



Let’s agree that an “application” is anything that consumes network services.

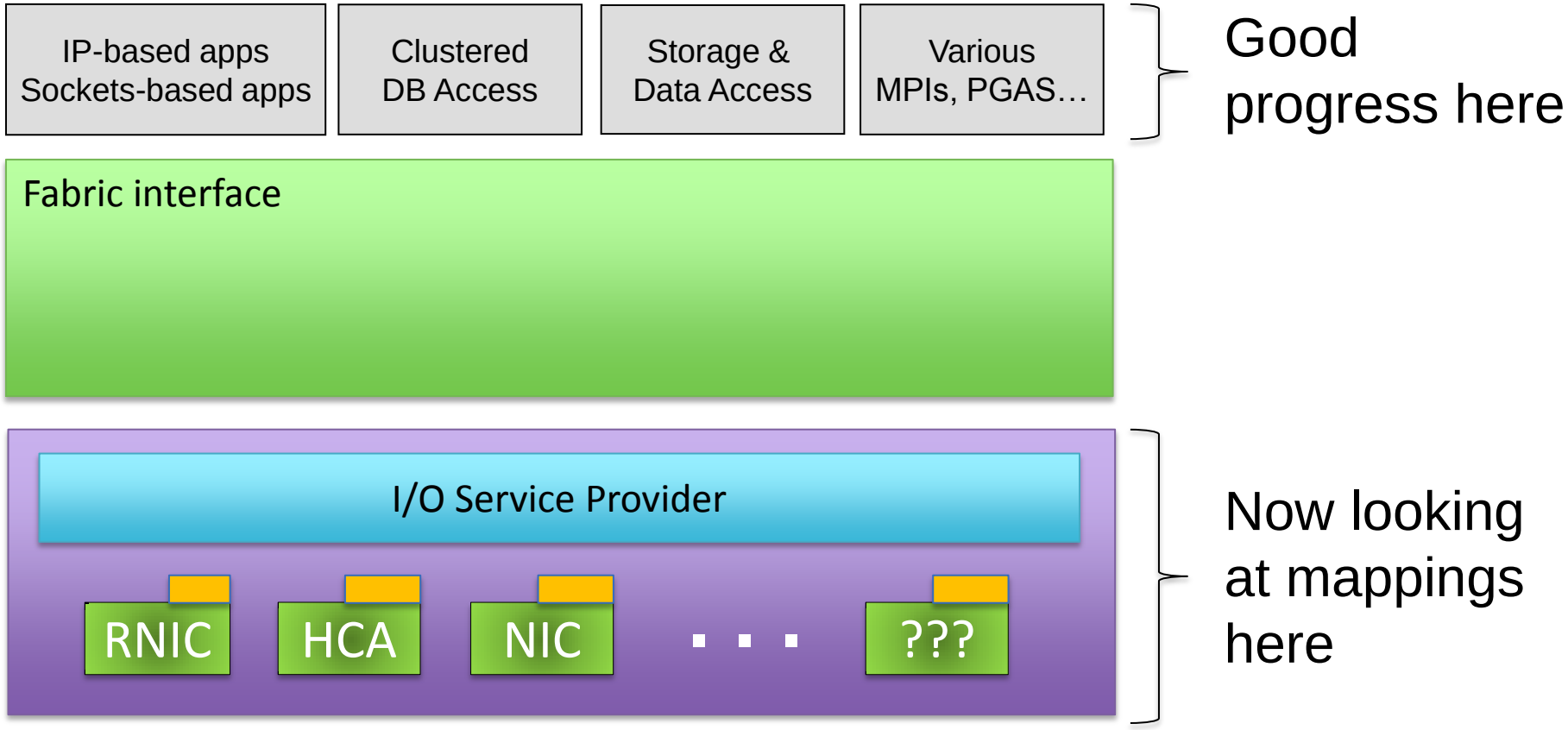


1. Software Transport Interface
2. RDMA Protocols
3. Network Transport Service

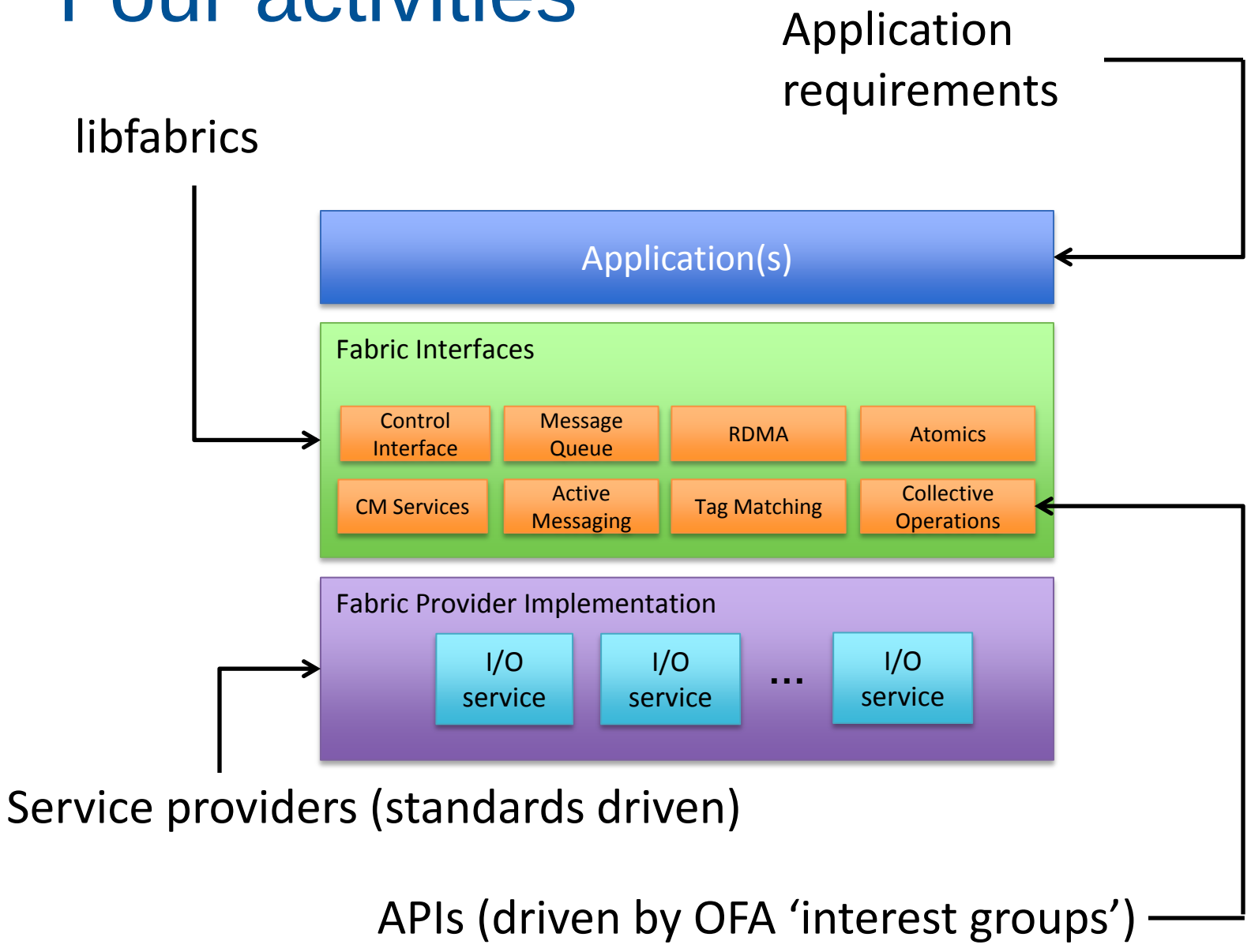
For example

IP-based, Sockets-based apps	Support for various types of legacy apps
Various MPIs, PGAS…	Distributed computing via message passing
File Systems	Network-attached file or object storage
Block Storage	Network-attached block storage
Clustered DB Access	Extracting value from structured data

Wire independence



Four activities



Some issues

- Memory registration – API or provider layer?
- Collectives operations
- Completions

OFI WG Process

Weekly telecons – Tuesdays at 9:00am PDT

All are welcome to participate

Group has well-defined processes to ensure progress

F-2-F meeting tonight following the OFA General Membership meeting

Thank You

OPENFABRICS ALLIANCE

10TH ANNUAL INTERNATIONAL DEVELOPER WORKSHOP