14th ANNUAL WORKSHOP 2018

NVMF TARGET OFFLOAD

Liran Liss

Mellanox Technologies

April 2018

AGENDA

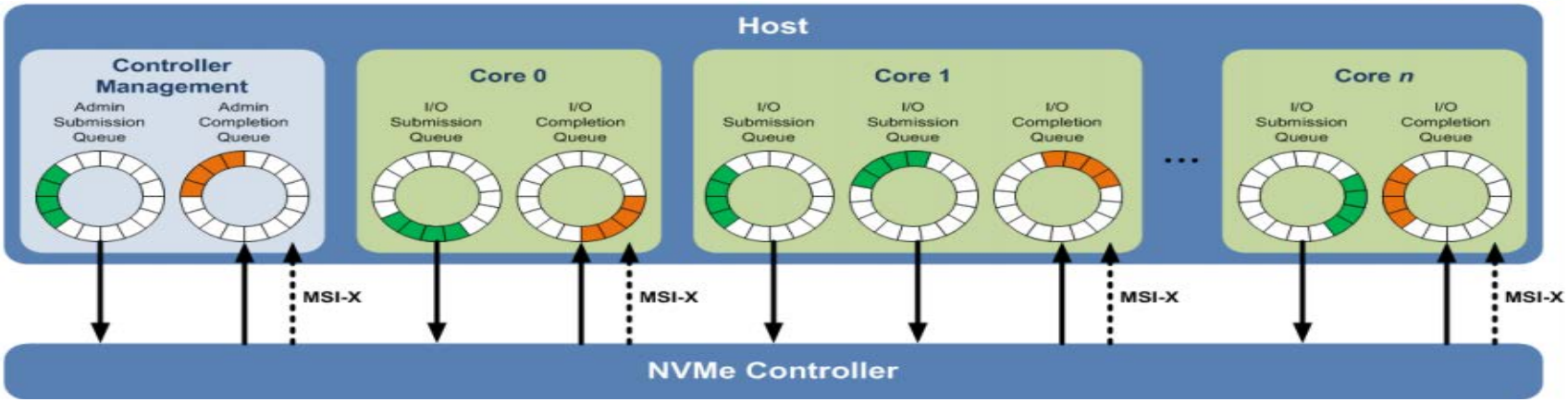
- **Introduction**
  - NVMe
  - NVMf
- **NVMf target driver**
- **Offload model**
- **Verbs interface**
- **Status**

INTRODUCTION

NVME

- **Standard PCIe host controller interface for solid-state storage**
  - Driven by industry consortium of 80+ members
  - Standardize feature, command, and register sets
  - Leverage PCIe capabilities: low latency, scalable BW, power efficiency etc.

- **Focus on efficiency, scalability and performance**
  - All parameters for 4KB command in single 64B DMA fetch
  - Simple command set (13 required commands)
  - MSI-X support



NVME OVER FABRICS

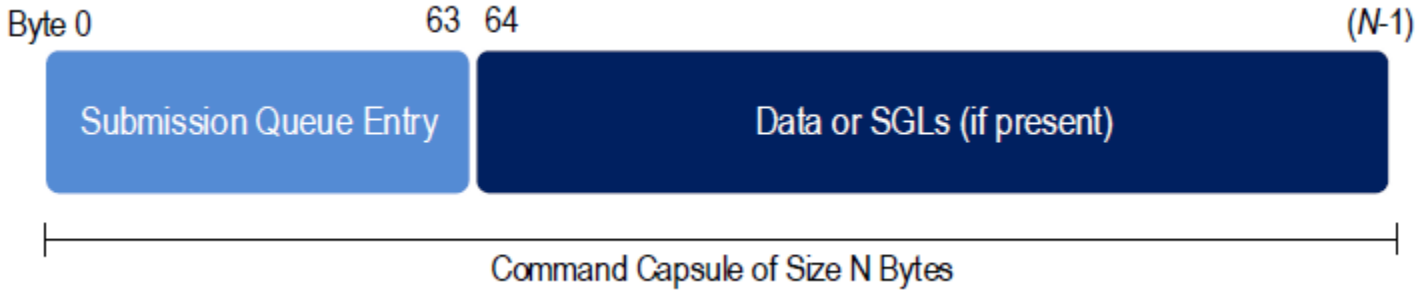
- **Standard NVMe Devices are constrained to the server / storage box**
  - Limited number of PCIe-attached devices
- **PCIe, SAS and SATA are not Scale-Out**
  - Distance limitations
  - Hard to share
  - Complex error handling
- **NVMe Over Fabrics (NVMf)**
  - Announced September 2014
  - Standard V1.0 done, published June 2016
- **NVMf properties**
  - Providers scale-out without requiring SCSI protocol translation
    - Preserves NVMe command set
  - Simplifies storage virtualization, migration, and failover
  - Goal: <10µs added latency compared to local NVMe

NVME MAPPING TO FABRICS

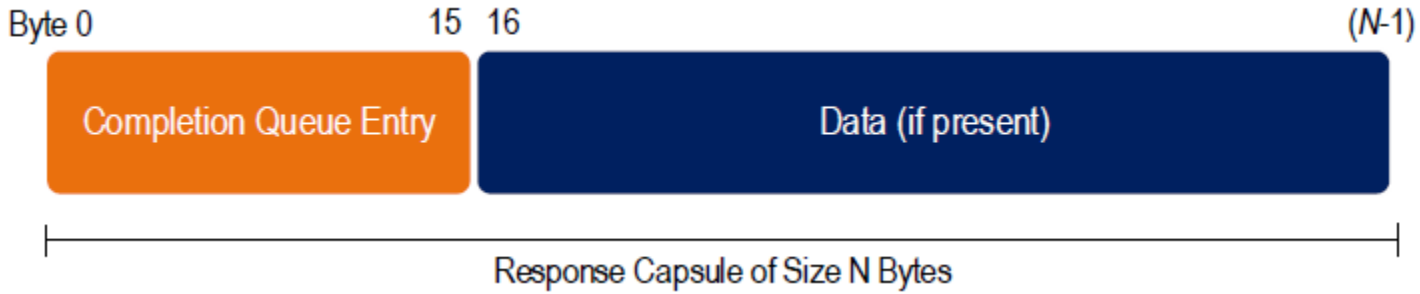
NVMe	NVMf
Submit Queue (SQ)	QP SQ
Completion Queue (CQ)	QP RQ (+CQ)
Host writes SQE and rings doorbell	Initiator sends SQE capsule
Device writes CQE, host awaits and interrupt and polls CQ	Target sends CQE capsule, initiator awaits an interrupt and polls Rx CQ
PCIe data exchange	RDMA Rd/Wr, immediate up to 8K

COMMAND/RESPONSE CAPSULES

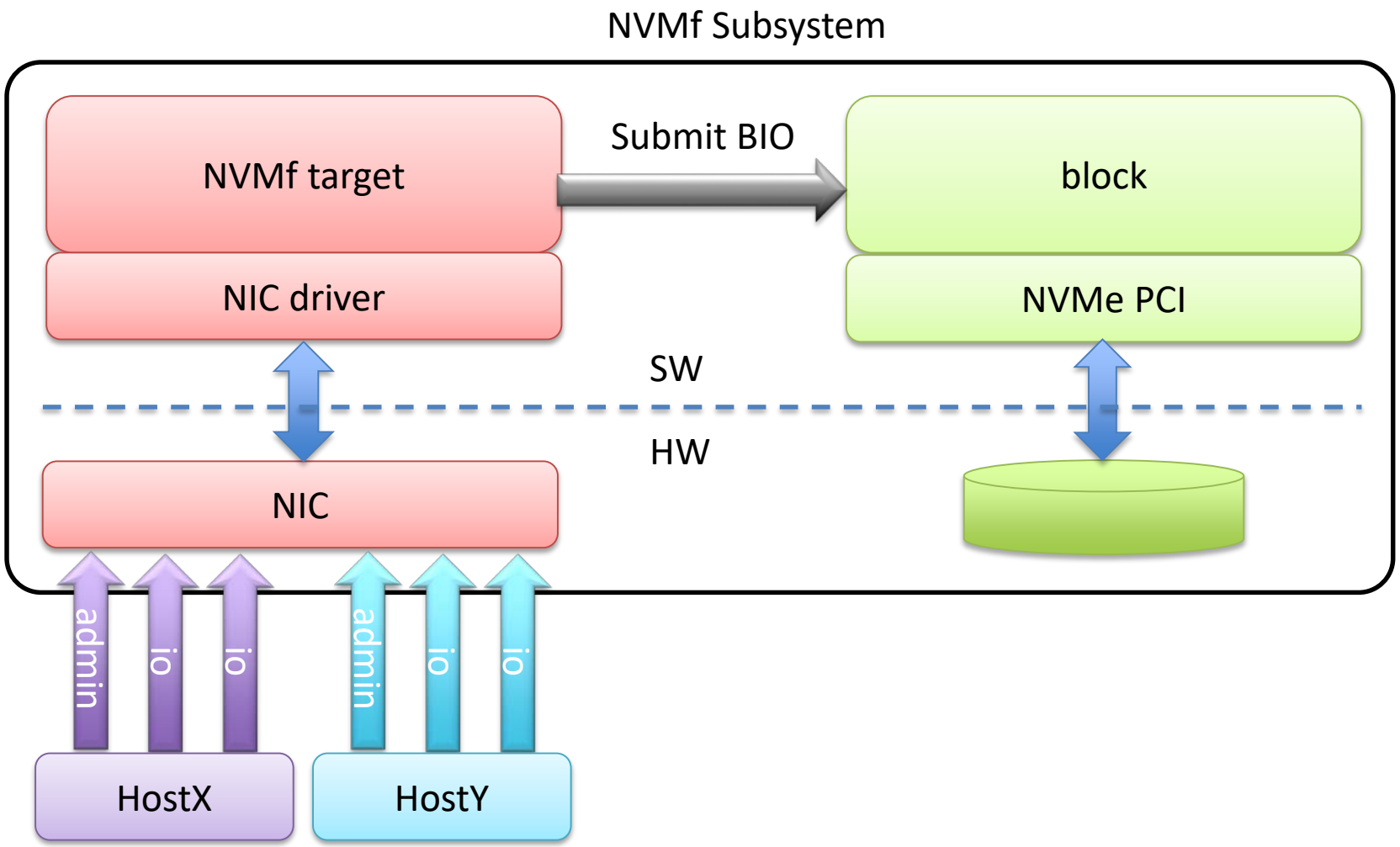
- **Command capsule**



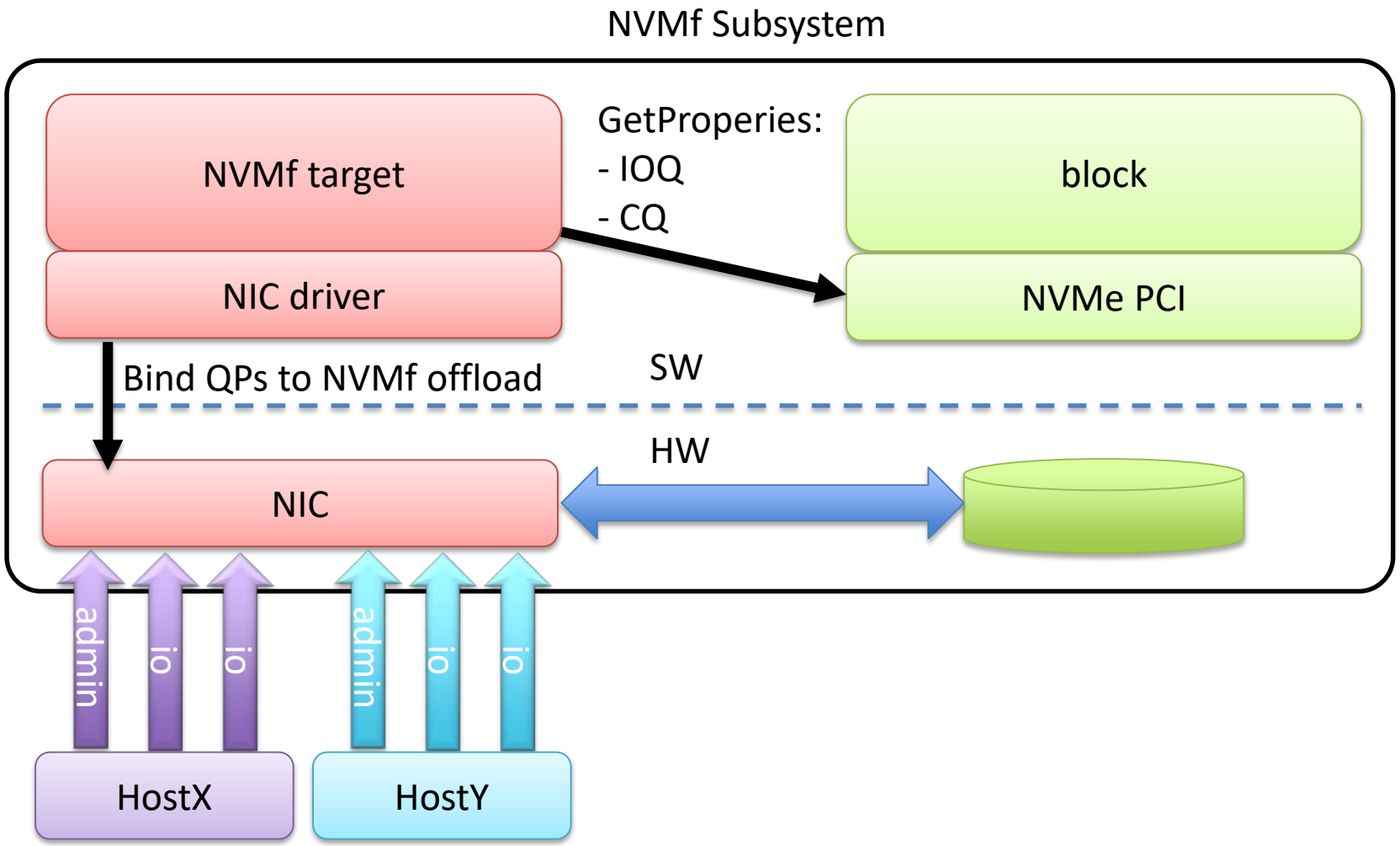
- **Response capsule**



LINUX NVMF TARGET

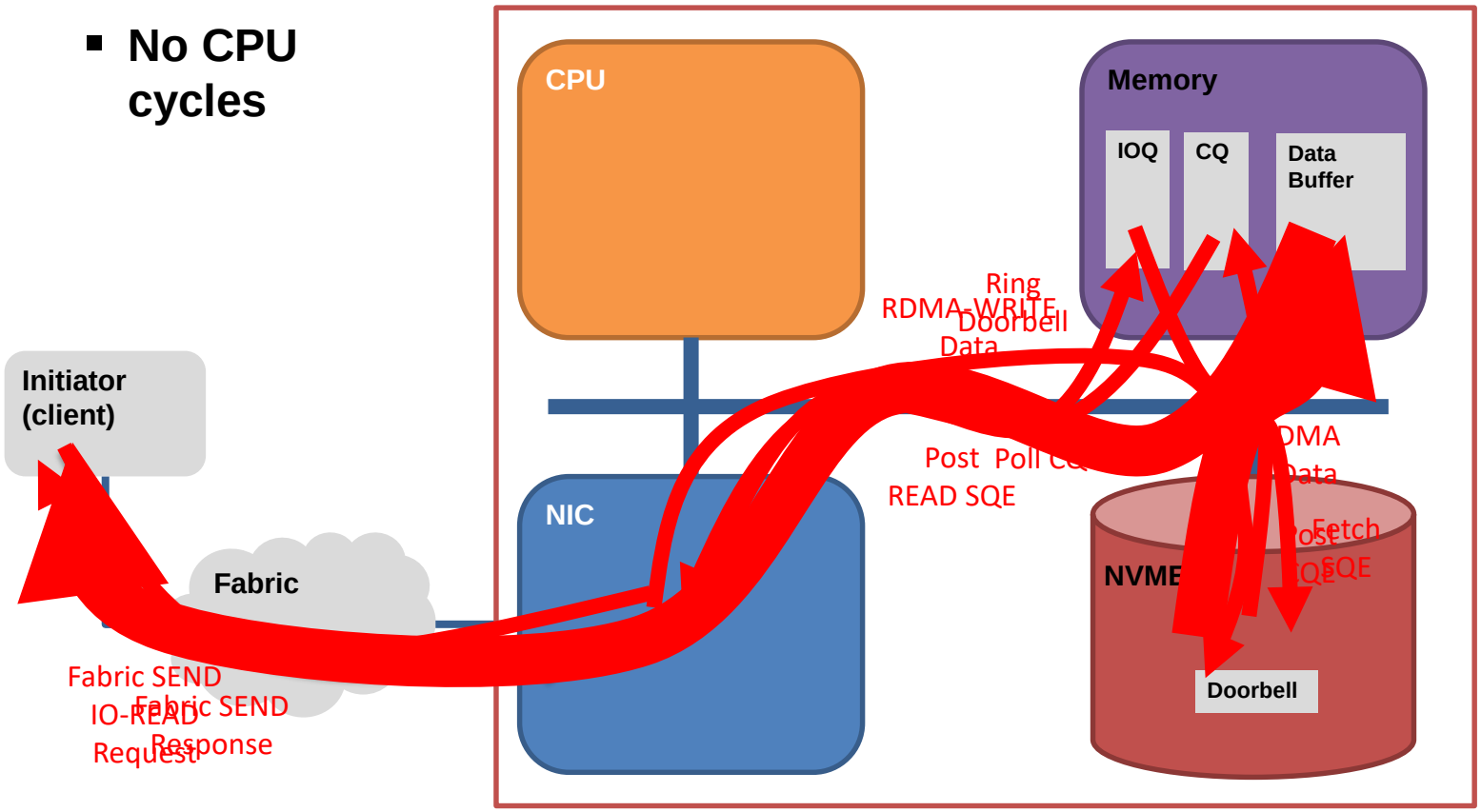


NVMF TARGET OFFLOAD



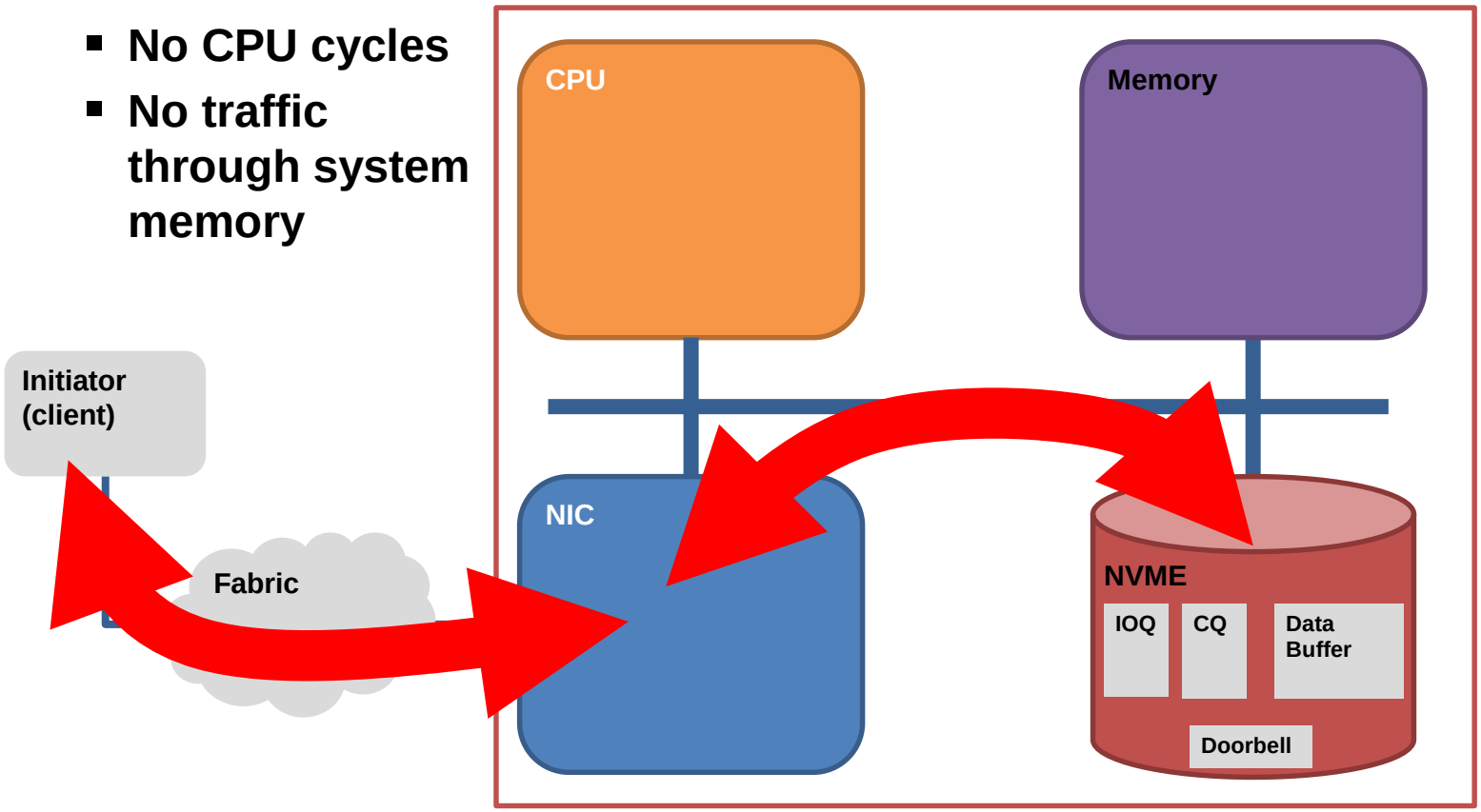
OFFLOAD FLOW

- **No CPU cycles**



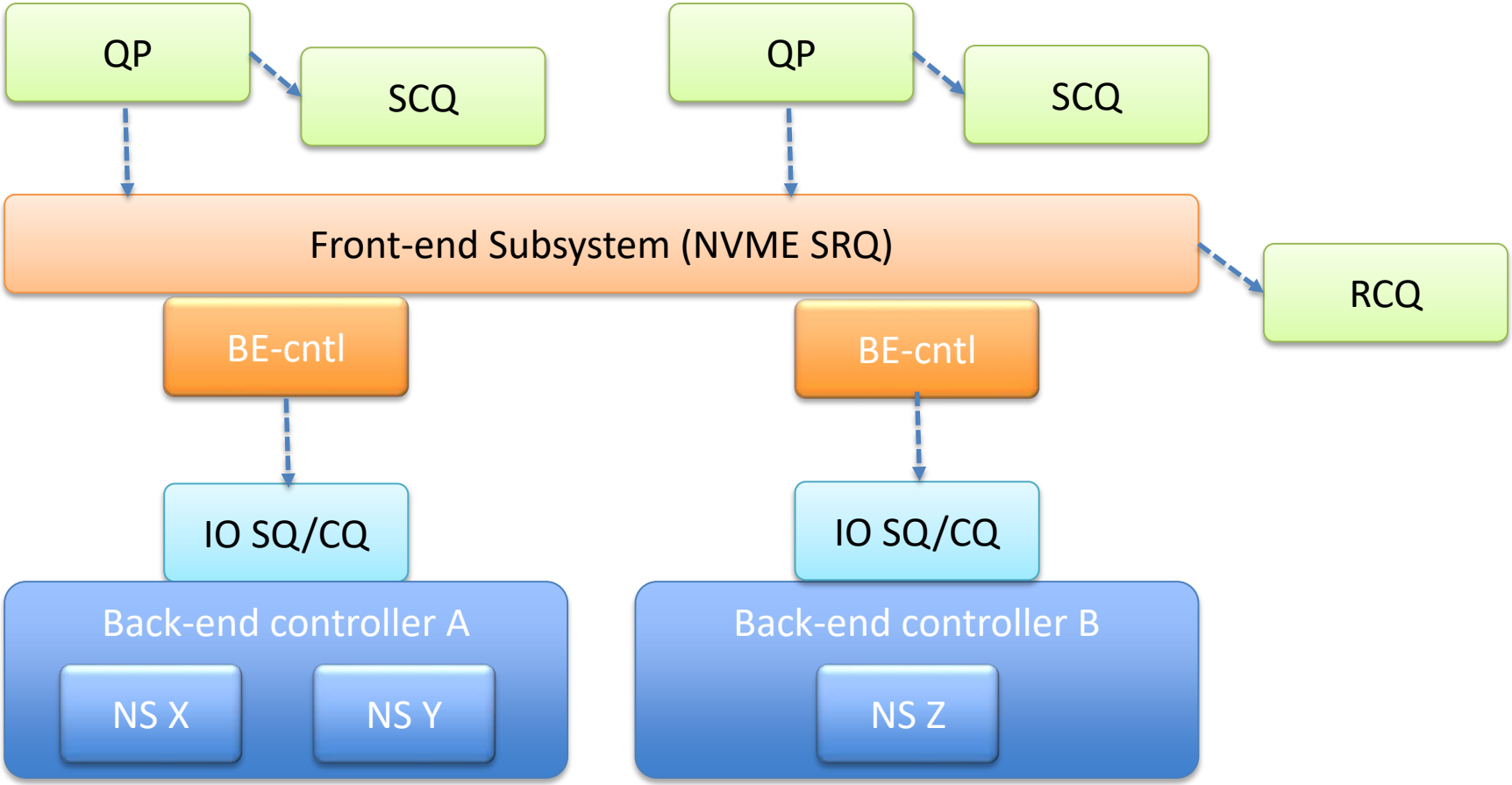
OFFLOAD FLOW – CMB

- No CPU cycles
- No traffic through system memory



NVMF OFFLOAD VERBS

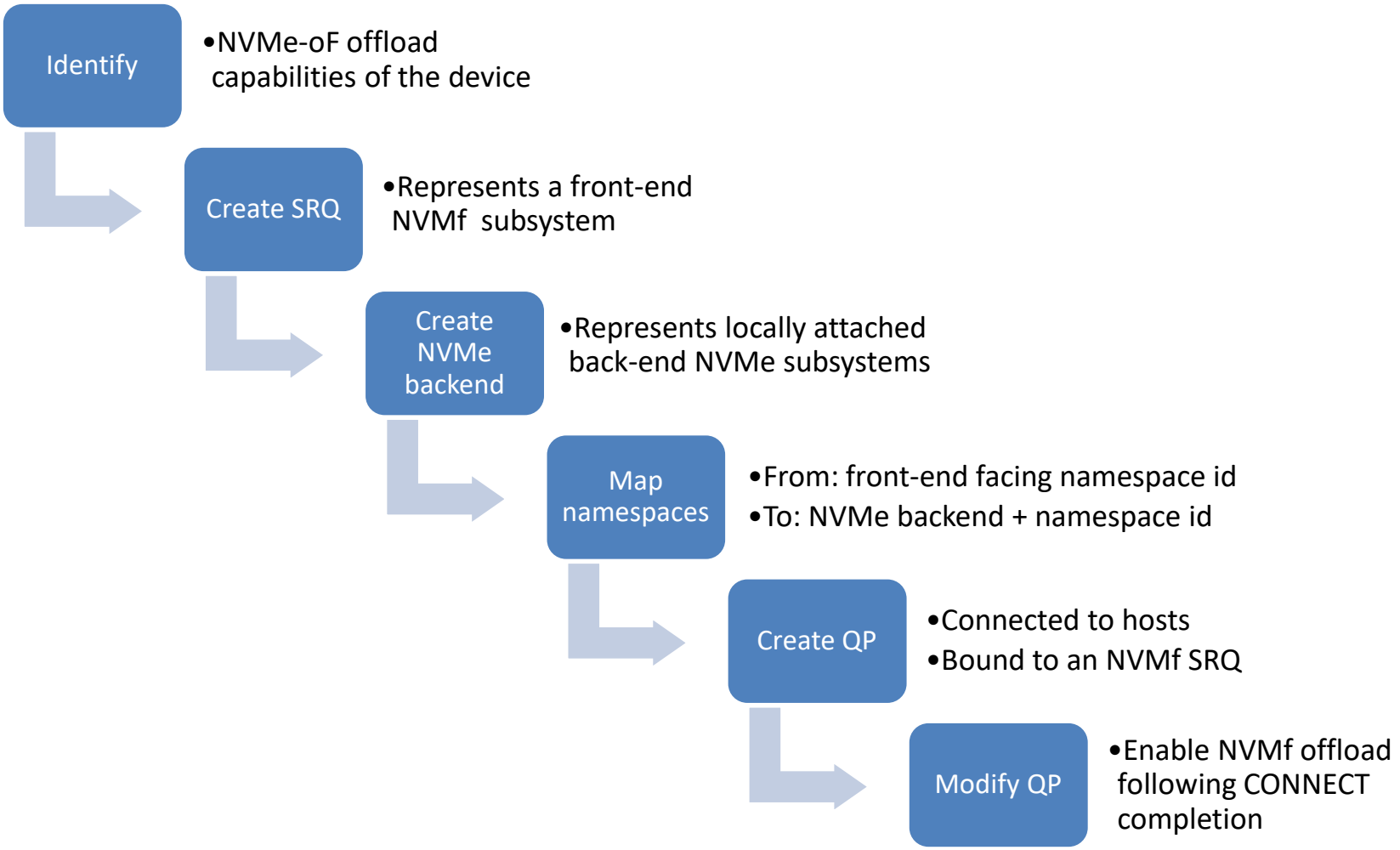
MODEL


QP
SCQ
QP
SCQ
Front-end Subsystem (NVME SRQ)
RCQ
BE-cntl
BE-cntl
IO SQ/CQ
IO SQ/CQ
Back-end controller A
NS X
NS Y
Back-end controller B
NS Z

OBSERVATIONS

- **Offloaded commands are not visible to SW**
  - Do not consume SRQ Rx WQEs
  - Do not generate SRQ completions
  - Do not consume WQEs on QP Send queues
  - Do not generate Send completions
- **Non-offloaded commands follow standard Verbs processing**
  - Consume SRQ Rx WQEs and generate CQEs on the SRQ CQ
  - Responses are posted on corresponding QP Send queues and generate completions
- **The back-end IOQ + CQ are under exclusive HW ownership**
  - SW may use other IOQs/CQs for non-offloaded operations
- **Front-end NSIDs not necessarily equal to back-end NSIDs**

VERBS FLOW


Identify
•NVMe-oF offload capabilities of the device
Create SRQ
•Represents a front-end NVMf subsystem
Create NVMe backend
•Represents locally attached back-end NVMe subsystems
Map namespaces
•From: front-end facing namespace id
•To: NVMe backend + namespace id
Create QP
•Connected to hosts
•Bound to an NVMf SRQ
Modify QP
•Enable NVMf offload following CONNECT completion

NVMF CAPABILITIES: IBV_QUERY_DEVICE_EX()

```
struct ibv_device_attr_ex {
    ...
    struct ibv_nvmf_caps nvmf_caps;
};

struct ibv_nvmf_caps {
    enum nvmf_offload_type offload_type;
    uint32_t        max_backend_ctrls_total;
    uint32_t        max_backend_ctrls;
    uint32_t        max_namespaces;
    uint32_t        max_staging_buf_pages;
    uint32_t        min_staging_buf_pages;
    uint32_t        max_io_sz;
    uint16_t        max_nvme_queue_sz;
    uint16_t        min_nvme_queue_sz;
    uint32_t        max_ioccsz;
    uint32_t        min_ioccsz;
    uint16_t        max_icdoff;
};
```

SUBSYSTEM REPRESENTATION: IBV_CREATE_SRQ_EX()

```
struct ibv_srq_init_attr_ex {
     ...
     struct ibv_nvmf_attrs   nvmf_attr;
};

struct ibv_nvmf_attrs {
     enum nvmf_offload_type  offload_type;
     uint32_t                max_namespaces;
     uint8_t                 nvme_log_page_sz;
     uint32_t                ioccsz;
     uint16_t                icdoff;
     uint32_t                max_io_sz;
     uint16_t                nvme_queue_sz;
     struct ibv_mr           *staging_buf_mr;
     uint64_t                staging_buf_addr;
     uint64_t                staging_buf_len;
};
```

- **SRQ type: IBV_SRQT_NVMF**

NVME BACKEND ASSOCIATION: IBV_SRQ_CREATE_NVME_CTRL()

```
struct ibv_mr_sg {
    struct ibv_mr *mr;
    union {
        void     *addr;
        uint64_t offset;
    };
    uint64_t    len;
};

struct nvme_ctrl_attrs {
    struct ibv_mr_sg      sq_buf;
    struct ibv_mr_sg      cq_buf;

    struct ibv_mr_sg      sqdb;
    struct ibv_mr_sg      cqdb;
    uint16_t              sqdb_ini;
    uint16_t              cqdb_ini;

    uint16_t              timeout_ms;
    uint32_t              comp_mask;
};
```

NVME BACKEND ASSOCIATION (CONT.)

```
struct ibv_nvme_ctrl *ibv_srq_create_nvme_ctrl(
    struct ibv_srq *srq,
    struct nvme_ctrl_attrs *nvme_attrs);

int ibv_srq_remove_nvme_ctrl(
    struct ibv_srq *srq,
    struct ibv_nvme_ctrl *nvme_ctrl);
```

MAP NAMESPACES: IBV_MAP_NVMF_NSID()

```
int ibv_map_nvmf_nsid(
    struct ibv_nvme_ctrl *nvme_ctrl,
    uint32_t fe_nsid,
    uint16_t lba_data_size,
    uint32_t nvme_nsid);

int ibv_unmap_nvmf_nsid(
    struct ibv_nvme_ctrl *nvme_ctrl,
    uint32_t fe_nsid);
```

- **Map <fe_nsid> to <nvme_ctrl, nvme_nsid>**
  - Front-end subsystem determined by the SRQ associated with the nmve_ctrl
- **Indicate the namespace LBA Data Size**

ENABLE OFFLOAD: IBV_QP_SET_NVMF()

```
enum {
     IBV_QP_NVMF_ATTR_FLAG_ENABLE = 1 << 0,
};

int ibv_modify_qp_nvmf(
          struct ibv_qp *qp,
          int flags);
```

- **Enables NVMf offload on the given QP**
- **A QP should be enabled for offload only after processing the CONNECT fabric command**

EXCEPTIONS

- **Transport error**
  - QP transitions to error state raising IBV_EVENT_QP_FATAL async event

- **NVMe errors**
  - Reported as ‘nmve_ctrl’ async events

```
enum ibv_event_type {
    ...
    IBV_EVENT_NVME_PCI_ERR,
    IBV_EVENT_NVME_TIMEOUT,
    ...
};

struct ibv_async_event {
    union {
        ...
        struct ibv_nvme_ctrl *nvme_ctrl
    } element;
    ...
};
```

STATUS

- **Submitted RFC for user-space Verbs**
  - https://www.spinics.net/lists/linux-rdma/msg58512.html

- **Added/Updated files**
  - `Documentation/nvmf_offload.md | 172`
  - `libibverbs/man/ibv_create_srq_ex.3 | 48`
  - `libibverbs/man/ibv_get_async_event.3 | 15`
  - `libibverbs/man/ibv_map_nvmf_nsid.3 | 89`
  - `libibverbs/man/ibv_qp_set_nvmf.3 | 53`
  - `libibverbs/man/ibv_query_device_ex.3 | 26`
  - `libibverbs/man/ibv_srq_create_nvme_ctrl.3 | 89`
  - `libibverbs/verbs.h | 107`

- **Kernel discussions ongoing**

14th ANNUAL WORKSHOP 2018

THANK YOU

Liran Liss

Mellanox Technologies