13th ANNUAL WORKSHOP 2017

ADVANCING OPEN FABRICS INTERFACES

Sean Hefty, OFIWG Co-Chair
Intel Corporation

March, 2017

MOVING LIBFABRIC FORWARD

Enhancements

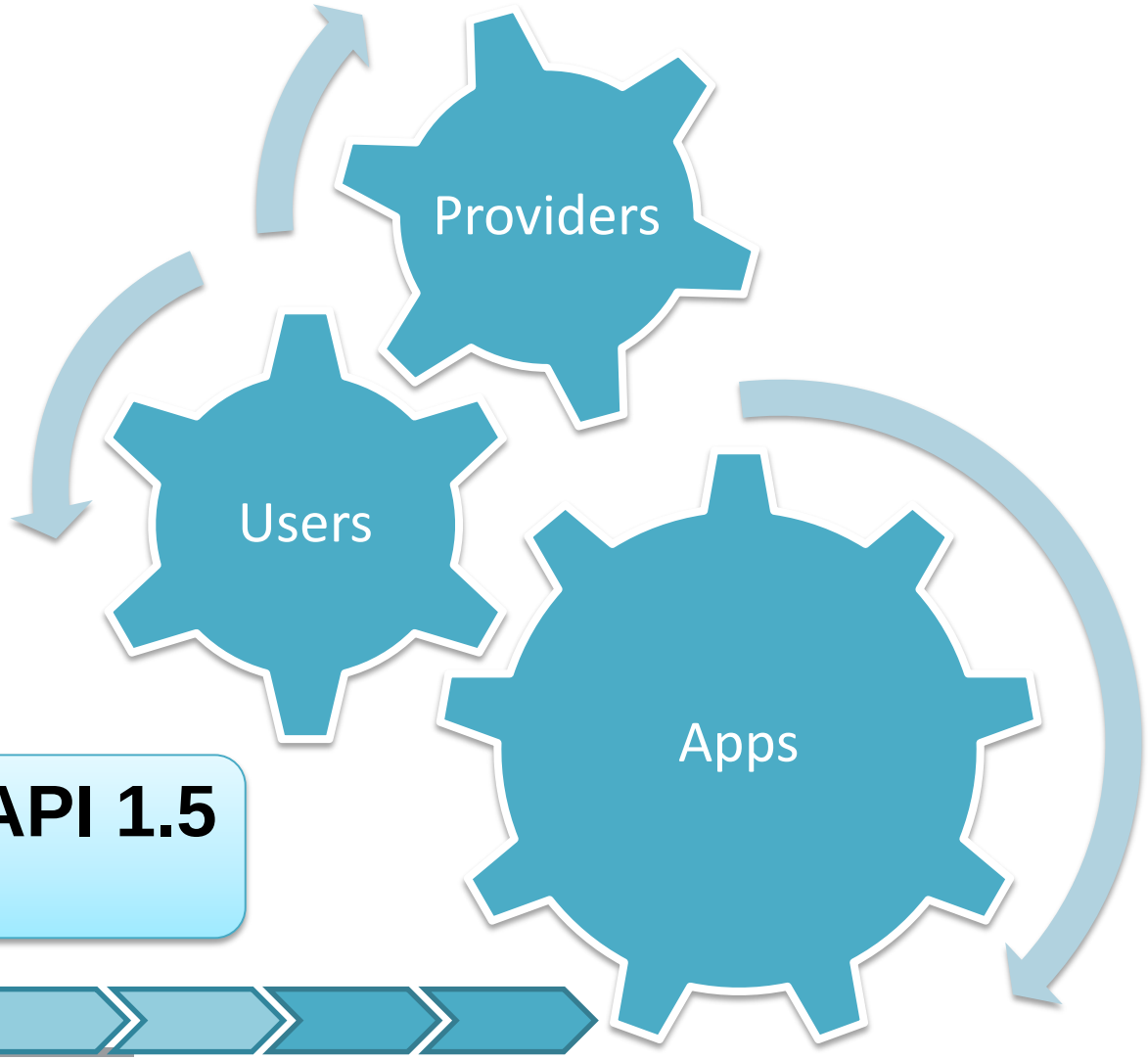
- Deferred and new features

API adjustments

- Developer feedback

Performance optimizations

- Provider implementation



Many features require migration to API 1.5

Existing apps work unmodified

OFI DEVELOPER GUIDE

New!

What a novel concept!

Broad coverage of libfabric

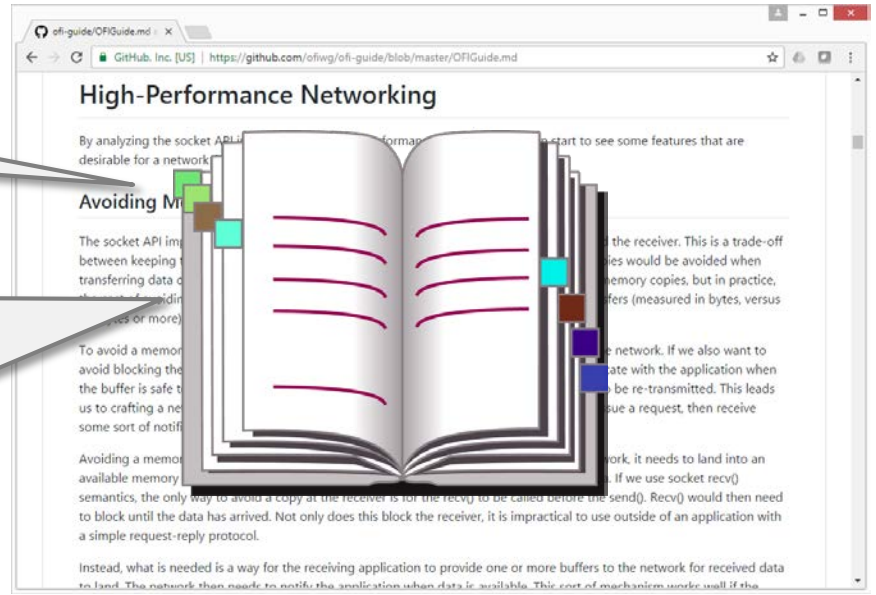
Design motivations, architecture, usage guidance

Assumes minimal background

Reviews sockets API for context

Analyzes HPC requirements and their impact on an API

Memory copies, network buffering, asynchronous operations, interrupts, direct HW access, ...



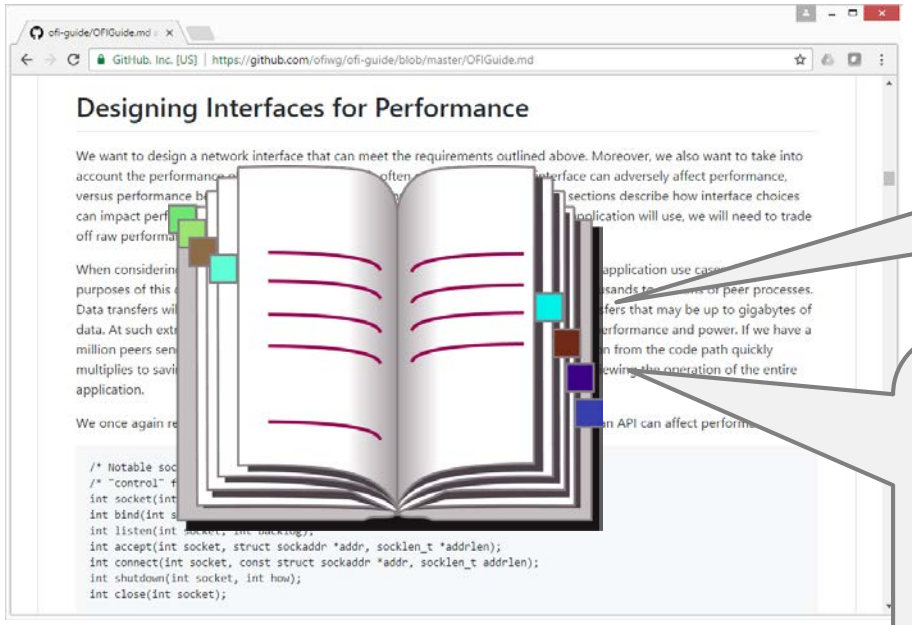
Lower the barrier to adoption

Many concepts are generic

OFI DEVELOPER GUIDE

New!

Owl read this later



Examines API trade-offs

Call setup, branches, memory footprint, …

Defines communication primitives

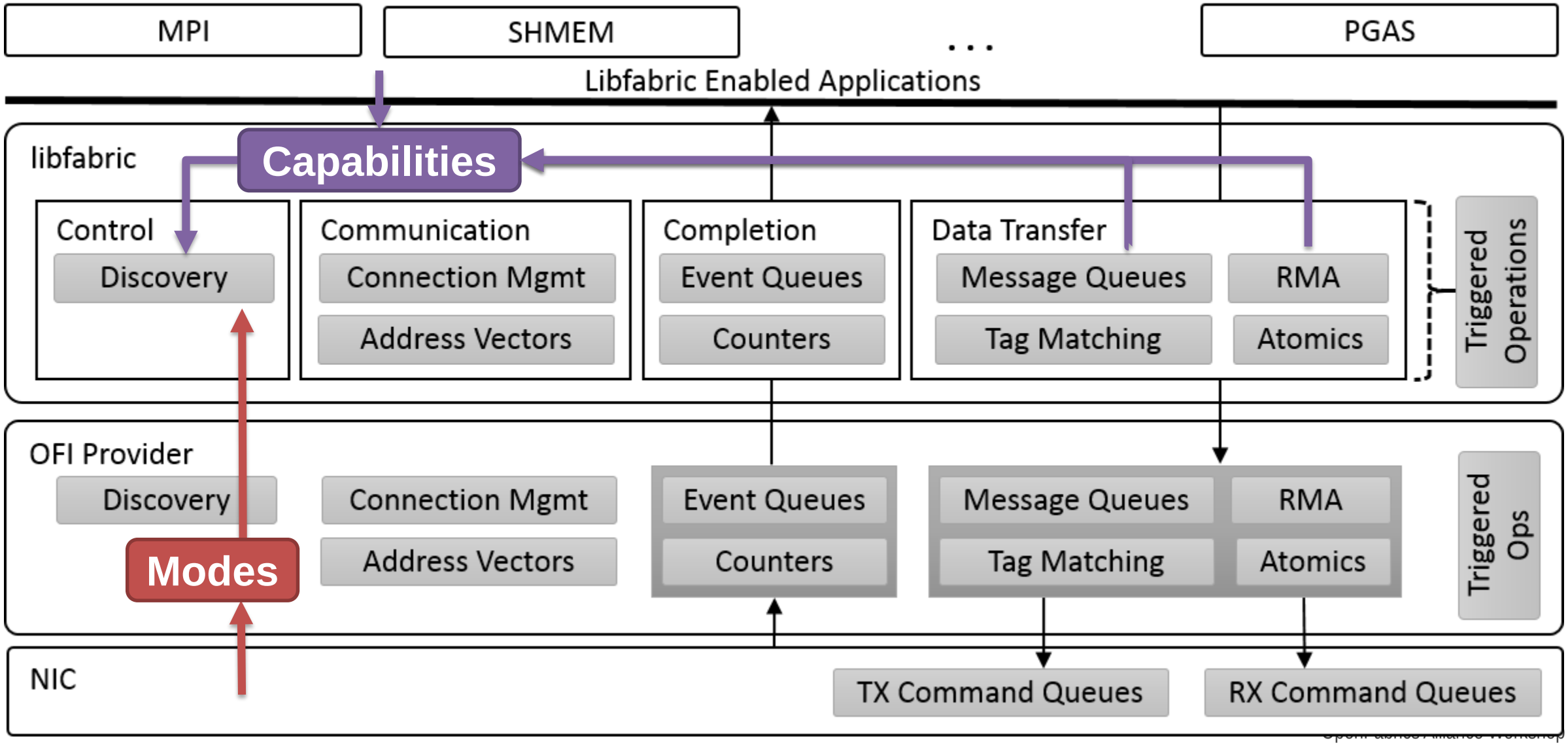
Command queues, memory buffers, completion notification, progress models, data and message ordering, …

***Then* it goes into detail about OFI**

https://github.com/ofiwg/ofi-guide

ARCHITECTURE

Old!

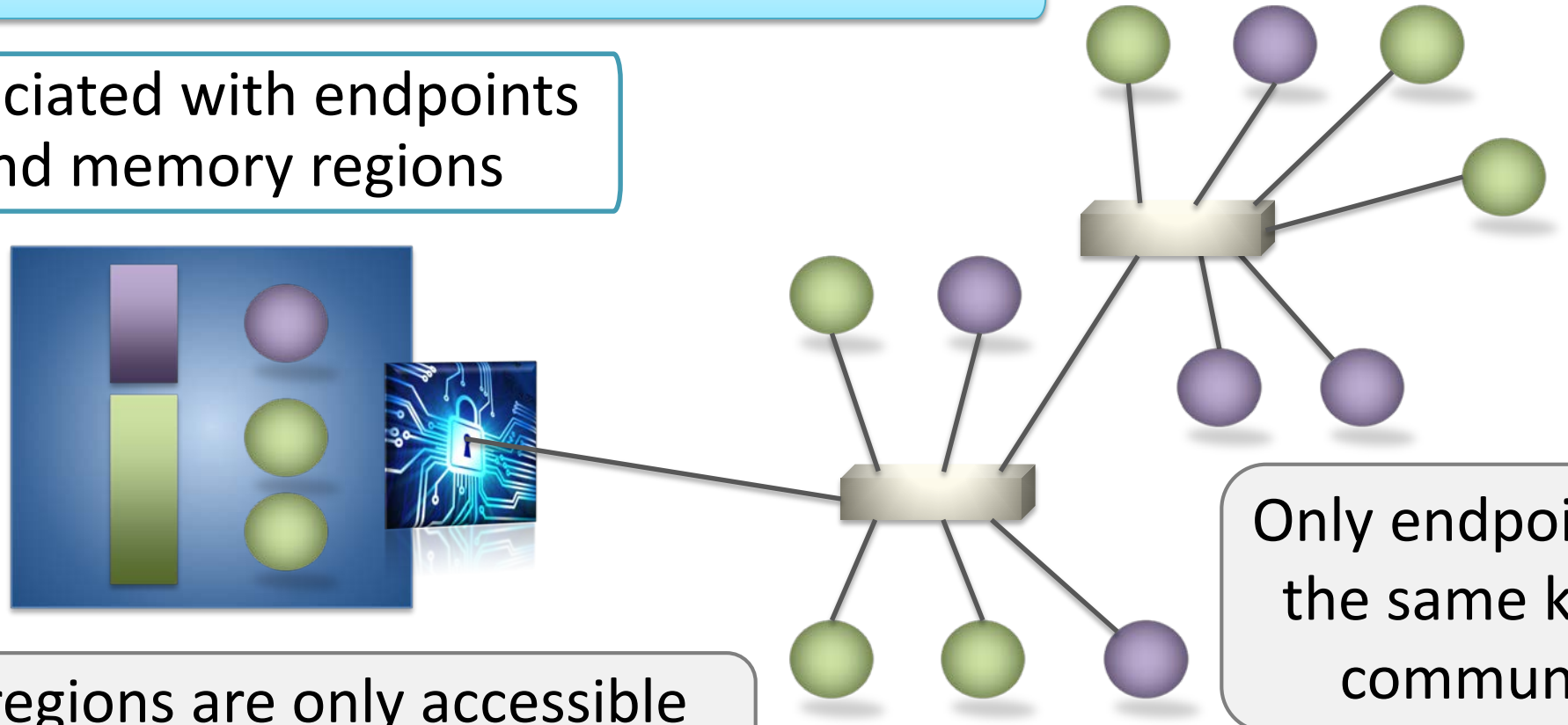


Job Keys

AUTHORIZATION KEYS

Isolate traffic between virtual sets of peers

Associated with endpoints and memory regions



Only endpoints with the same key may communicate

Memory regions are only accessible through endpoints with matching keys

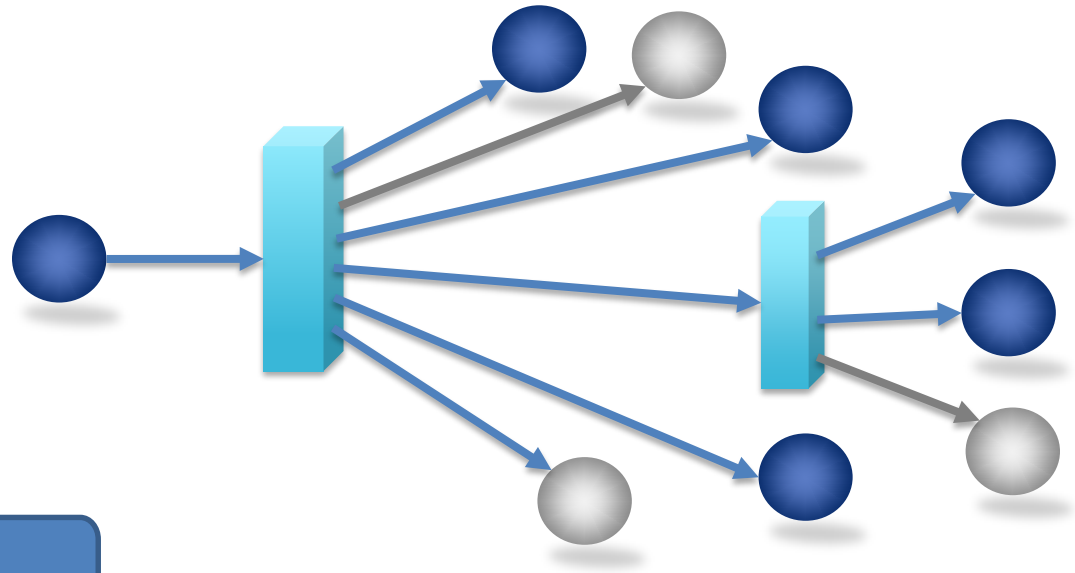
At Long Last

MULTICAST

Datagram endpoint capability

New fi_join call to connect to group

- Supports send-only joins
- Returns fabric address (fi_addr_t) for group



Uses existing data transfer APIs

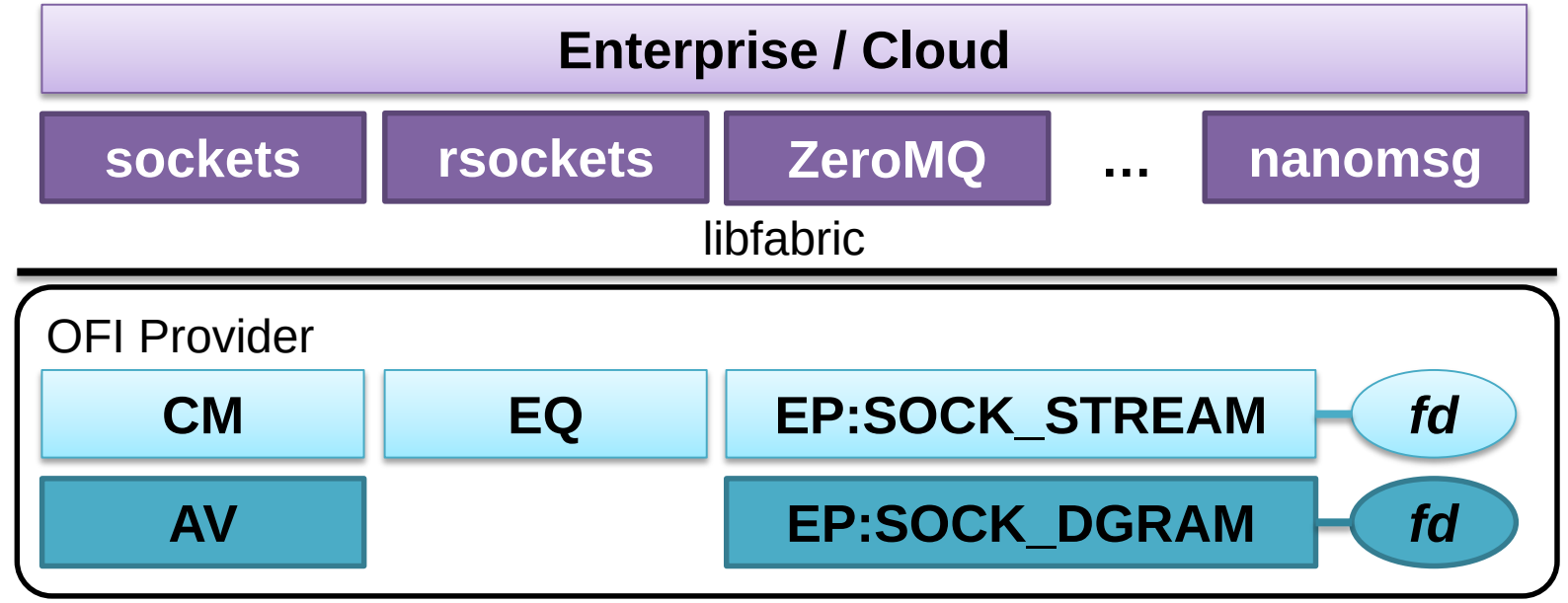
- Only message queue interface currently defined
- Multicast sends require use of FI_MULTICAST flag
  - Distinguish between unicast and multicast address
- Received messages also marked with FI_MULTICAST

Beyond HPC

NEW ENDPOINT TYPES

Apps/middleware designed around socket semantics

Synchronous completions
- No completion queue
- Application owns buffers

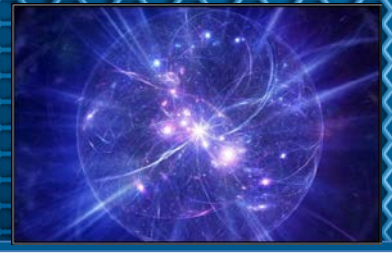


Enable provider specific optimizations
- RMA, circular receive buffers
- Minimize data copies, use offload

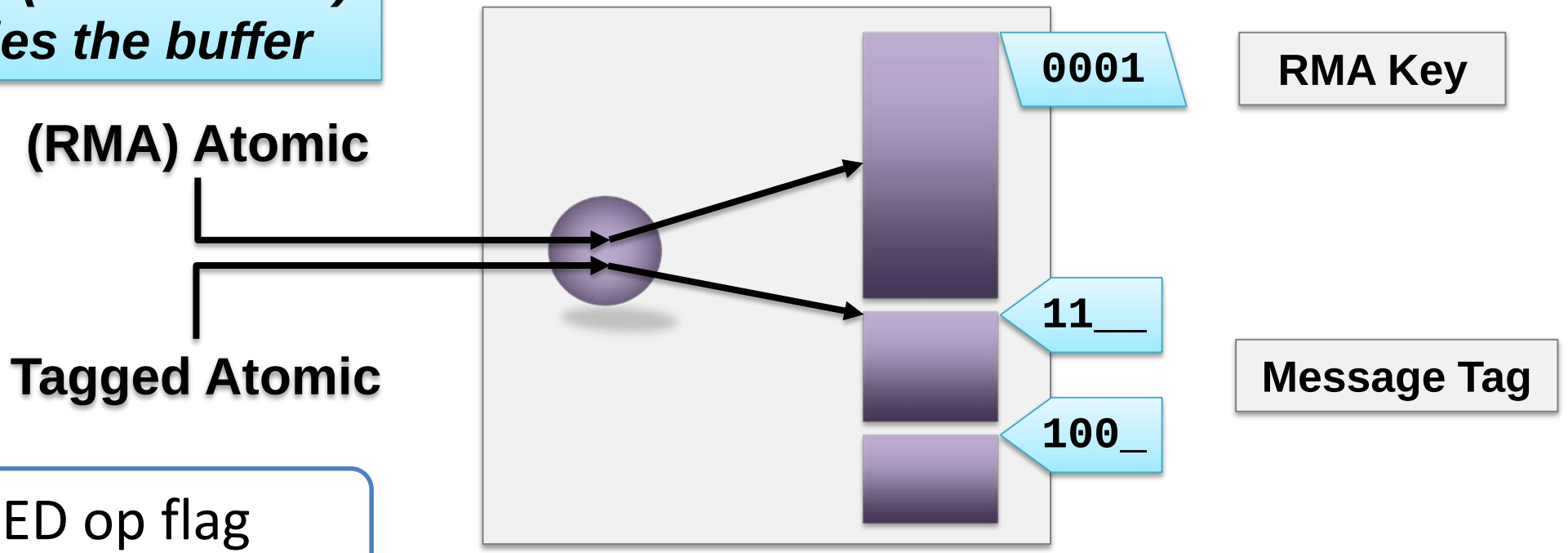
Associated with file descriptor
- Support for select/poll

Free:
100% More

EXPANDING ATOMICS



RMA key (not address) identifies the buffer



FI_TAGGED op flag
- Replace key with tag

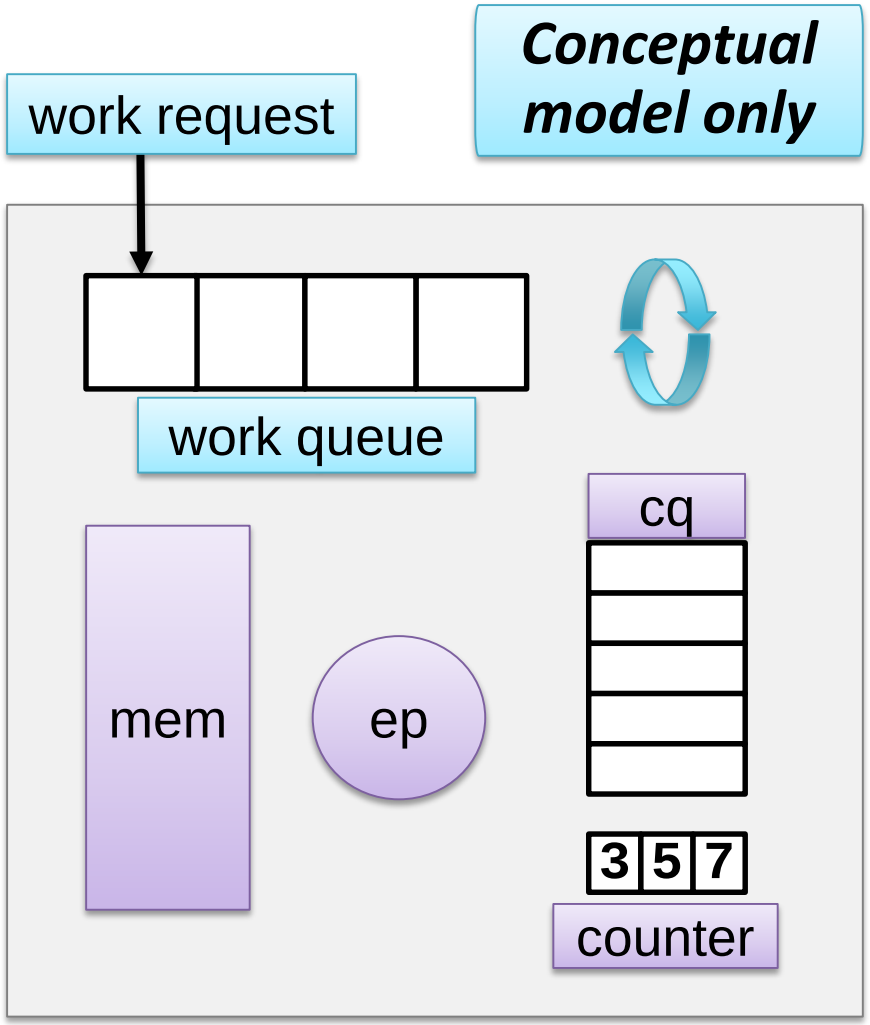
fi_atomic_query()
- Domain level call
- Also reports datatype size

DEFERRED WORK QUEUES

Experimental

Generalizes triggered operations

Conceptual domain level work queue



Allows (theoretical) associations between any domain level object

Definition is limited

Primitives to develop and evaluate collective APIs

e.g. directed acyclic graphs

Reducing Overhead

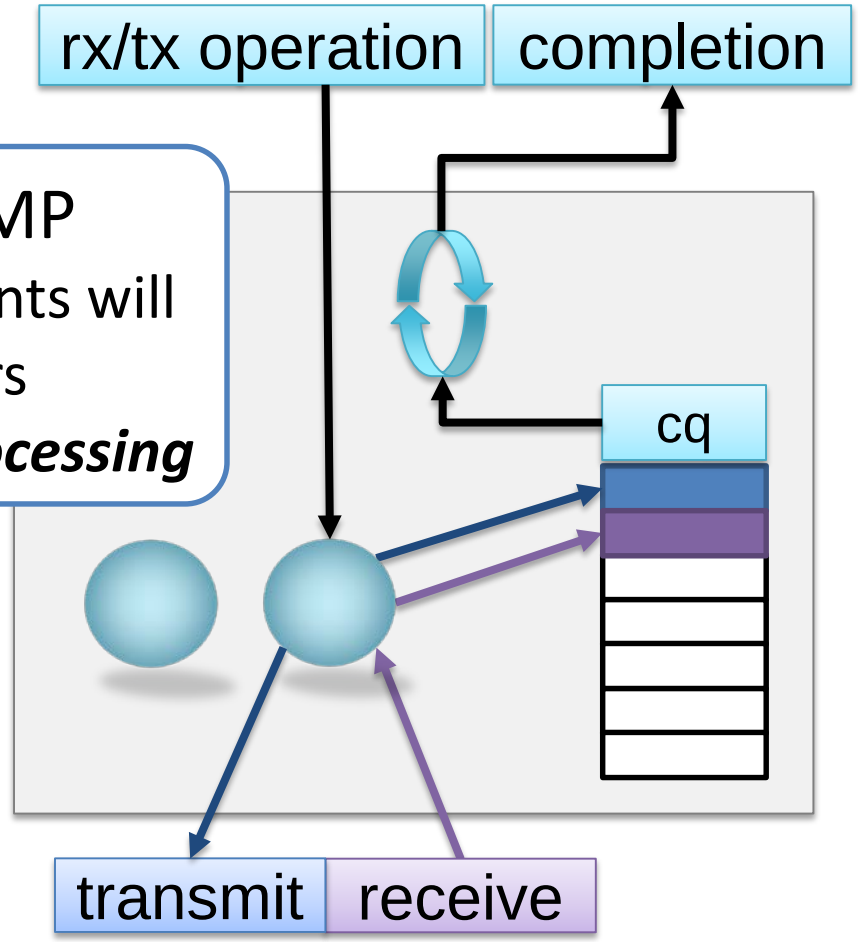
OPTIMIZING COMPLETIONS

FI_RESTRICTED_COMP

Only similar endpoints will share CQs / counters

Eliminates post-processing

New mode bits



FI_NOTIFY_FLAGS_ONLY

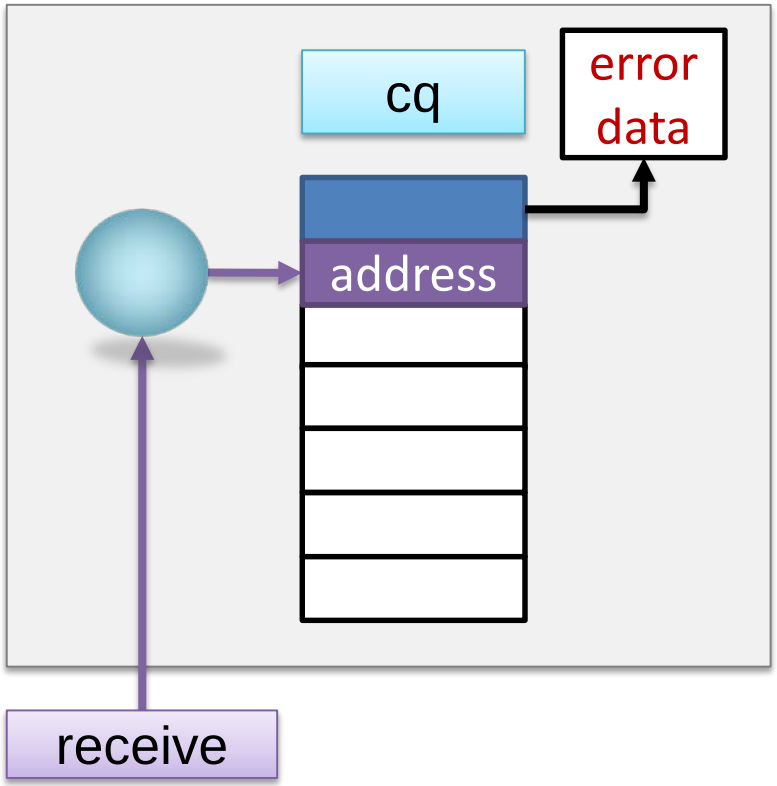
- Drops operation flag
- Retains notification flags:
  - Remote CQ data
  - Multi-recv buffer freed

Avoids op save/restore

App Request

COMPLETION DATA

Multi-Threading



Reporting errors

- Provider specific data reported for debugging
  - Data is opaque to application
  - Storage managed by provider
- Report maximum size of error data
- Application provides buffer

Improves multi-threaded completion processing

FI_SOURCE_ERR capability

- Validate source address against local AV
- Report raw fabric address if not found

  EADDRNOTAVAIL completion error

Pushes 'connect' address exchange to provider

Feedback

FEATURE GRAB BAG

New attributes

- Fabric: api version
- Domain: counter limits,
  memory region iov limits

String based addressing

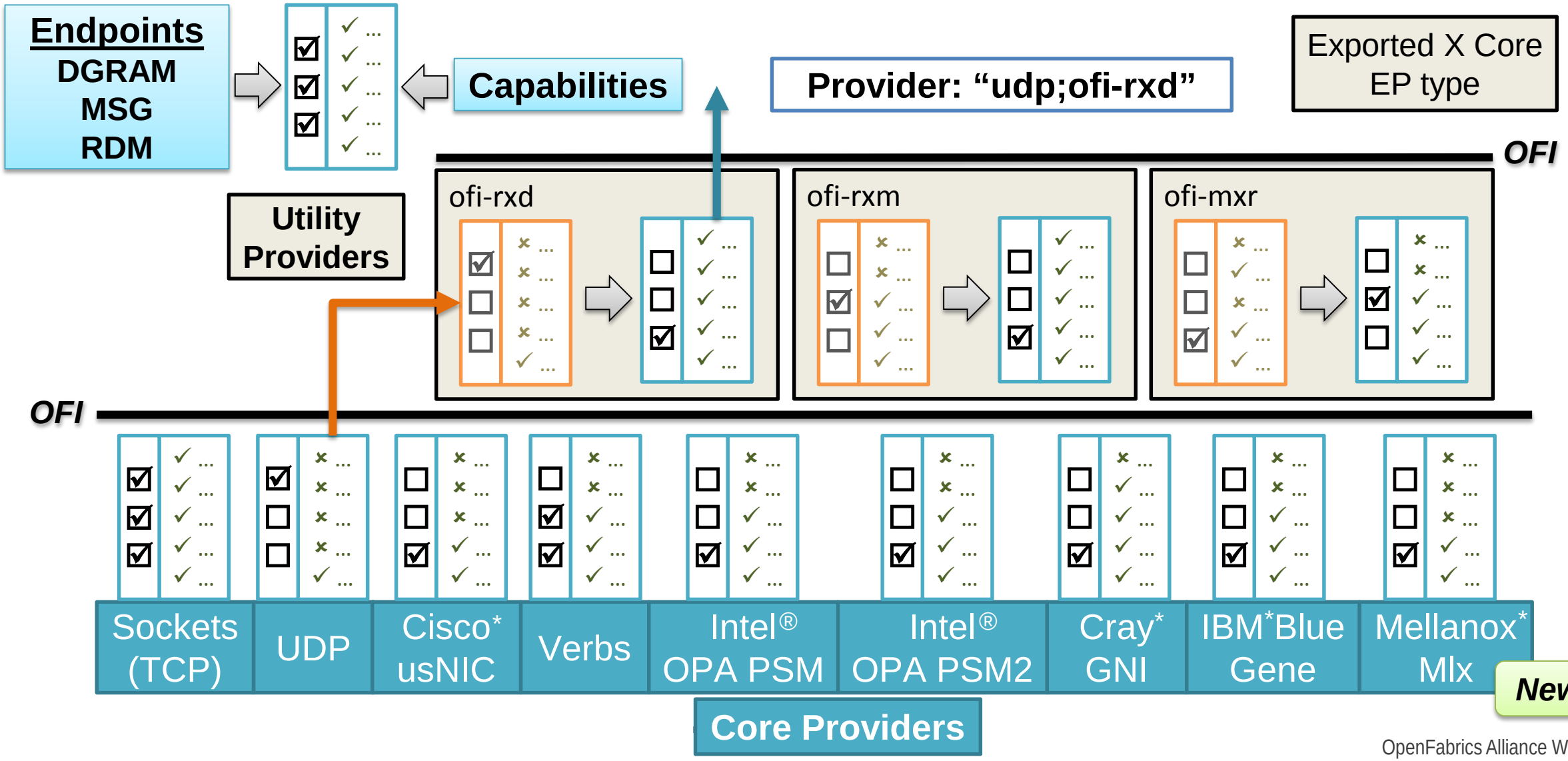
- FI_ADDR_STR address format
- Generic string format for any address
  - Application does not *need* to interpret
  - Well-known format for sockaddr
- Pass directly to fi_getinfo() or AV insertion

Communication scope

- Local: within the local node
  - Shared memory, most NICs
- Remote: with external nodes
  - Any NIC
- Evading more refined definitions

Simplify Development

PROVIDER UPDATES



Exposing the Pain

MEMORY REGISTRATION

FI_MR_BASIC
Traditional RDMA fabrics

FI_MR_SCALABLE
Application driven capability

FI_MR_NEITHER
What other fabrics support

Exposing the Pain

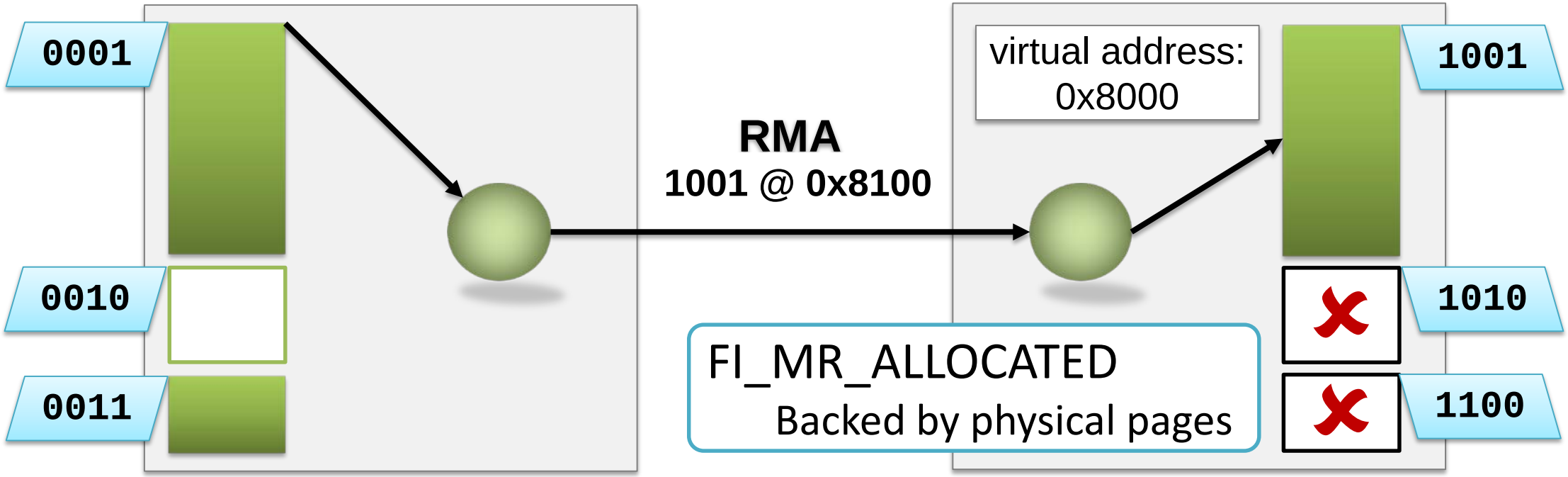
MR MODE BITS

- **Redefine domain attribute mr_mode**
  - enum → integer flags
- **Divide FI_MR_BASIC into specific restrictions**
  - FI_MR_SCALABLE implied if mr_mode = 0
- **FI_MR_BASIC, FI_MR_SCALABLE kept for compatibility**
  - Behavior dependent on selected API version

Exposing the Pain

'BASIC' MR MODE BITS

FI_MR_VIRT_ADDR
Target offset is virtual address



FI_MR_ALLOCATED
Backed by physical pages

FI_MR_LOCAL
Locally accessed buffers must be registered

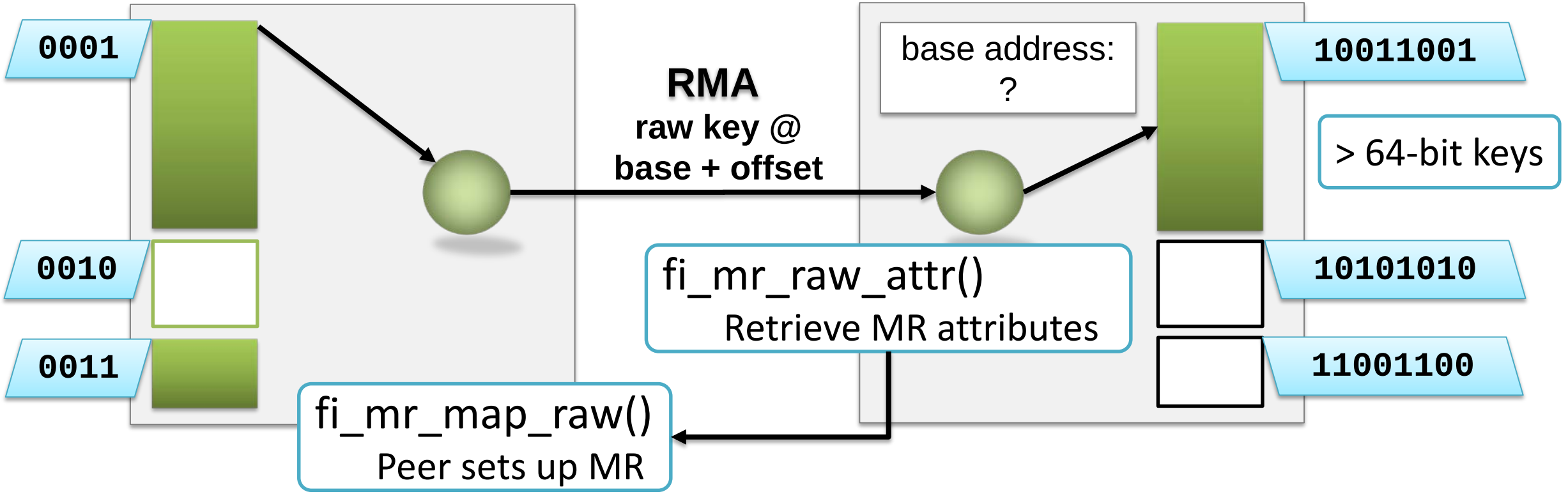
FI_MR_PROV_KEY
Provider sets key

It's only a flesh wound

RAW MEMORY REGIONS

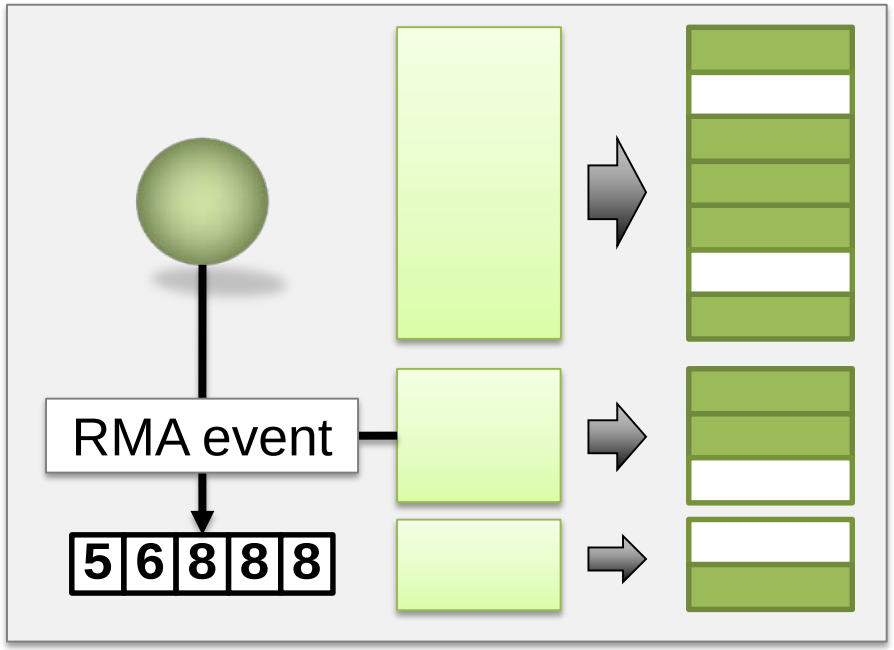
FI_MR_RAW
More generic version of 'basic' registration

Provider selects base address



Annoying itch

NON-OMNIPOTENT PROVIDERS

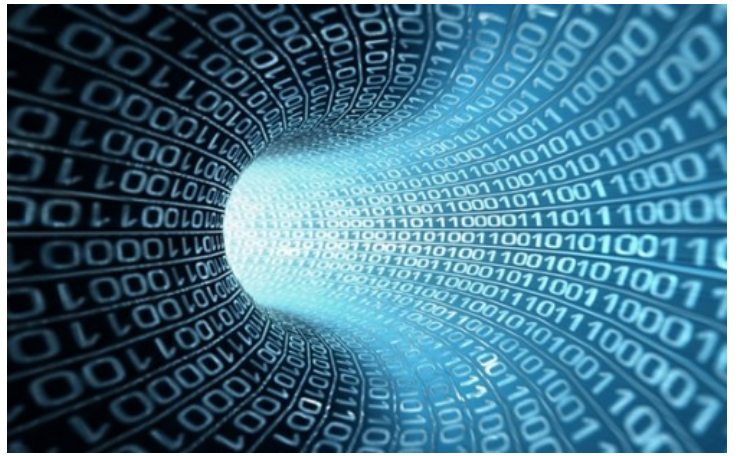


FI_MR_MMU_NOTIFY

- App notifies provider when physical pages backing MR change
- I.e. almost scalable, but NIC not linked into MMU
- Only those addresses accessed need to be backed

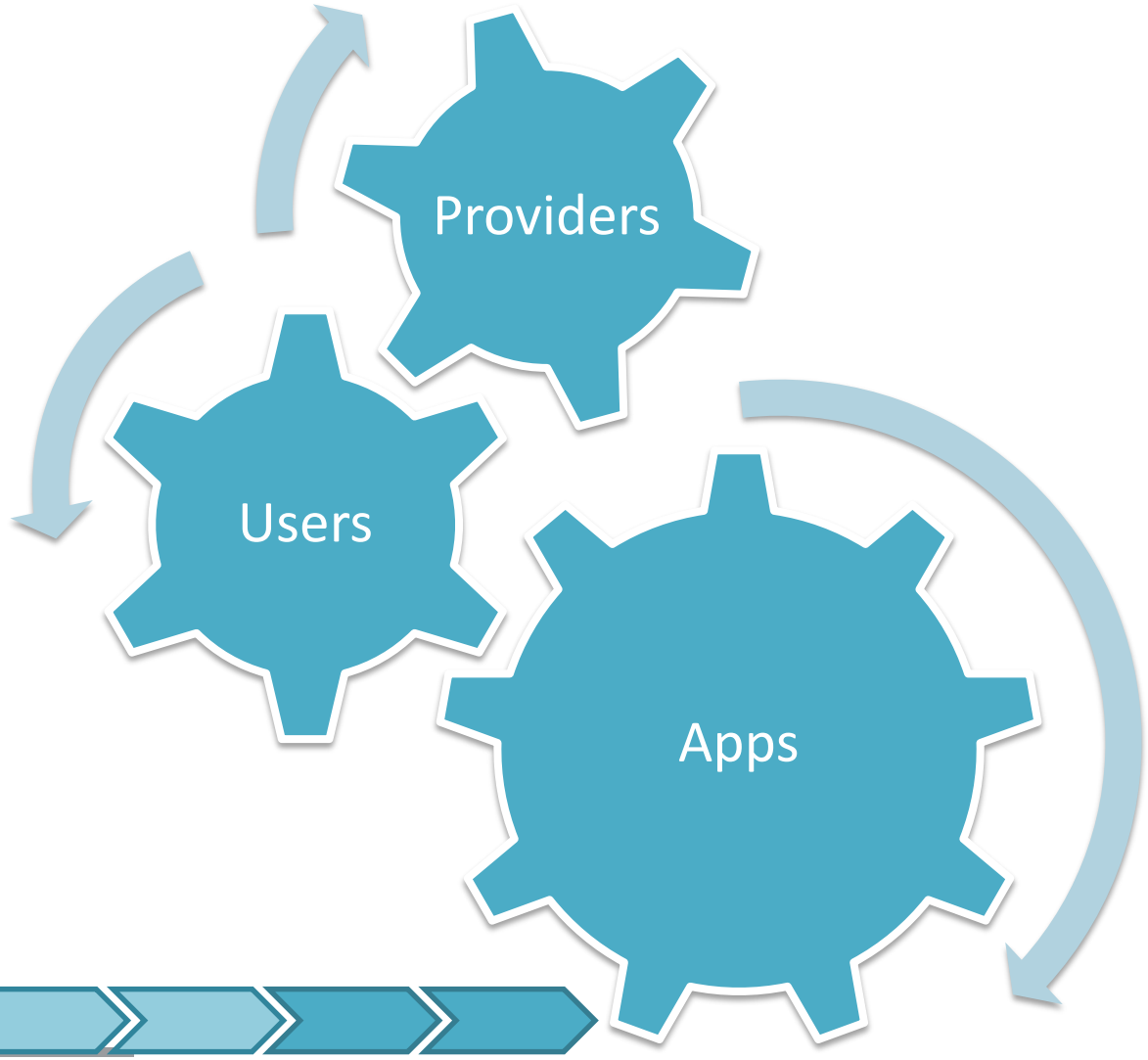
FI_MR_RMA_EVENT

- App indicates if MR will be used with RMA event counting
- MRs must be enabled after resource binding



LOOKING BACK

- Balance application needs, developer requests, and provider capabilities
- New features for expanding use cases
- Developer assistance
- Sensible performance optimizations



13th ANNUAL WORKSHOP 2017

THANK YOU

Sean Hefty, OFIWG Co-Chair

Intel