OPENFABRICS ALLIANCE

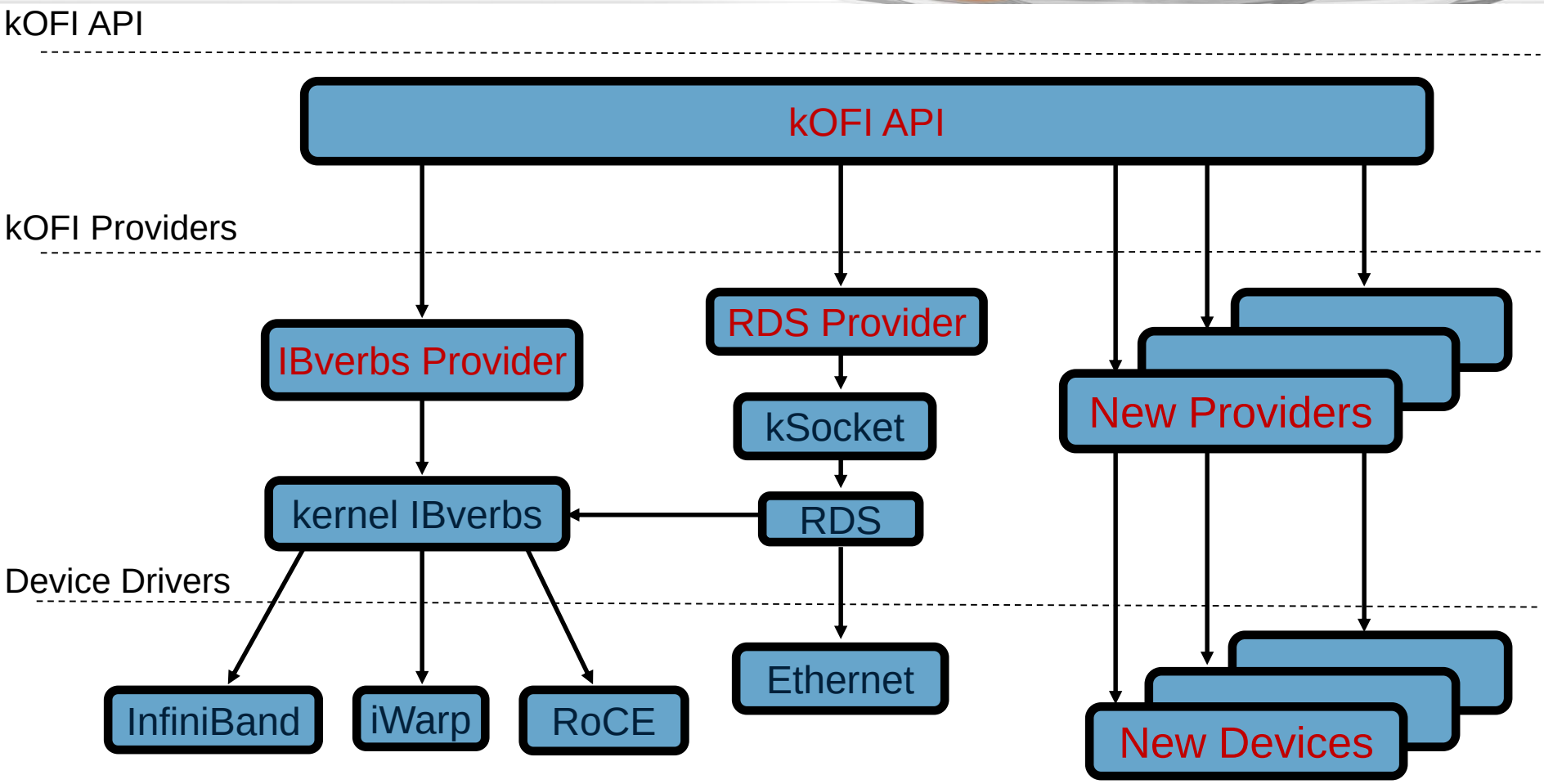
Kernel OpenFabrics Interface

kOFI Framework

Stan Smith Intel SSG/DPD

February, 2015

kOFI Framework



* Red indicates new kernel components

kOFI API

kOFI interfaces are designed such that they are cohesive and not simply a union of disjoint interfaces. The interfaces are logically divided into two groups:

- **control interfaces** are a common set of operations that provide access to local communication resources.

- **communication interfaces** expose particular models of communication and fabric functionality, such as message queues, remote memory access, and atomic operations. Communication operations are associated with fabric endpoints.

Kofi applications will typically use the control interfaces to discover local capabilities and allocate necessary resources. They will then allocate and configure a communication endpoint to send and receive data, or perform other types of data transfers, with remote endpoints.

kOFI API

kOFI API exports up

- fi_getinfo() fi_fabric() fi_domain() fi_endpoint() fi_cq_open() fi_ep_bind()
- fi_listen() fi_accept() fi_connect() fi_send() fi_recv() fi_read() fi_write()
- fi_cq_read() fi_cq_sread() fi_eq_read() fi_eq_sread() fi_close() …

KOFI API (extremely thin code layer)

kOFI API exports down

- **kofi_provider_register()**
  During kofi provider module load a call to kofi_provider_register() supplies the kofi-api with a dispatch vector for fi_* calls.

- **kofi_provider_deregister()**
  During kofi provider module unload/cleanup kofi_provider_deregister() destroys the fi_* runtime linkage for the specific provider (ref counted).

kOFI Application Flow

- Initialization
- Server connection setup (if required)
- Client connection setup (if required)
- Connection finalization (if required)
- Data transfer
- Shutdown

kOFI Initialization

- fi_getinfo(&fi)
  Acquire a list of desirable/available fabric providers.

- Select appropriate fabric (traverse provider list).

- fi_fabric(fi, &fabric)
  Create a fabric instance based on fabric provider selection.

- fi_domain(fabric, fi, &domain) create a fabric access domain object.

kOFI End Point setup

- fi_ep_open(domain, fi, &ep) create a communications endpoint.
- fi_cq_open(domain, attr, &CQ) create/open a Completion Queue.
- fi_ep_bind(ep, CQ, send/recv) bind the CQ to an endpoint
- fi_enable(ep) Enable end-point operation (e.g. QP->RTS).

kOFI connection components

- fi_listen() listen for a connection request
- fi_bind() bind fabric address to an endpoint
- fi_accept() accept a connection request
- fi_connect() post an endpoint connection request
- fi_eq_sread() blocking read for connection events.
- fi_eq_error() retrieve connection error information

kOFI Reliable Datagram transfer

- fi_sendto() post a Reliable Datagram send request
- fi_recvfrom() post a Reliable Datagram receive request.
- fi_cq_sread() synchronous/blocking read CQ event(s).
- fi_cq_read() non-blocking read CQ event(s).
- fi_cq_error() retrieve data transfer error information
- fi_close() close any kofi created object.

kOFI message data transfer

- fi_mr_reg(domain, &mr) register a memory region
- fi_close(mr) release a registered memory region
- fi_send(ep, buf, len, fi_mr_desc(mr), ctx)
  post async send from memory request.
- fi_recv(ep, buf, len, fi_mr_desc(mr), ctx)
  post async read into memory request.
- fi_sendmsg() post send using fi_msg (kvec + imm data).
- fi_readmsg() post read using fi_msg (kvec + imm data).

kOFI RDMA data transfer

- fi_write() post RDMA write.
- fi_read() post RDMA read.
- fi_writemsg() post RDMA write msg (kvec).
- fi_readmsg() post RDMA read msg (kvec).

kOFI message data transfer

- fi_send() post send.
- fi_recv() post read.
- fi_sendmsg() post write msg (kvec + imData).
- fi_recvmsg() post read msg (kvec+ imData).
- fi_recvv(), fi_sendv() post recv/send with kvec.

OPENFABRICS ALLIANCE

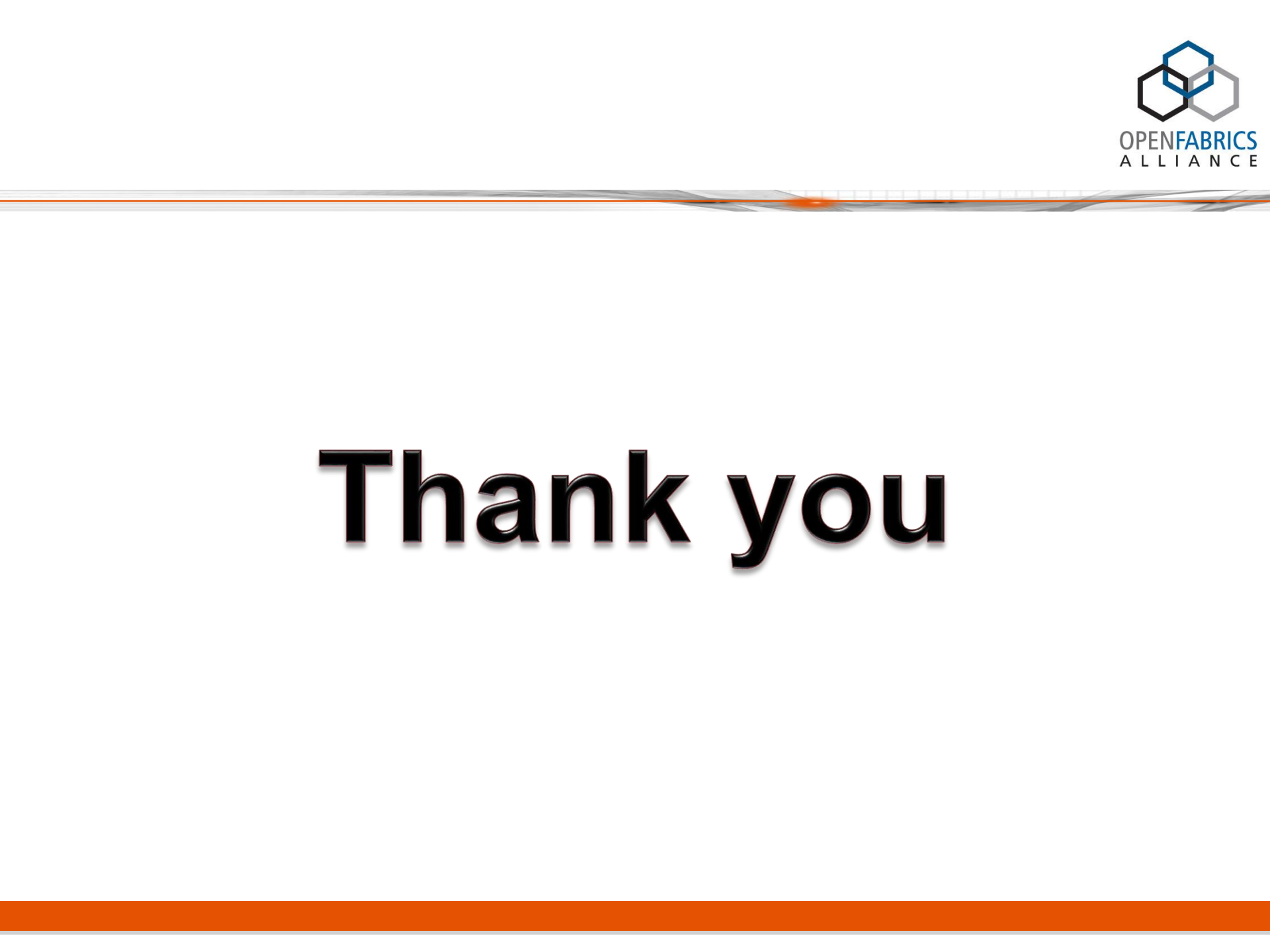
Back-up

KOFI Provider

kofi_provider_register (uint version, struct kofi_provider *provider)

kofi_provider_deregister (struct kofi_provider *provider)

```
struct kofi_provider {
        const char *name;
        uint32_t version;
        int     (*getinfo)(uint32_t version, const char *node,
                        const int service, uint64_t flags,
                        struct fi_info *hints, struct fi_info **info);
        int     (*freeinfo)(struct fi_info *info);
        int     (*fabric)(struct fi_fabric_attr *attr,
                        struct fid_fabric **fabric, void *context);
};
```


OPENFABRICS
ALLIANCE
Thank you