OPENFABRICS ALLIANCE

2022 OFA Virtual Workshop

COMMUNICATION AND COMPUTATION API COMPOSABILITY

XPU SUPPORT IN OFI

Sean Hefty

Intel Corporation

April 2022

OVERVIEW

Introduction:

- Heterogeneous memory
- Computational awareness

NIC-XPU coordination:

- Workflow model
- XPU initiated communication
- NIC initiated computation

Other thoughts:

- Restrictions
- Exploration

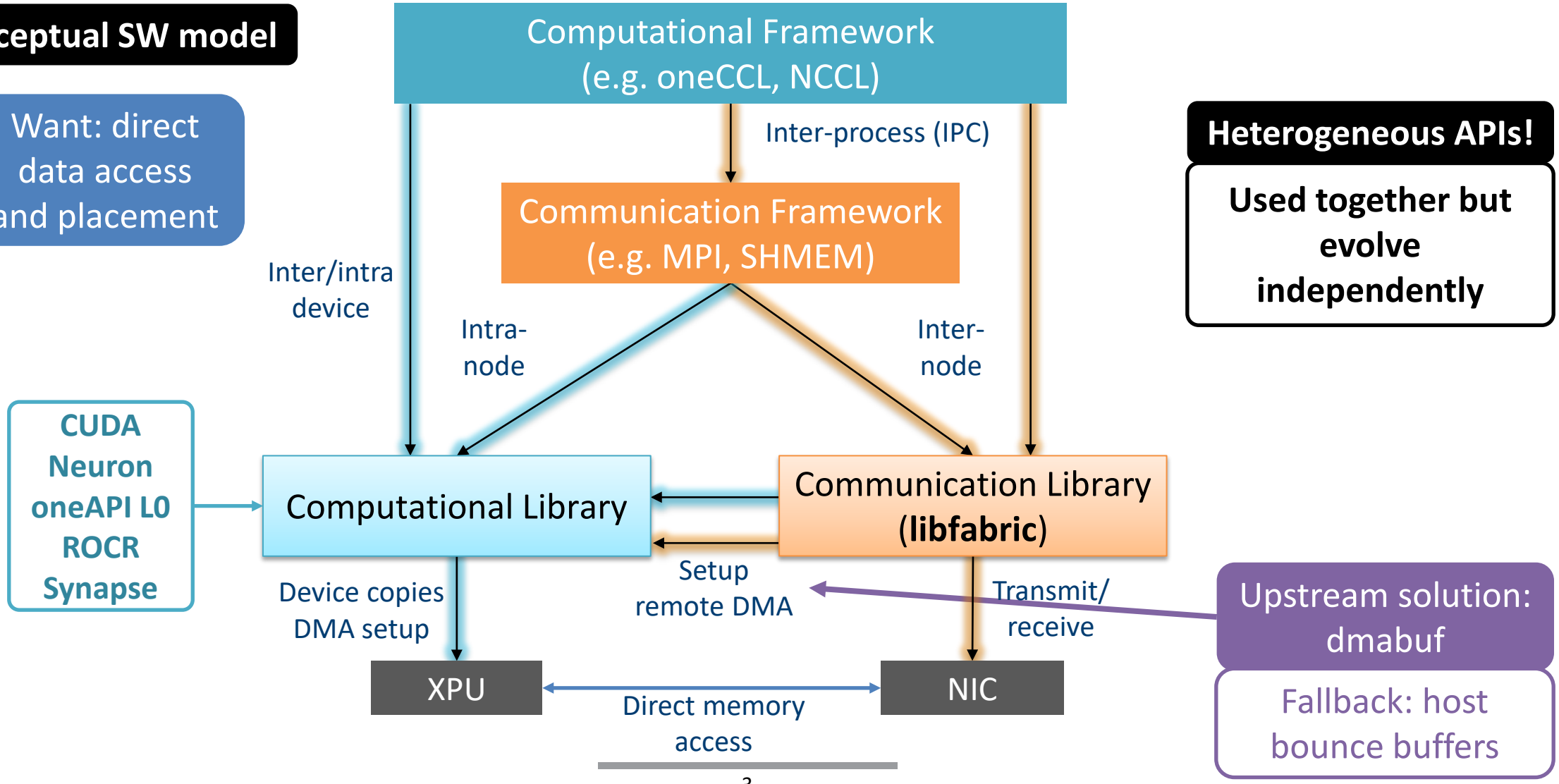
Focus on enabling the technology

No discussion on effectiveness

INTRODUCTION
HETEROGENEOUS MEMORY SUPPORT

Conceptual SW model

Want: direct data access and placement



Heterogeneous APIs!

Used together but evolve independently

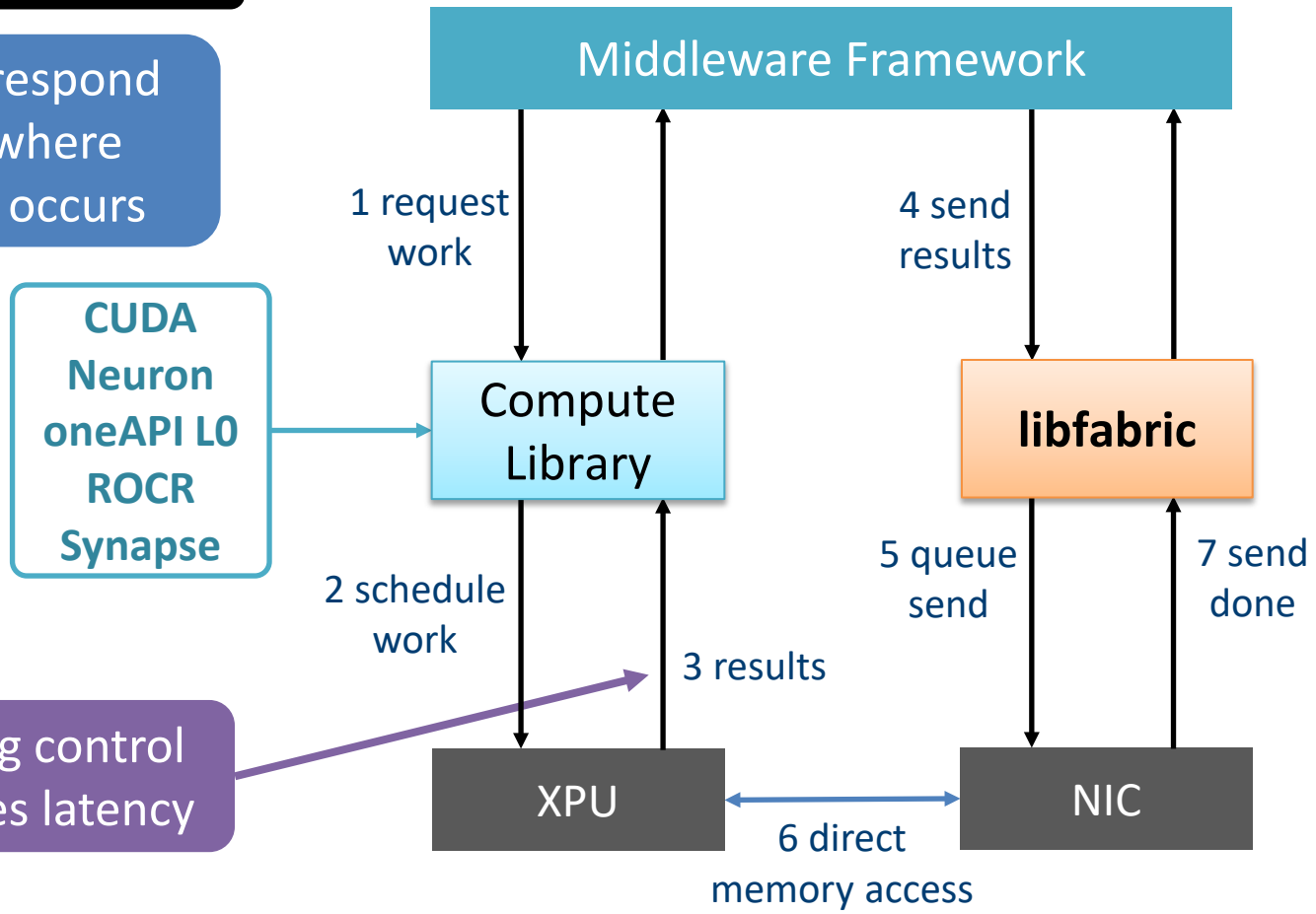
Upstream solution: dmabuf

Fallback: host bounce buffers

INTRODUCTION
COMPUTATIONAL AWARENESS

Conceptual SW flow

Want: correspond directly where *execution* occurs



Returning control introduces latency

Optimizations

XPU initiated communication

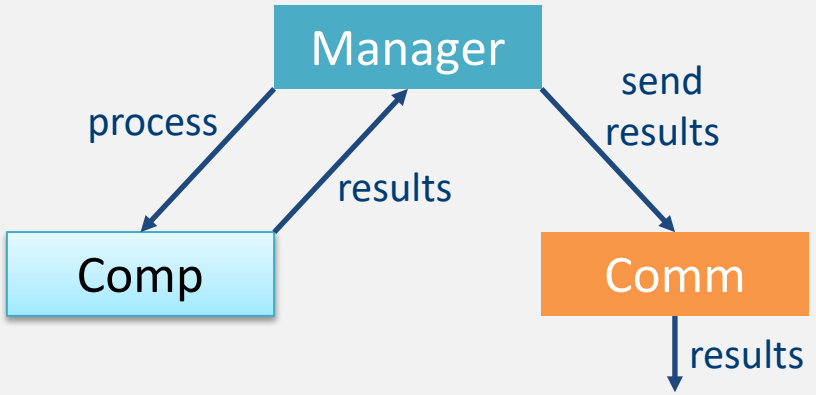
NIC initiated compute

Heterogeneous HW!

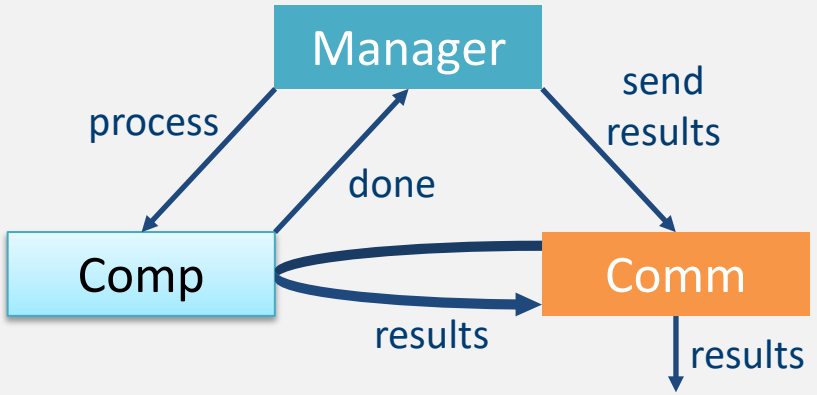
Do not assume implementation!

NIC-XPU COORDINATION
WORKFLOWS

Transmit



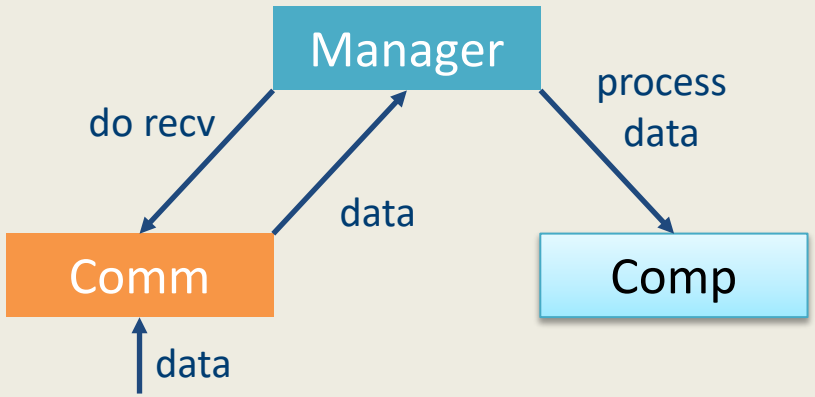
Example: GPU + standard NIC



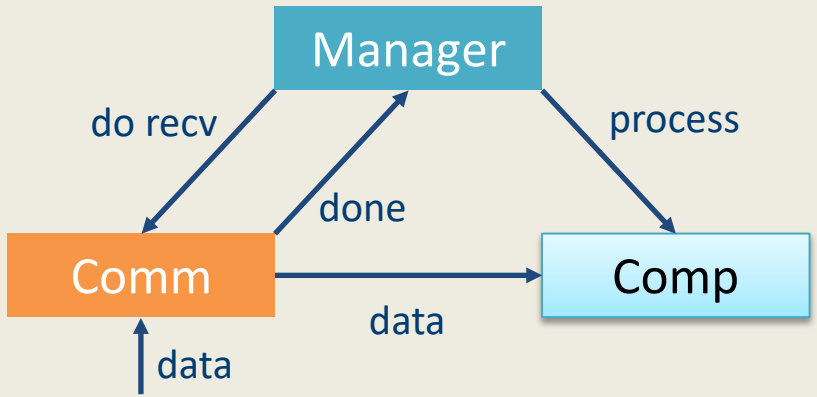
Example: GPU + RDMA NIC

Receive

"receive" may be RMA setup



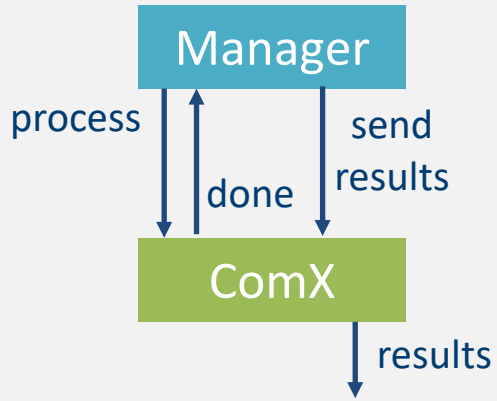
Example: standard NIC + GPU



Example: RDMA NIC + GPU

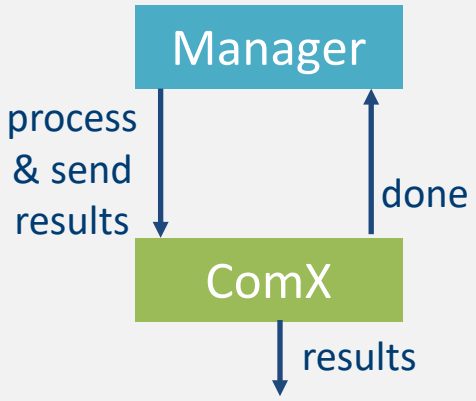
NIC-XPU COORDINATION
WORKFLOWS

Transmit



Example: need to coordinate with recipient

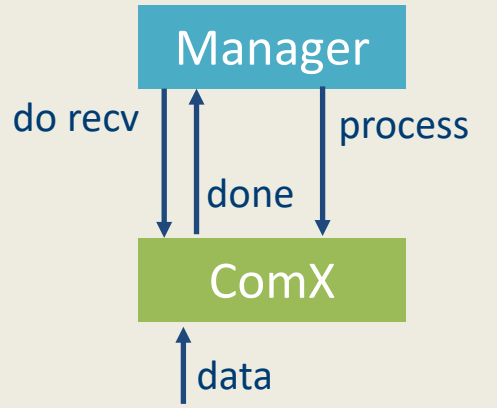
Example: GPU + Integrated NIC



Example: can immediately send results

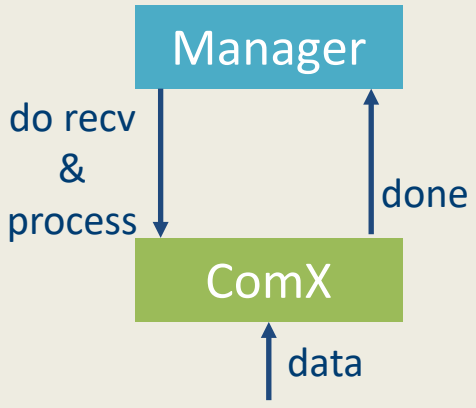
Example: GPU + Integrated NIC

Receive



Example: need to coordinate other data

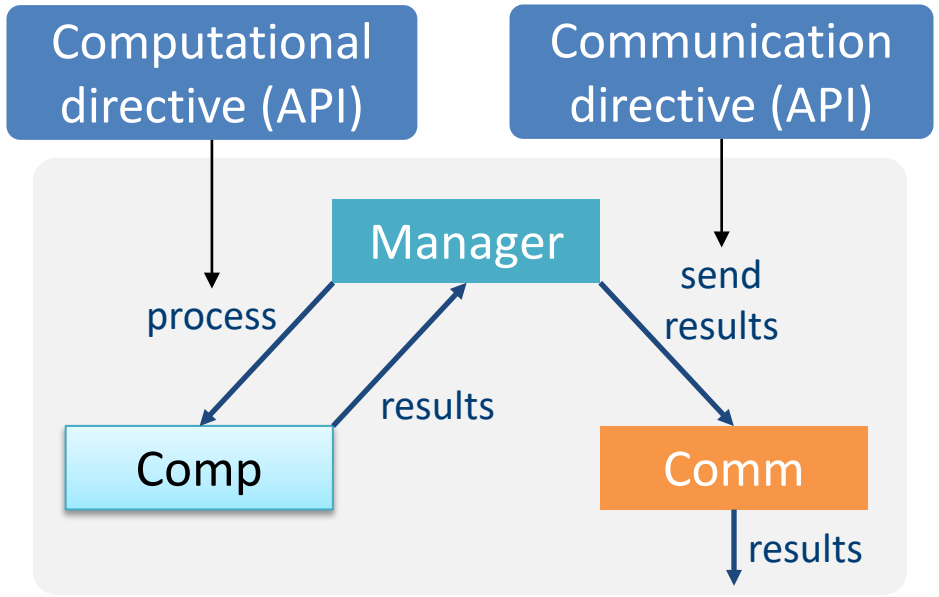
Example: GPU + Integrated NIC



Example: can immediately process data

Example: GPU + Integrated NIC

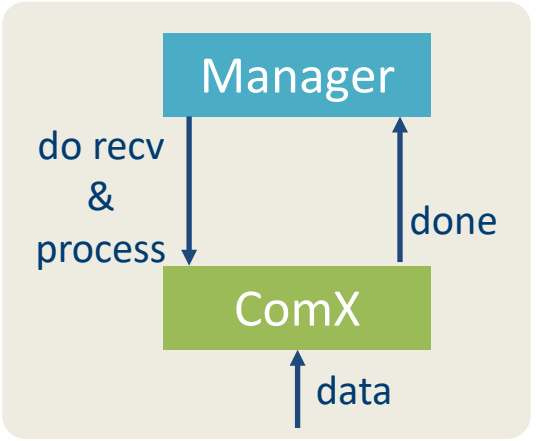
NIC-XPU COORDINATION
WORKFLOWS



Example: GPU + standard NIC

Want: common model

extremes



Example: GPU + Integrated NIC

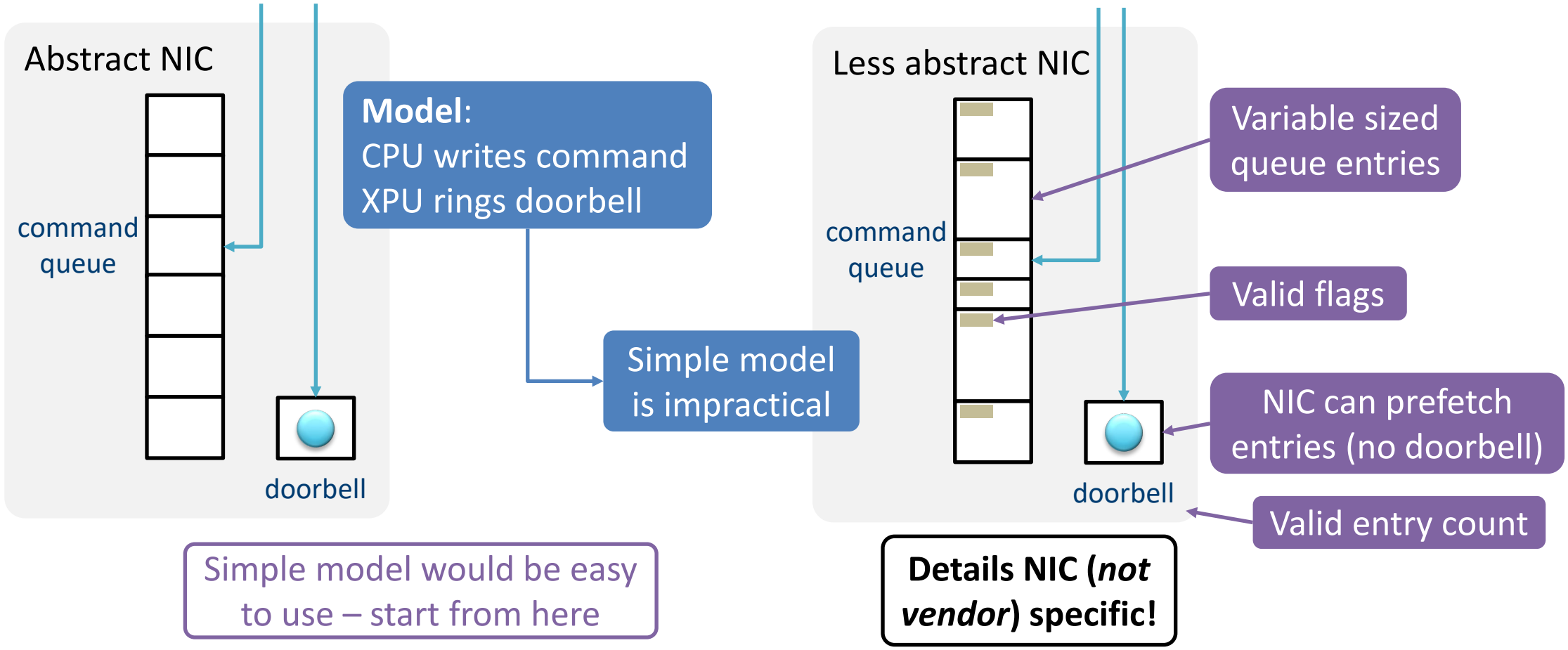
Challenge:

Find a coordinating solution without assuming HW implementation on either side!

Critical Challenge:

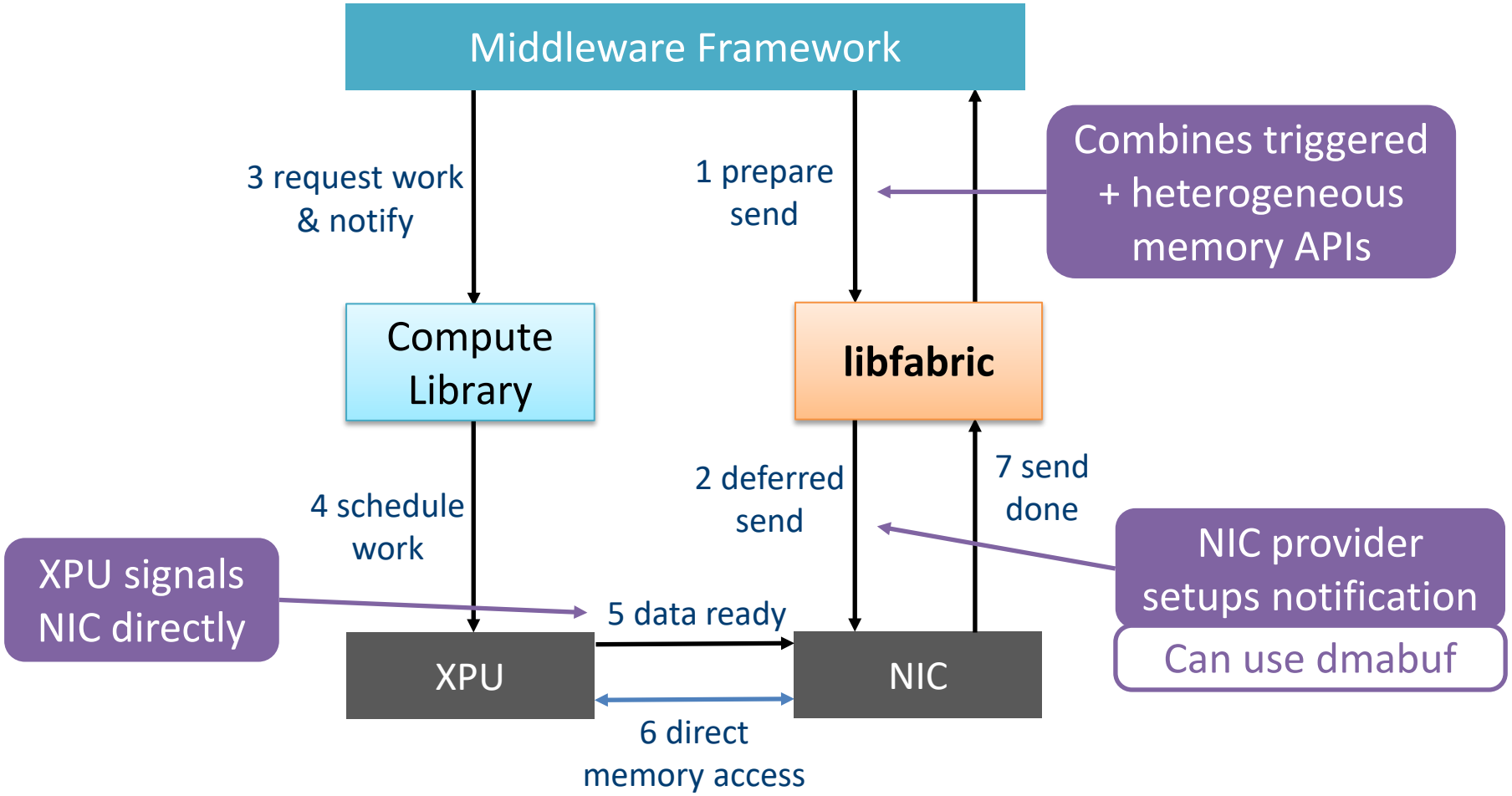
Define APIs that mere humans can figure out how to use!

XPU INITIATED COMMUNICATION
CONCEPTUAL MODEL



XPU INITIATED COMMUNICATION

PHASE I: XPU TRIGGERED OPS



XPU INITIATED COMMUNICATION

PHASE I: XPU TRIGGERED OPS

Universal triggering mechanism

Supports broad range of NIC implementations

CPU queues XPU triggerable transfer

```
fi_writemsg(endpoint, msg, FI_TRIGGER)
```

references

```
struct fi_triggered_xpu {
        enum fi_hmem_iface iface;
        union {
                int cuda;
                int ze;
        } device;

        int count;
        struct fi_trigger_var var[];
};
```

XPU

array of triggering variables

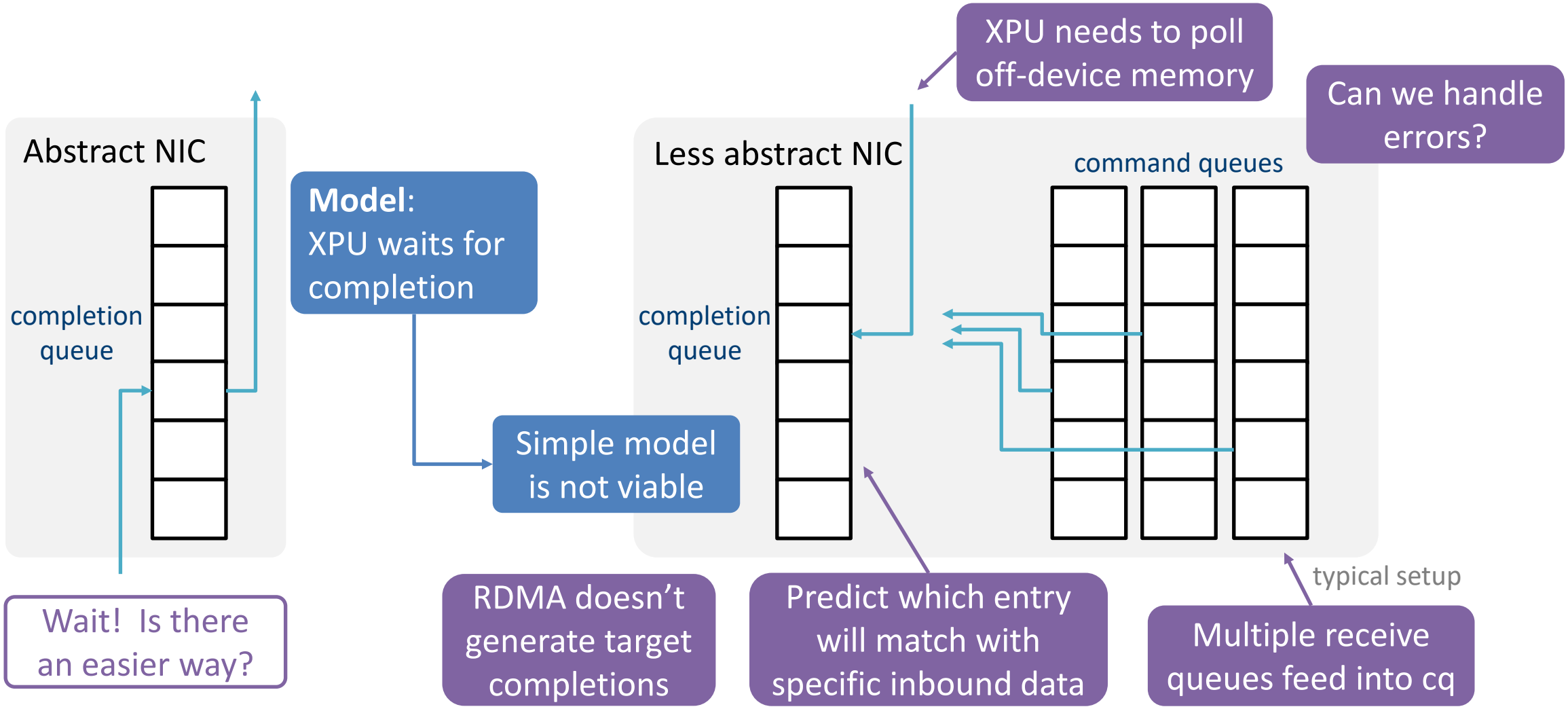
Writing to variables trigger the transfer

Likely: couple variables

```
struct fi_triggered_var {
        enum fi_datatype data_type;
        int count
        void *addr;
        union {
                uint8_t val8;
                uint16_t val16;
                uint32_t val32;
                uint64_t val64;
                uint8_t *data
        } value;
};
```

Compute side requirement: write memory (possibly across PCI)

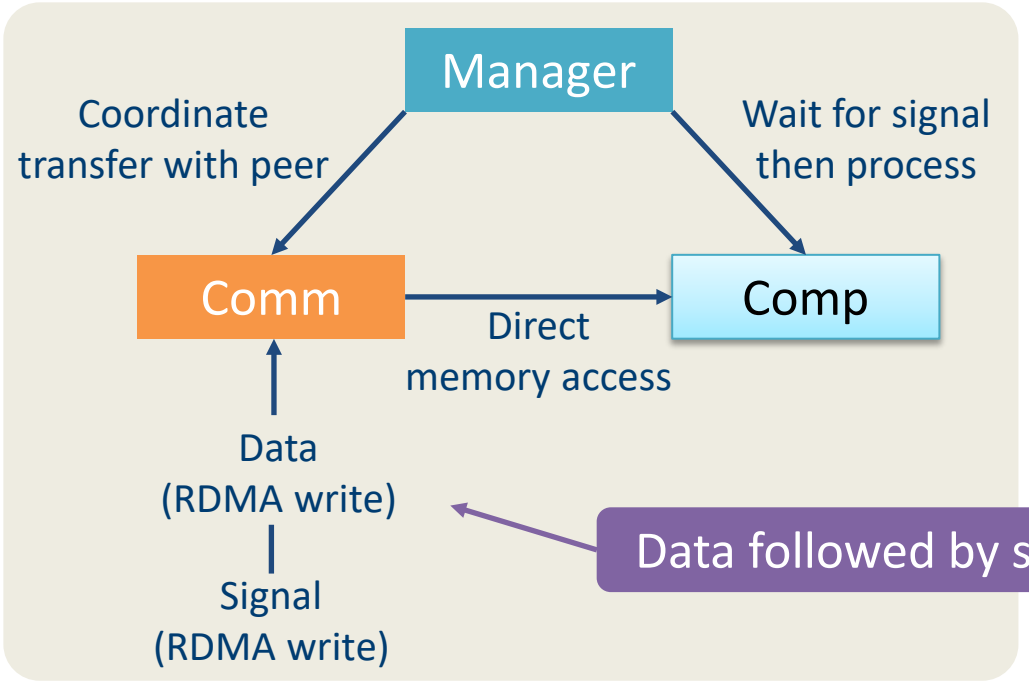
NIC INITIATED COMPUTATION
CONCEPTUAL MODEL



NIC INITIATED COMPUTATION

PHASE I: PUNT!

What could the app do?



Data followed by signal

App writes 'completion' to known location

Use atomic add to receive from multiple peers

- Works with standard and offload NICs
- App could watch multiple signals
- XPU needs ability to poll on local memory change

OTHER THOUGHTS
RESTRICTIONS

- **XPU triggerable endpoint not sharable with host**
  - Increases resources
  - Significant impact for underlying connections
- **Triggers must be signaled in order**
  - *Some* NICs may allow signaling out of order, but will be processed in order
- **Want write-after-write ordering**
  - Or delay writing signal until data transfer complete
- **Defer processing until data from all peers have arrived**
  - Atomic add can help here

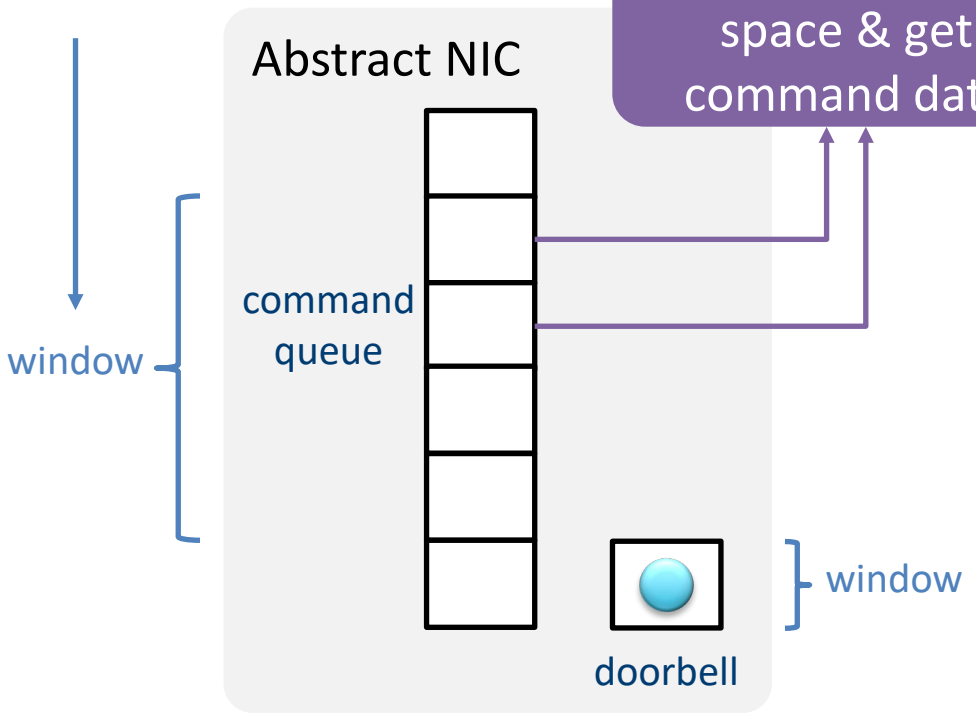
Proposal: application managed command windows

May need new network protocols – write + atomic add

OTHER THOUGHTS
EXPLORATION

Command queue windows

```
struct fi_tx_window {
      void *base;
      size_t length;
};
```



Combine with triggering

```
struct fi_tx_var {
      struct fi_tx_window *window;
      size_t offset;
      size_t window_increment;

      size_t size;
      uint8_t data[];
};
```

App submits in any order

example

```
memcpy(window->base + tx_var->offset,
        tx_var->data, tx_var->size);

window->base += tx_var->window_increment;
```

OTHER THOUGHTS
EXPLORATION

NIC initiated computation

THANK YOU