SNIA DSI '16 DATA STORAGE INNOVATION CONFERENCE
Innovation in Storage Products, Services, and Solutions
June 13-15, 2016 | Marriott San Mateo | San Mateo, CA

Under the Hood with NVMe over Fabrics

J Metz, Ph.D
R&D Engineer, Advanced Storage
Cisco Systems, Inc.
@drjmetz

Agenda

- NVM Express Ecosystem
- Base NVMe Architecture
- NVMe Communication Model
- NVMe over Fabric Deployment Models

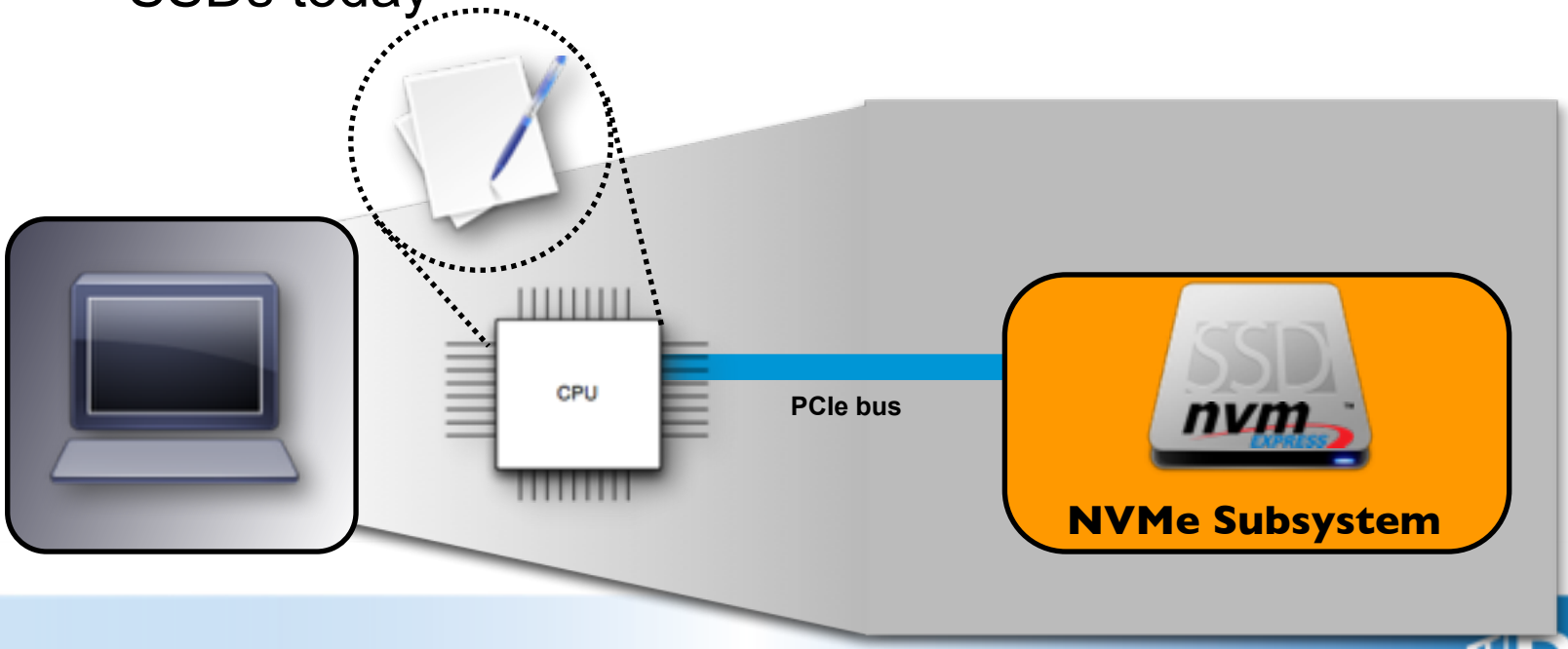
NVM Express Ecosystem
The Big Picture

What is NVM Express™?

- Industry standard for PCIe SSDs
  - High-performance, low-latency, PCIe SSD interface
    - Command set + PCIe register interface
  - In-box NVMe host drivers for Linux, Windows, VmWare, …
  - Standard h/w drive form factors, mobile to enterprise
- NVMe community is 80+ companies strong and growing
  - Learn more at nvmexpress.org

Snapshot of NVM Express Version 1.2

- Non-Volatile Memory Express (NVMe) began as an industry standard solution for efficient PCIe attached non-volatile memory storage (e.g., NVMe PCIe SSDs)
  - Low latency and high IOPS direct-attached NVM storage
  - Multiple companies shipping and deploying NVMe PCIe SSDs today



Expanding NVMe to Fabrics

- Built on common NVMe architecture with additional definitions to support message-based NVMe operations
- Standardization of NVMe over a range Fabric types
  - Initial fabrics; RDMA(RoCE, iWARP,InfiniBand™) and Fibre Channel
  - First release candidate specification in early 2016
  - NVMe.org Fabrics Linux Driver WG developing host and target drivers

NVMe Over Fabrics Philosophy

- End-to-End NVMe semantics across a range of topologies
  - Retains NVMe efficiency and performance over network fabrics
  - Eliminates unnecessary protocol translations
  - Enables low-latency and high IOPS to remote NVMe storage solutions

Yet *Another* Storage Protocol?

- May it please the court…
  - NVMe SSD technology has moved the bottleneck from the drive to the network
  - We'll show how NVMe over Fabrics extends efficiencies in local storage across a network
- Therefore
  - NVMe and NVMe over Fabrics (NVMeoF) is the right solution to this problem

NVMe Base Architecture

In This Section...

- NVMe Base Architecture Elements
- NVMe Implementation Examples

NVMe Enabled Host

NVMe Communications

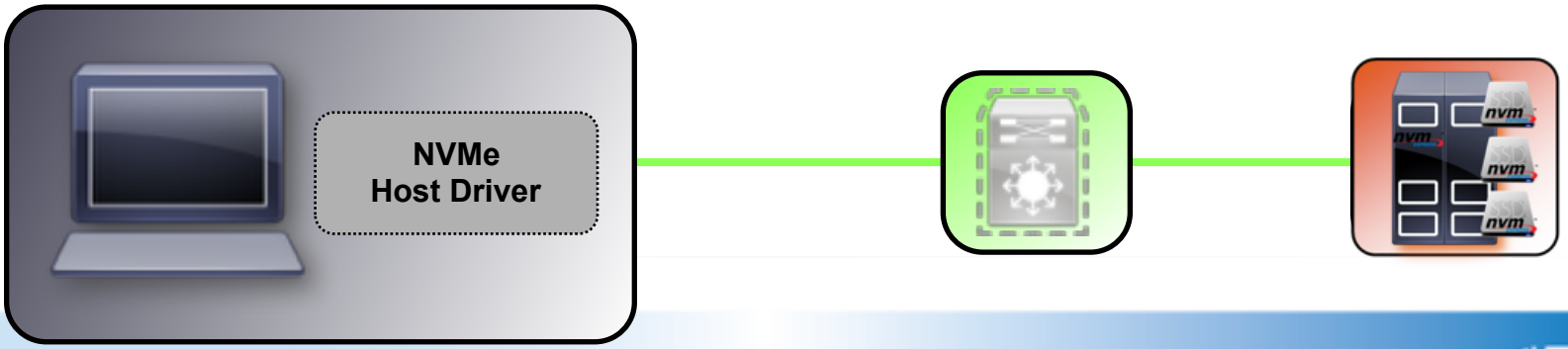
NVMe Storage Subsystem

What you need - NVMe Enabled Host

The Host is the consumer of NVMe Storage

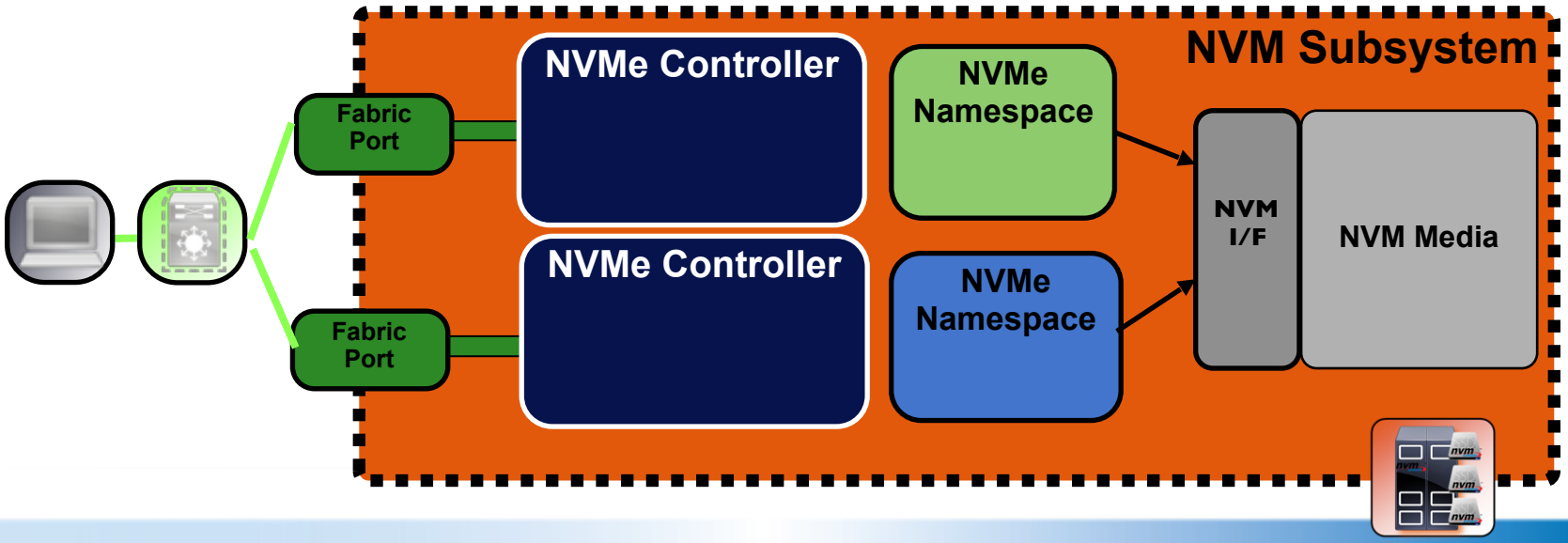
Windows*	• Windows* 8.1 and Windows* Server 2012 R2 include inbox driver • Open source driver in collaboration with OFA
Linux*	• Native OS driver since Linux* 3.3 (Jan 2012)
Unix	• FreeBSD driver released
Solaris*	• Delivered to S12 and S11 Update2 • Compliant with 1.0e
VMware*	• vmklinux driver certified targeted for Q2 '14 release
UEFI	• Open source driver available on SourceForge

- NVMe Host Drivers
  - In-box PCIe NVMe drivers in all major operating systems
  - Driver provides streamlined interface between the O/S storage stack and NVMe SSDs
  - NVMe.org Linux Fabric Driver WG developing multi-fabric NVMe host driver



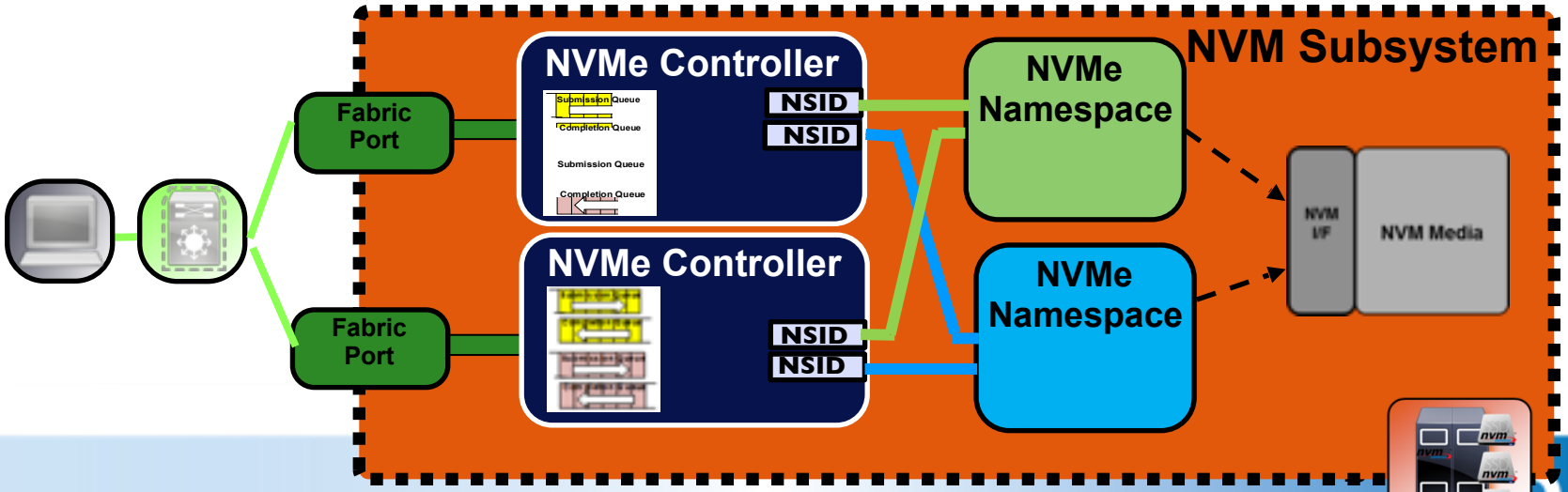
What you need - NVM Subsystem

- Architectural Elements
  - NVMe Controllers
  - NVMe Namespaces
  - Fabric Ports
- Implementation Dependent Elements
  - NVM Media and Interface



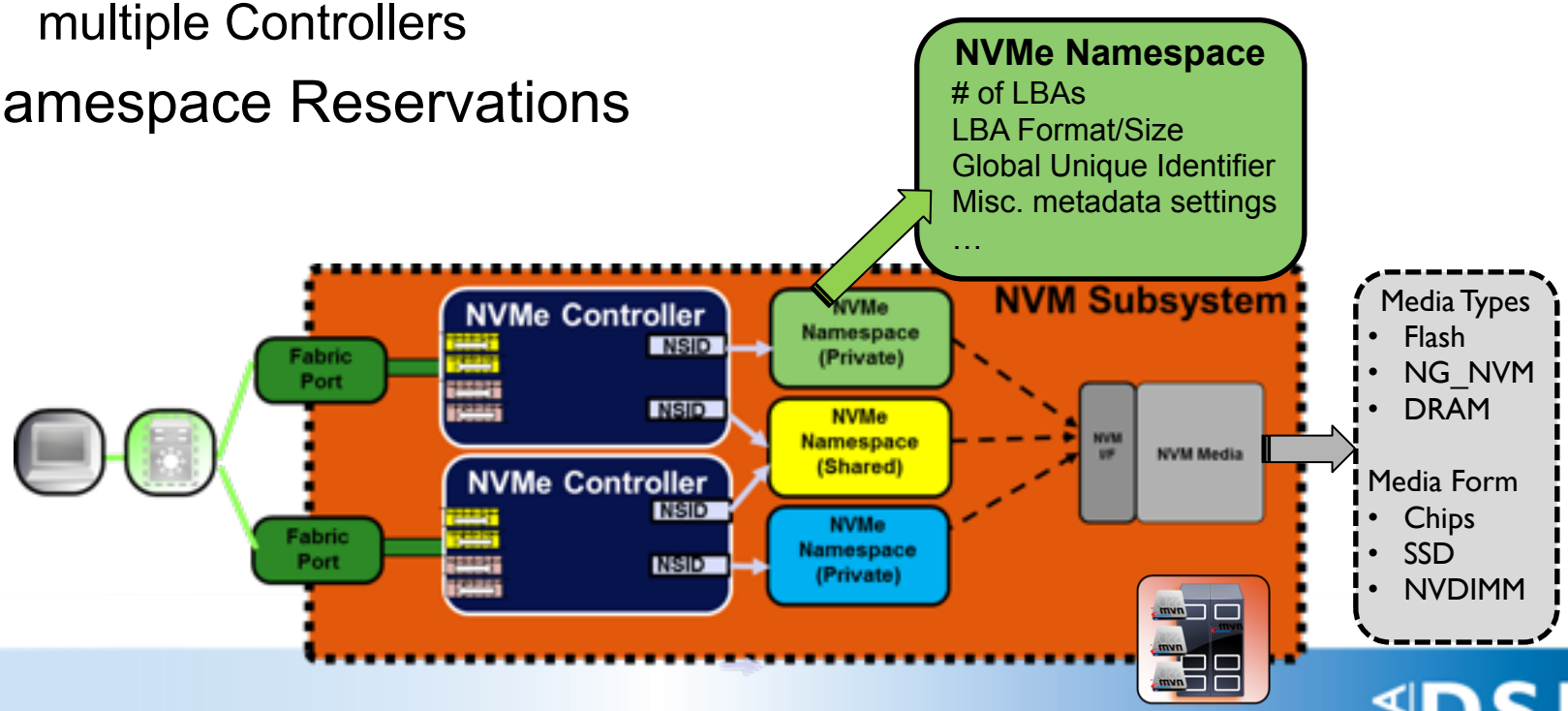
NVMe Controller

- NVMe Command Processing
- Access to NVMe Namespaces
  - Namespace ID (NSID) associates a Controller to Namespaces(s)
- May have multiple Controllers per NVM Subsystem
  - Used in multi-host and multi-path configurations
- NVMe Queue Host Interface
  - Paired Command Submission and Completion Queues
  - Admin Queue for configuration, Scalable number of IO Queues



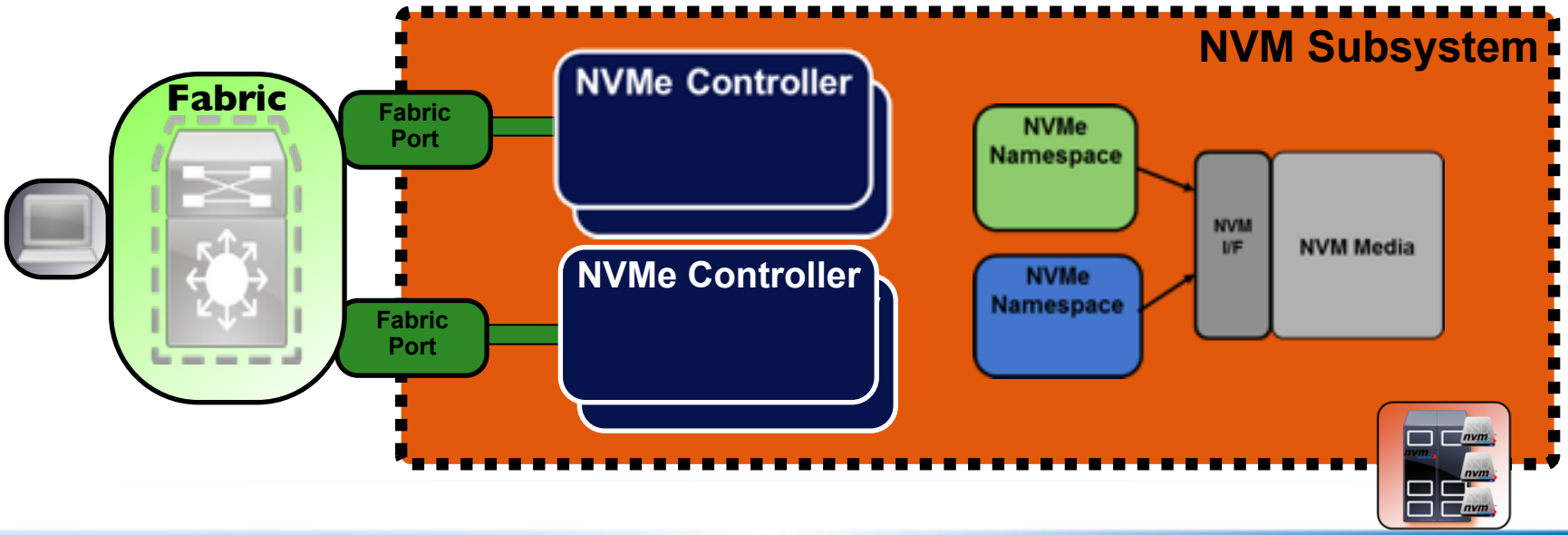
NVMe Namespaces and NVM Media

- Defines the mapping of NVM Media to a formatted LBA range
  - Multiple formats supported with/without end-to-end protection
  - NVM Subsystem may have multiple Namespaces
- Private or Shared Namespaces
  - Private is accessible by one Controller, Shared accessible by multiple Controllers
- Namespace Reservations



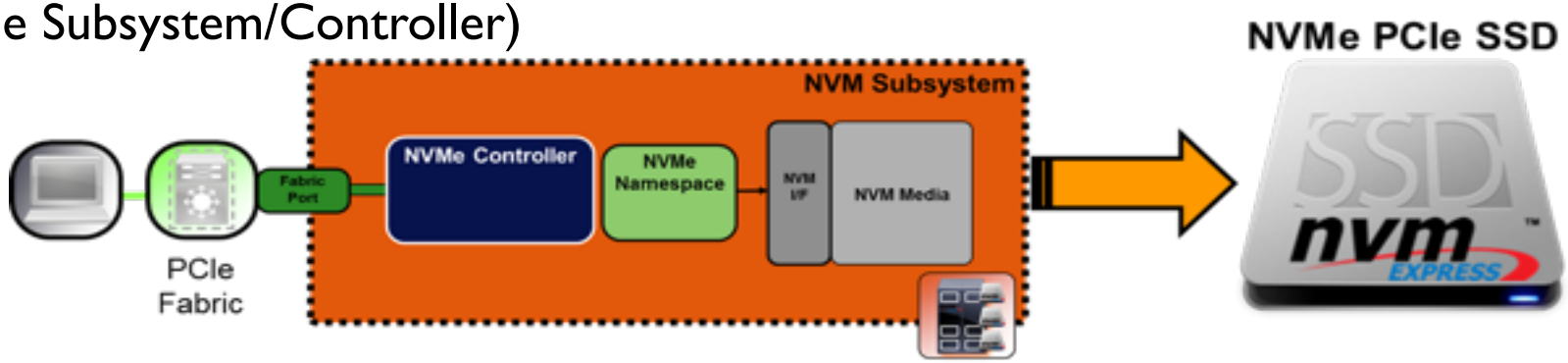
Fabric Ports

- Subsystem Ports are associated with Physical Fabric Ports
- Multiple NVMe Controllers may be accessed through a single port
- NVMe Controllers are associated with one port
- Fabric Types; PCIe, RDMA (Ethernet RoCE/iWARP, InfiniBand™), Fibre Channel/FCoE



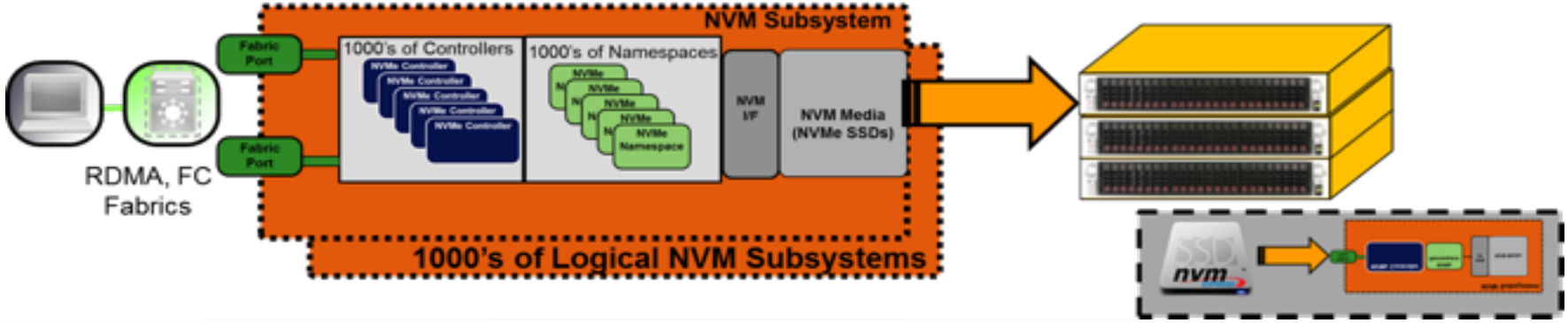
NVMe Subsystem Implementations

NVMe PCIe SSD Implementation
(single Subsystem/Controller)



NVMe all NVM Storage Appliance Implementation
(1000's of Subsystems/Controllers)

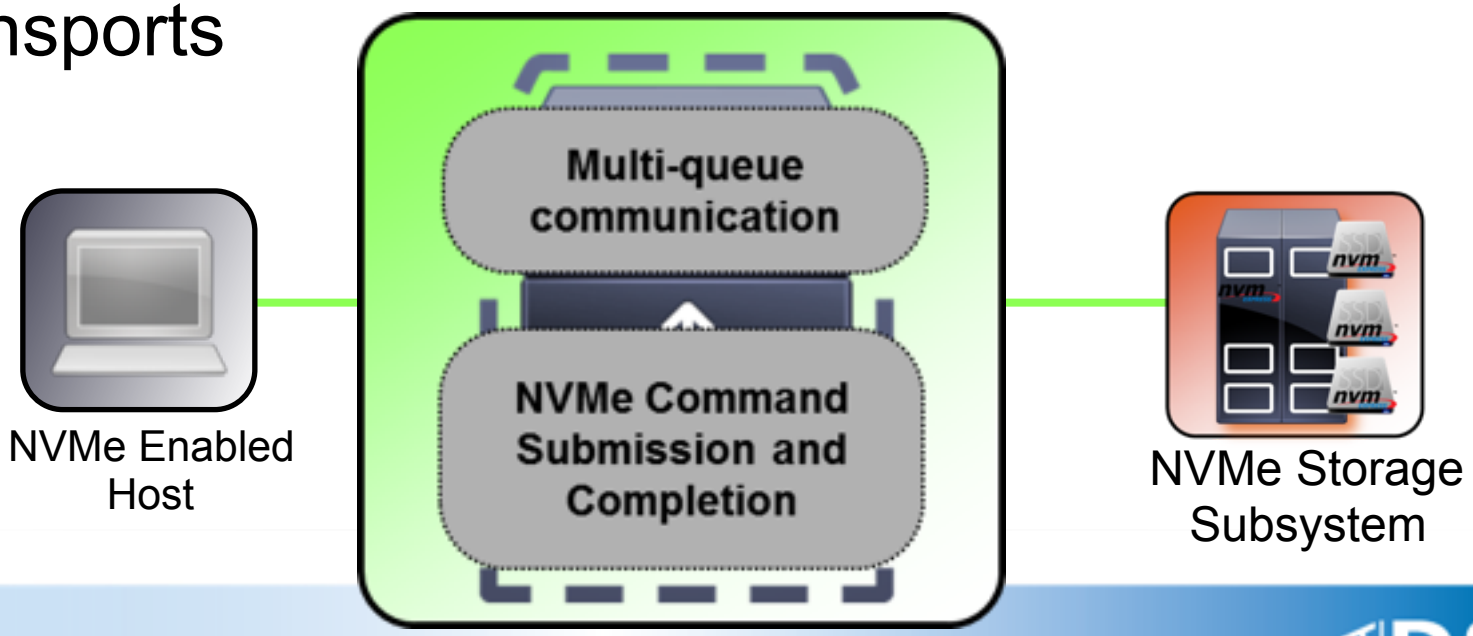
All NVM Appliance with PCIe NVMe SSDs



NVMe Host to Controller Communications

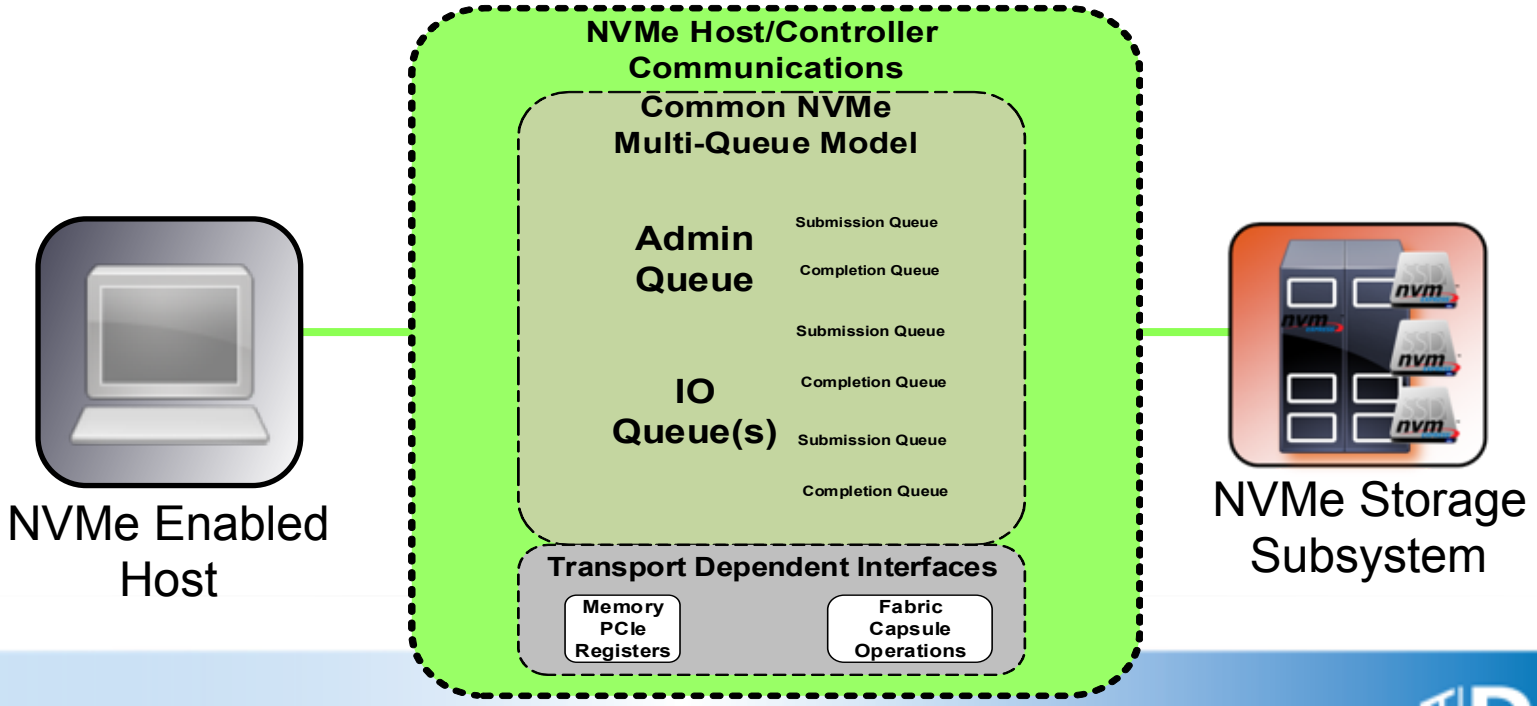
In this Section, …

- NVMe Host/Controller Communications
  - Command Submission and Completion
  - NVMe Multi-Queue Model
  - Command Data Transfers
- NVMe communications over multiple fabric transports



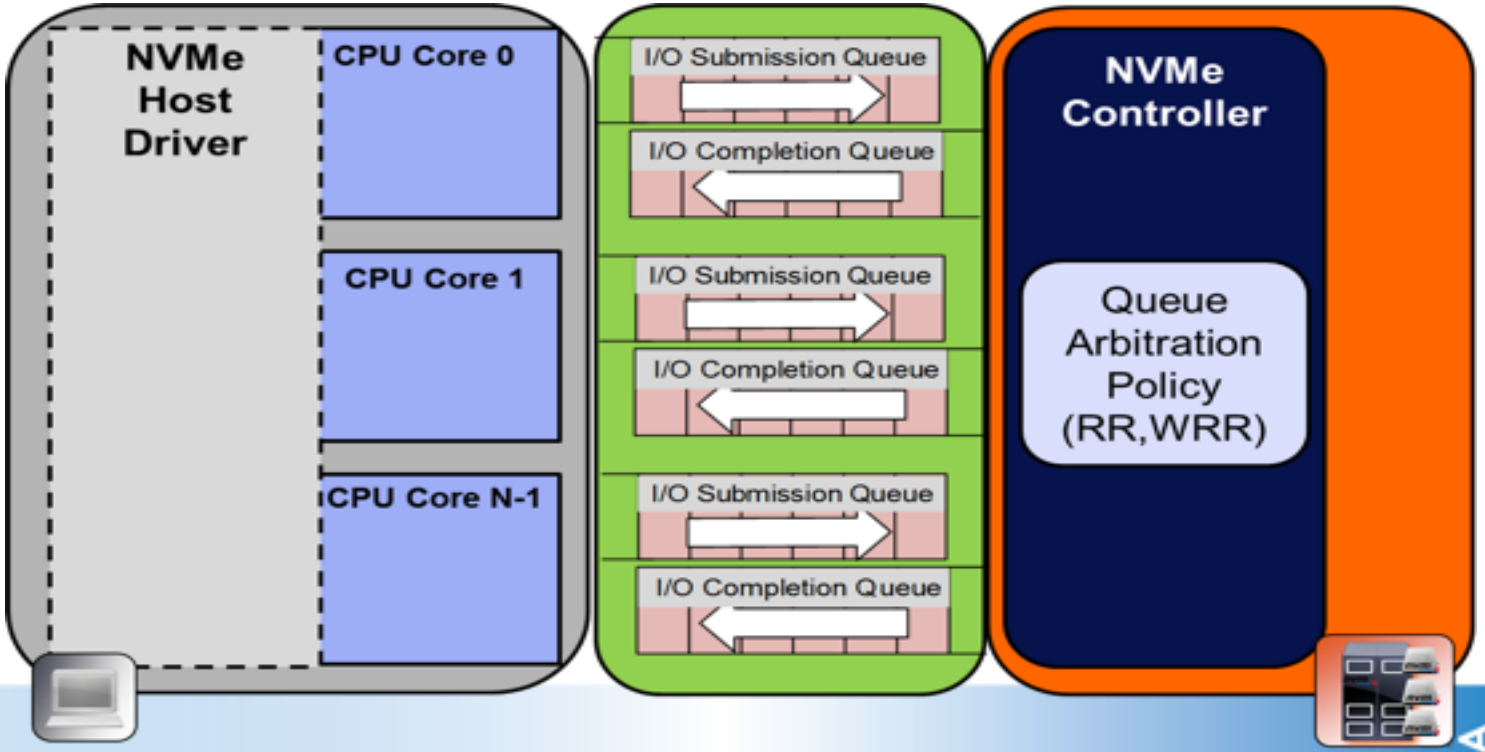
NVMe Host/Controller Communications

- NVMe Multi-Queue Interface Model
  - Single Administrative and Multiple IO Queues
  - Host sends NVMe Commands over the Submission Queue (SQ)
  - Controller sends NVMe Completions over a paired Completion Queue (CQ)
  - Transport type dependent interfaces facilitate the queue operations and NVMe Command Data transfers



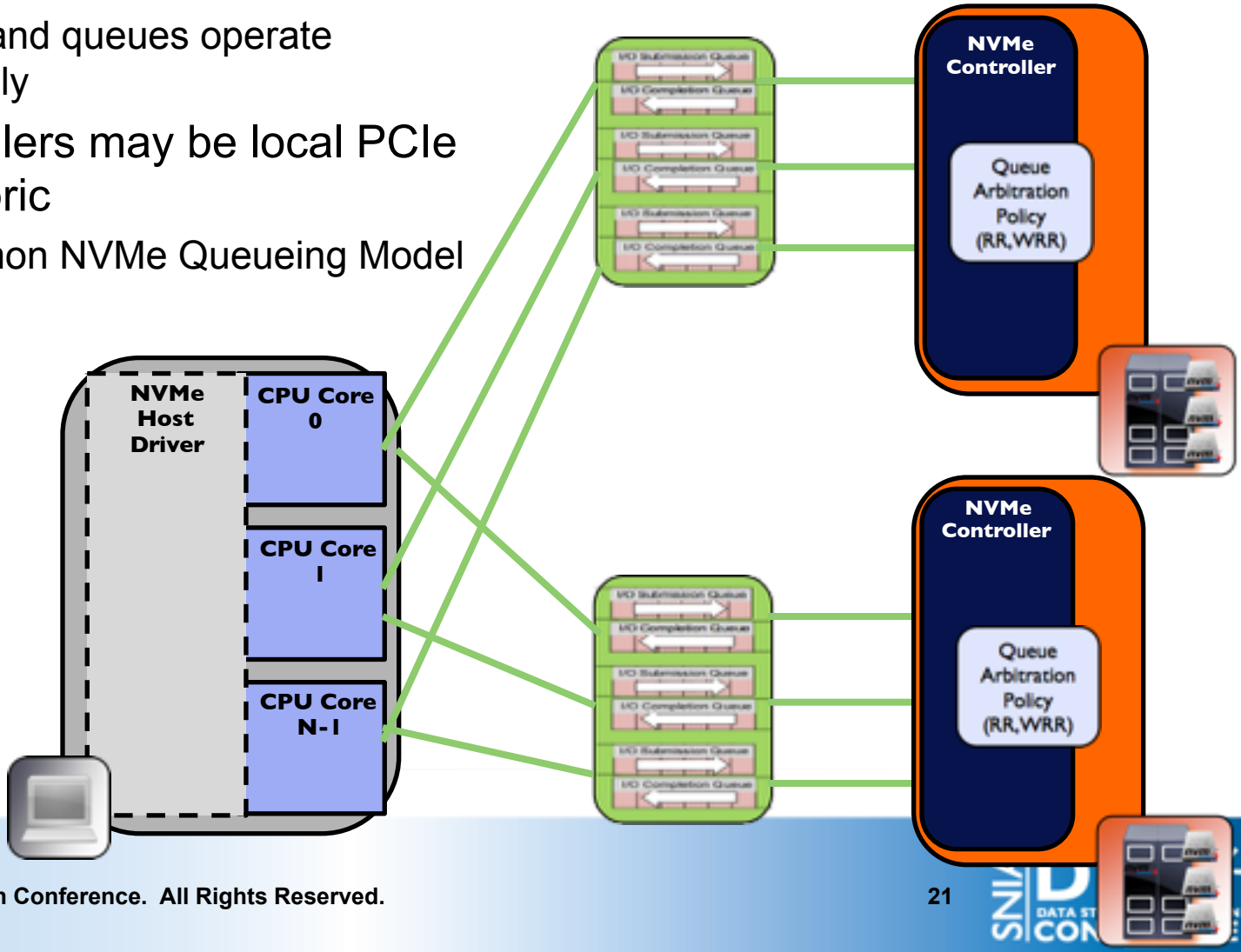
NVMe Multi-Queue Interface

- I/O Submission and Completion Queue Pairs are aligned to Host CPU Cores
  - Independent per queue operations
  - No inter-CPU locks on command Submission or Completion
  - Per Completion Queue Interrupts enables source core interrupt steering



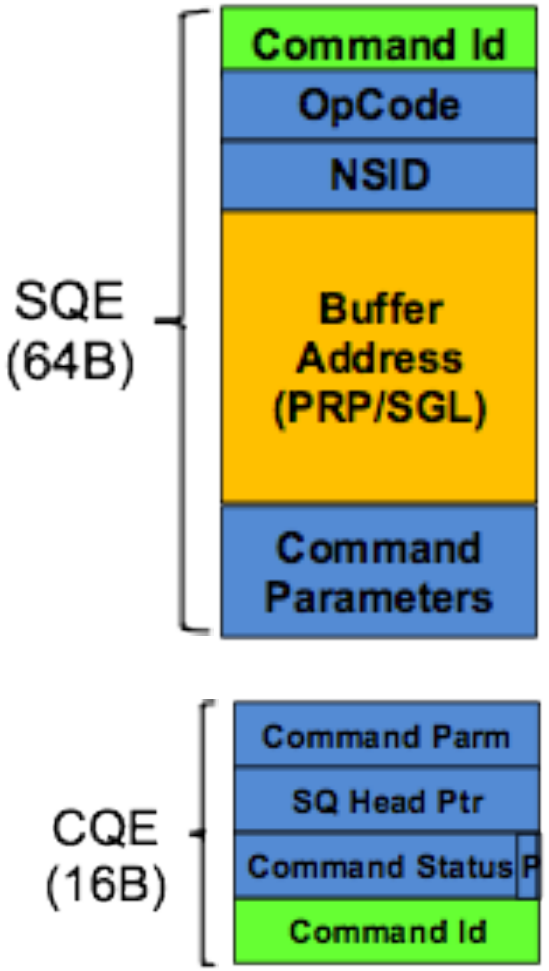
Queues Scale With Controllers

- Each Host/Controller pair have an independent set of NVMe queues
  - Controllers and queues operate autonomously
- NVMe Controllers may be local PCIe or remote Fabric
  - Use a common NVMe Queueing Model

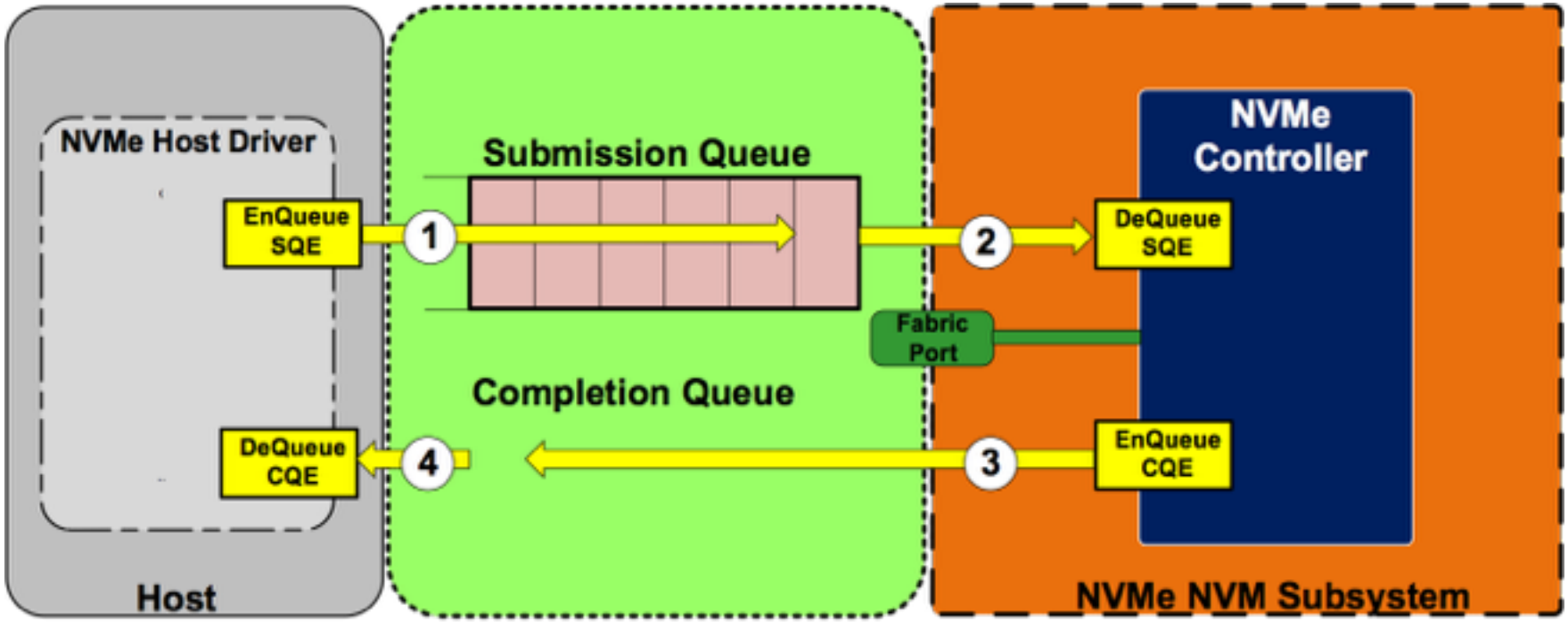


NVMe Commands and Completions

- NVMe Commands are sent by the Host to the Controller in Submission Queue Entries (SQE)
  - Separate Admin and IO Commands
  - Three mandatory IO Commands
  - Added two fabric-only Commands
  - Commands may complete out of order
- NVMe Completions are sent by the Controller to the Host in Completion Queue Entries (CQE)
  - **Command Id** identifies the completed command
  - **SQ Head Ptr** indicates the consumed SQE slots that are available for posting new SQEs

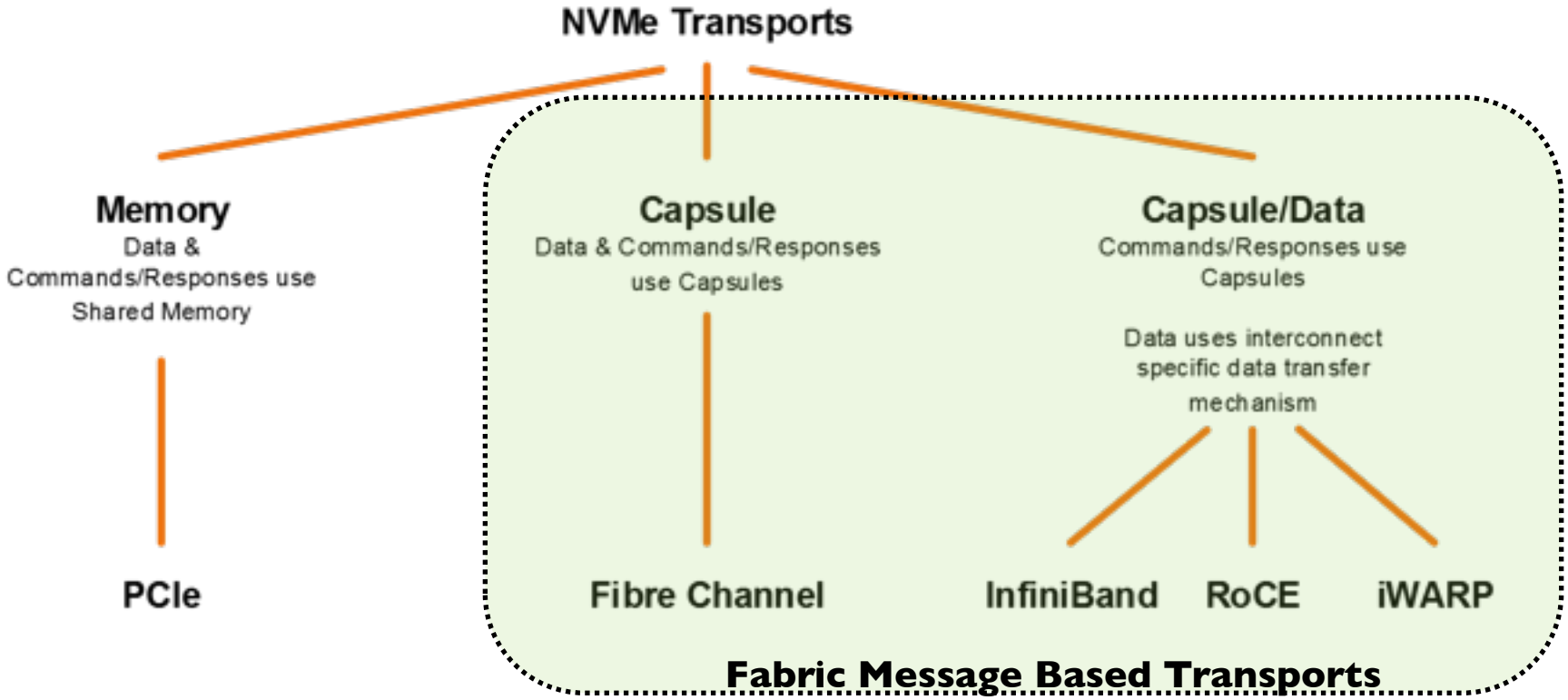


NVMe Queuing Operational Model



- 1. Host Driver enqueues the SQE into the SQ
- 2. NVMe Controller dequeues SQE
- 3. NVMe Controller enqueues CQE into the CQ
- 4. Host Driver dequeues CQE

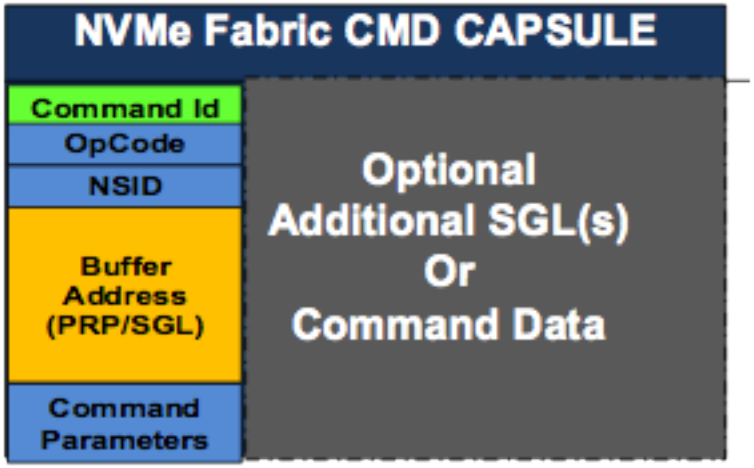
NVMe Multi-Fabric Transport Mapping



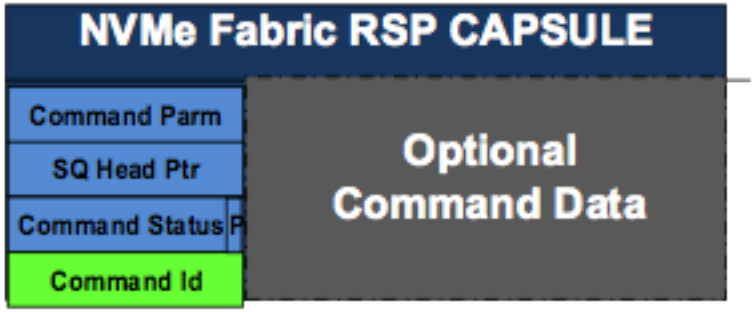
Capsule = Encapsulated NVMe Command/Completion within a transport message

Data = Transport data exchange mechanism (if any)

NVMe over Fabrics Capsules



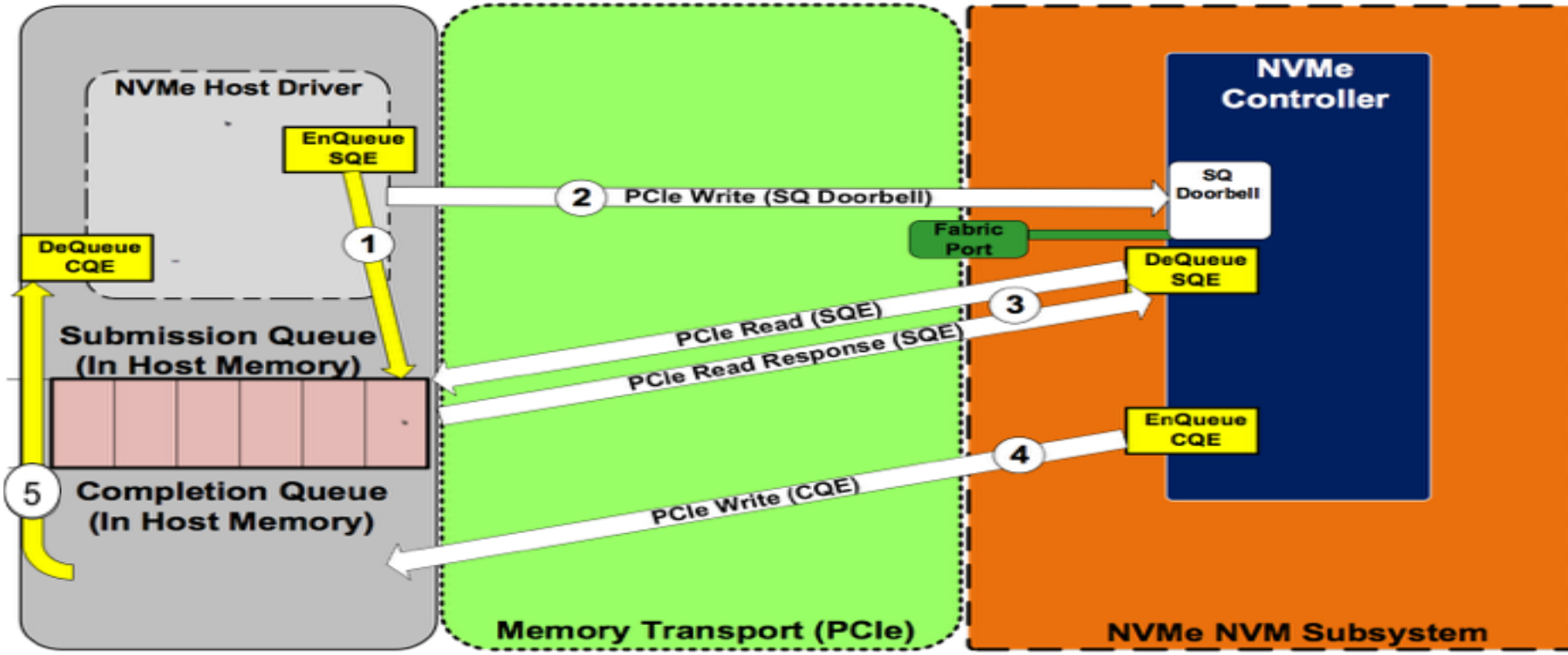
- NVMe over Fabric Command Capsule
  - Encapsulated NVMe SQE Entry
  - May contain additional Scatter Gather Lists (SGL) or NVMe Command Data
  - Transport agnostic Capsule format



- NVMe over Fabric Response Capsule
  - Encapsulated NVMe CQE Entry
  - May contain NVMe Command Data
  - Transport agnostic Capsule format

NVMe Queuing on Memory Transport

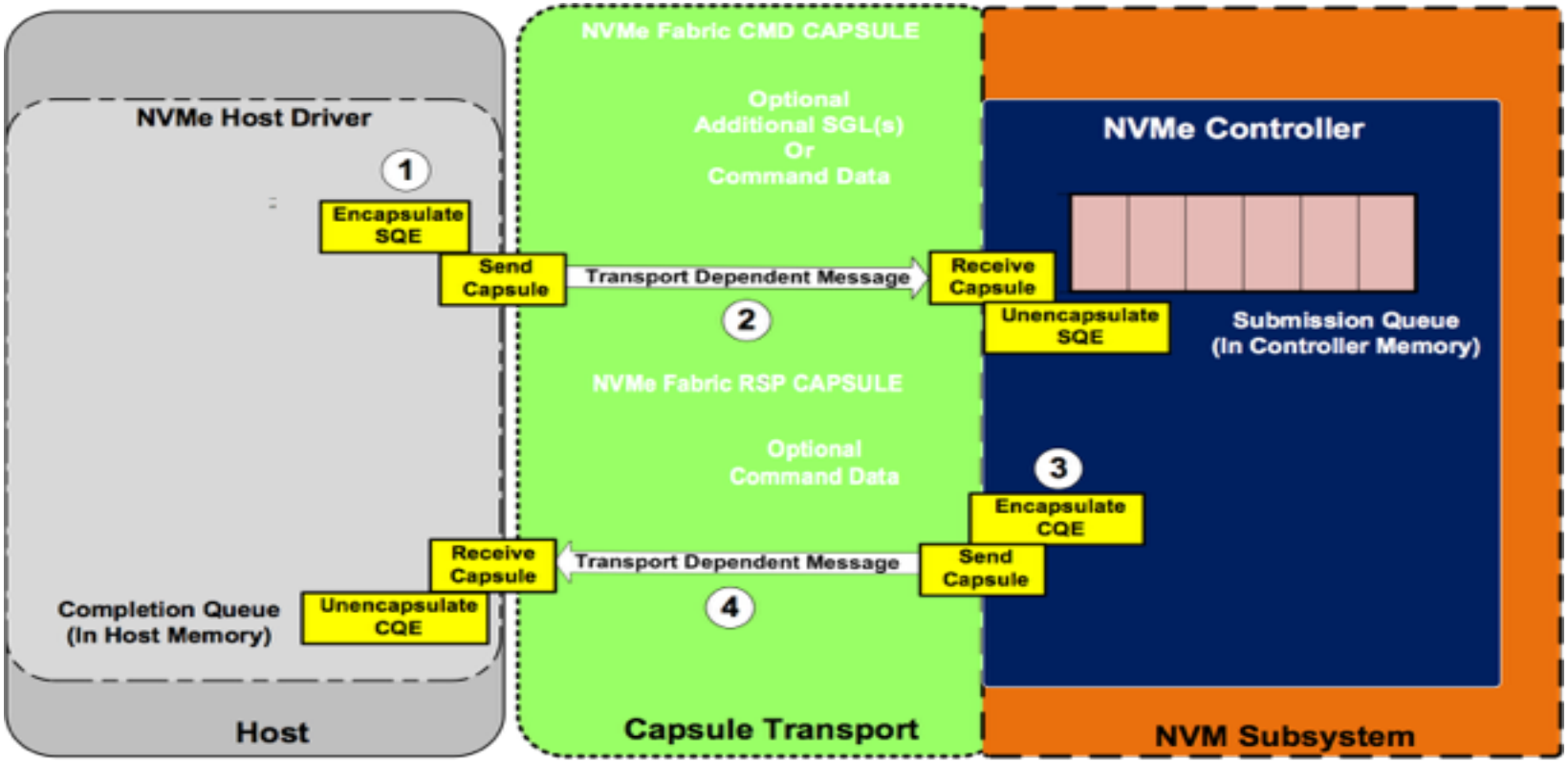
PCIe



- 1. Host Driver enqueues the SQE in host-memory resident SQ
- 2. Host Driver notifies controller about new SQE by writing doorbell register
- 3. NVMe Controller dequeues SQE by reading it from the host memory SQ
- 4. NVMe Controller enqueues CQE by writing it to host-resident CQ
- 5. Host Driver dequeues CQE

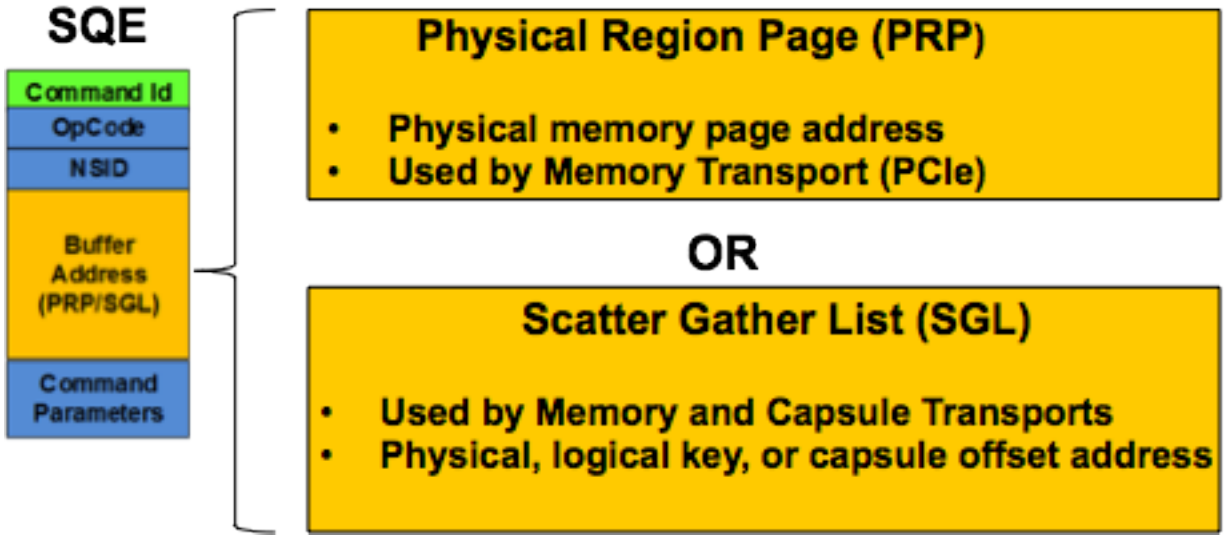
NVMe Queuing on Capsule Transports

NVMe over Fabrics



- 1. Host Driver encapsulates SQE into an NVMe Command Capsule
- 2. Fabric enqueues the SQE into the remote SQ by sending the Capsule
- 3. Controller encapsulates CQE into an NVMe Response Capsule
- 4. Fabric enqueues the CQE into the remote CQ by sending the Capsule

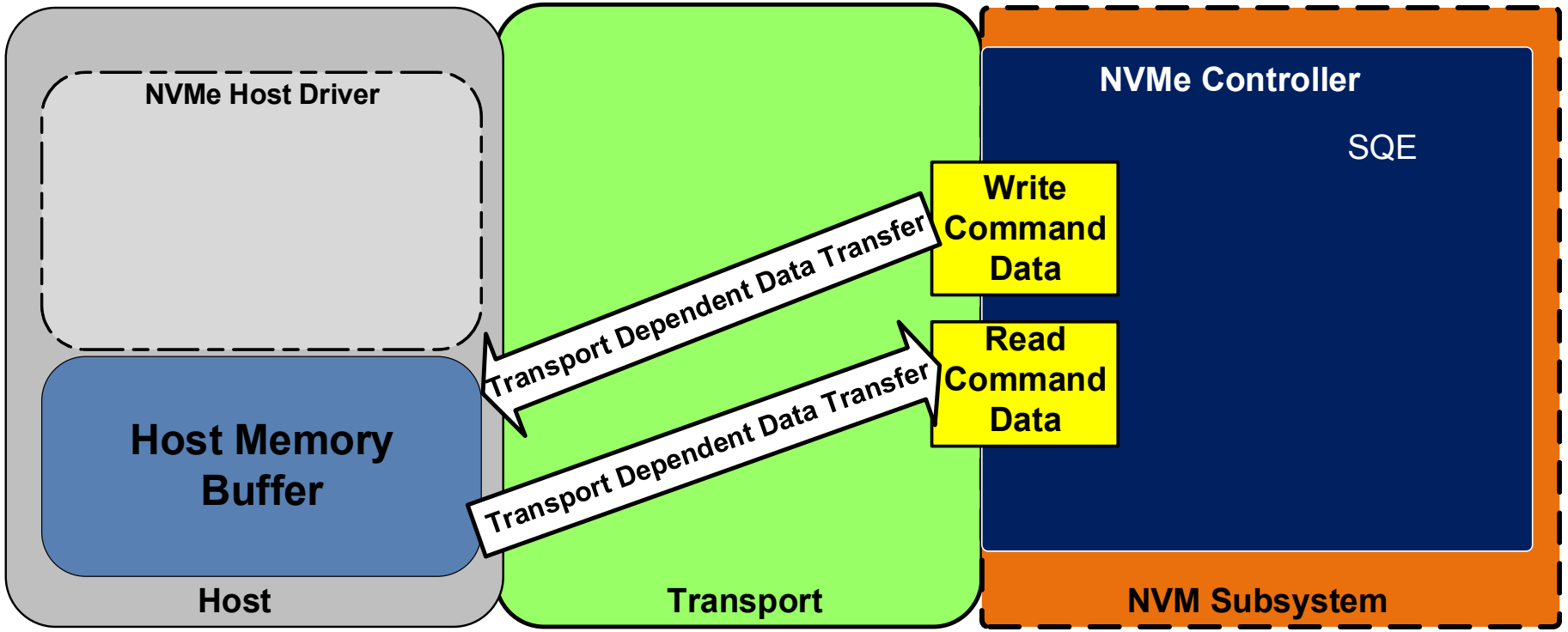
NVMe Command Data Transfers



- SQE contains the NVMe Command Data buffer address
  - Physical Region Page (PRP) used only for PCIe Transport
  - Scatter Gather List used by both PCIe and Capsule Transports
    - SGL = [Address, Length]
    - Address may be physical, logical with key, or capsule offset based
    - Supports SGL lists; { [Address,Length]...[Address,Length] }

NVMe Command Data Transfers

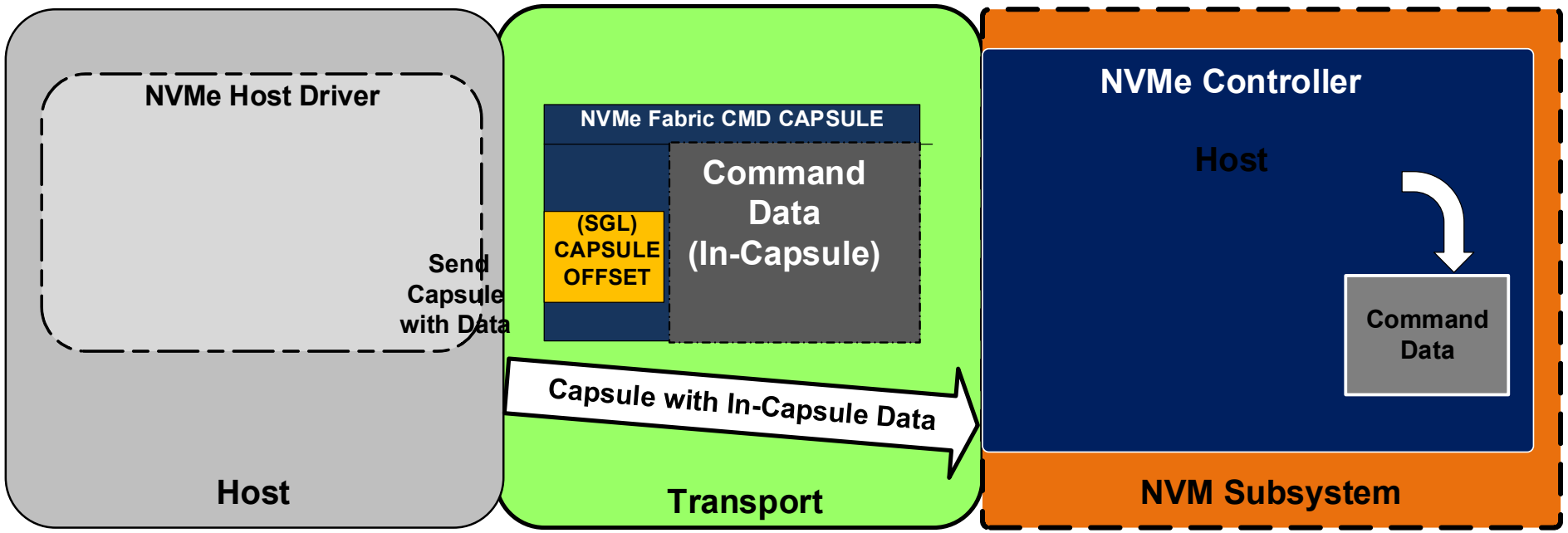
Controller Initiated



- Controller initiates the Read or Write of the NVMe Command Data to/from
- Host Memory Buffer
- Data transfer operations are transport specific; examples
  - PCIe Transport: PCIe Read/ PCIe Write Operations
  - RDMA Transport: RDMA_READ/RDMA_WRITE Operations

NVMe Command Data Transfers

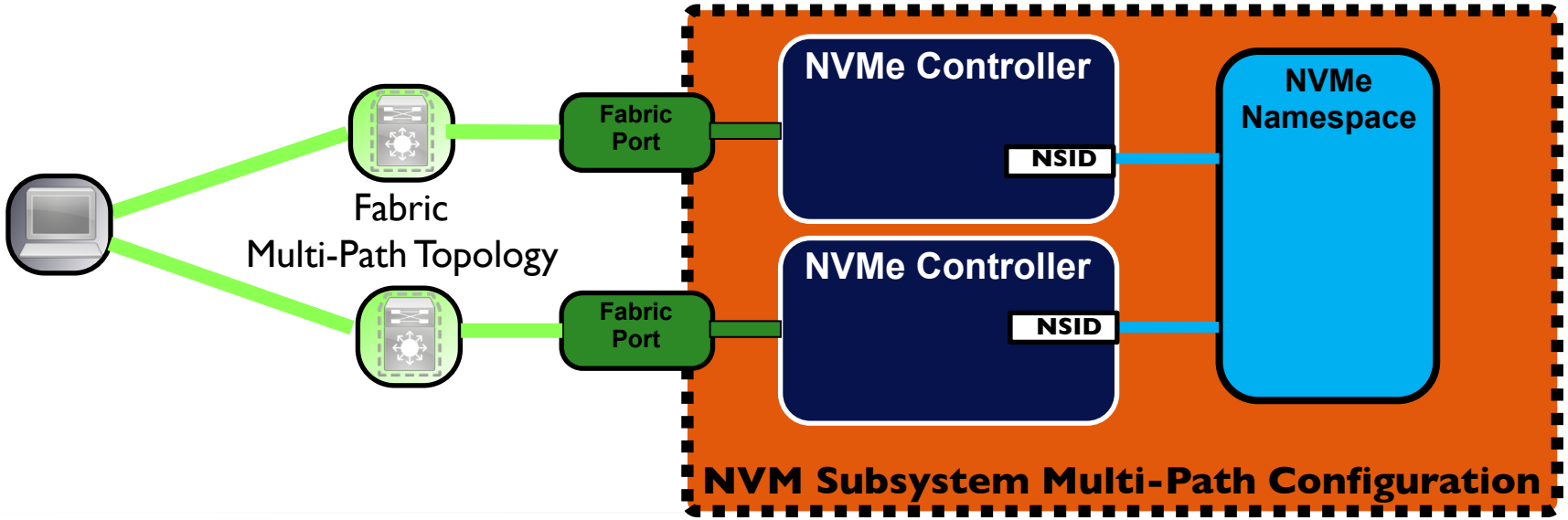
In-Capsule Data



- NVMe Command and Command Data sent together in Command Capsule
- Reduces latency by avoiding the Controller having to fetch the data from Host
- SQE SGL Entry will indicate Capsule Offset type address

Subsystem Multi-Path Configuration

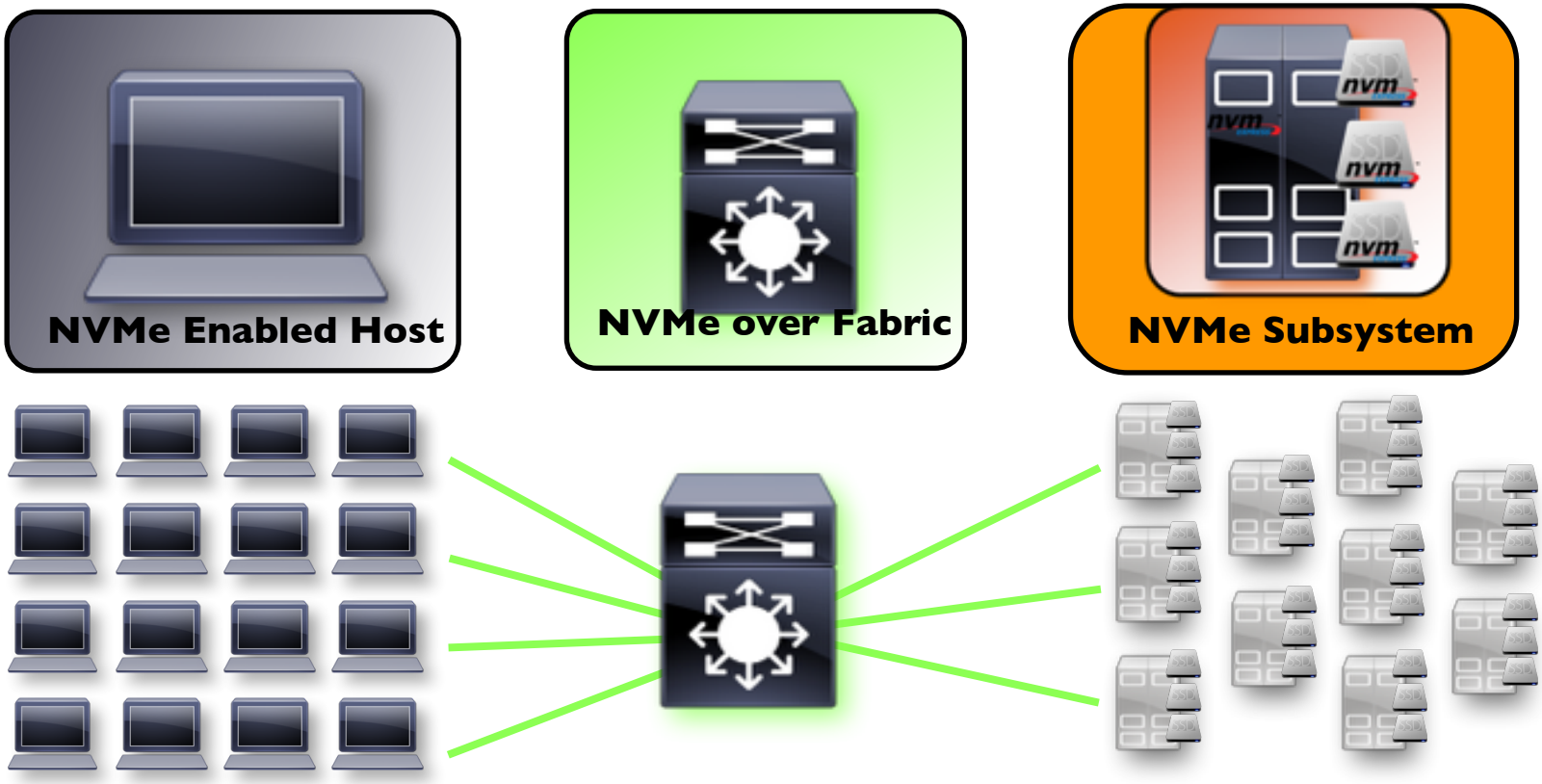
- Multiple fabric Ports attach to independent Fabric Paths between the Host and Subsystem
- One or more Controllers per Fabric Port
- Controllers share common Namespaces
- Host Multi-Path coordinates access to shared namespaces



NVMe over Fabrics Deployment

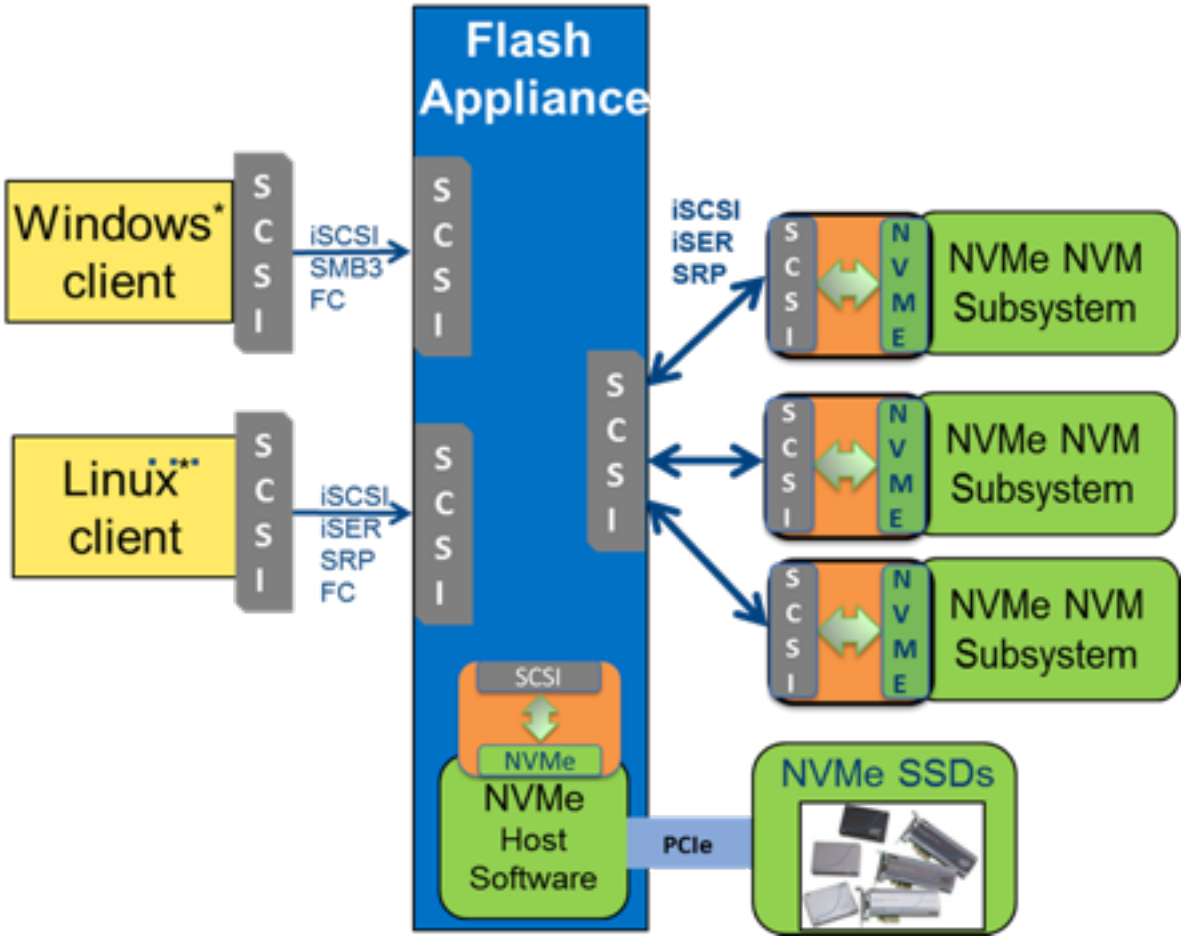
In This Section…

- NVMe over Fabrics Deployments



NVMe in all Flash Storage Appliances

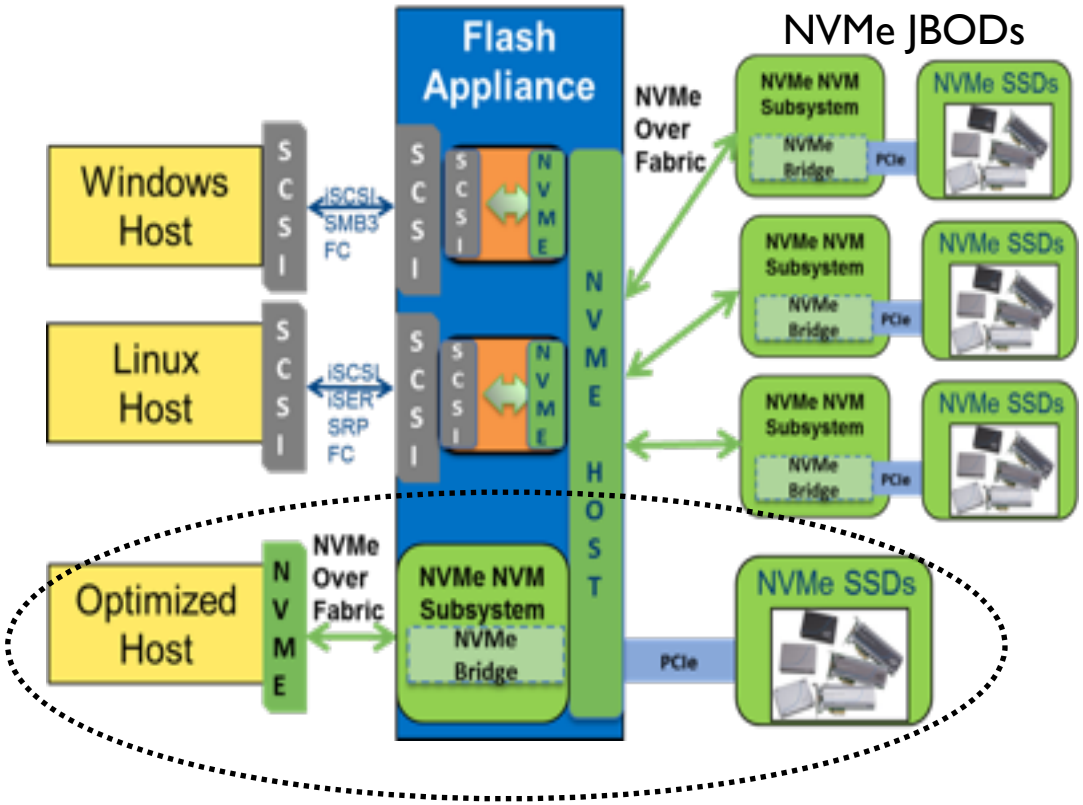
- A primary use case for NVMe PCIe SSDs is in an all Flash appliance
- Hundreds or more SSDs may be attached – too many for PCIe based attach scale-out
- Concern: Today back-end SSD scale-out over a fabric attach uses SCSI or proprietary block protocols



Requires protocol translation

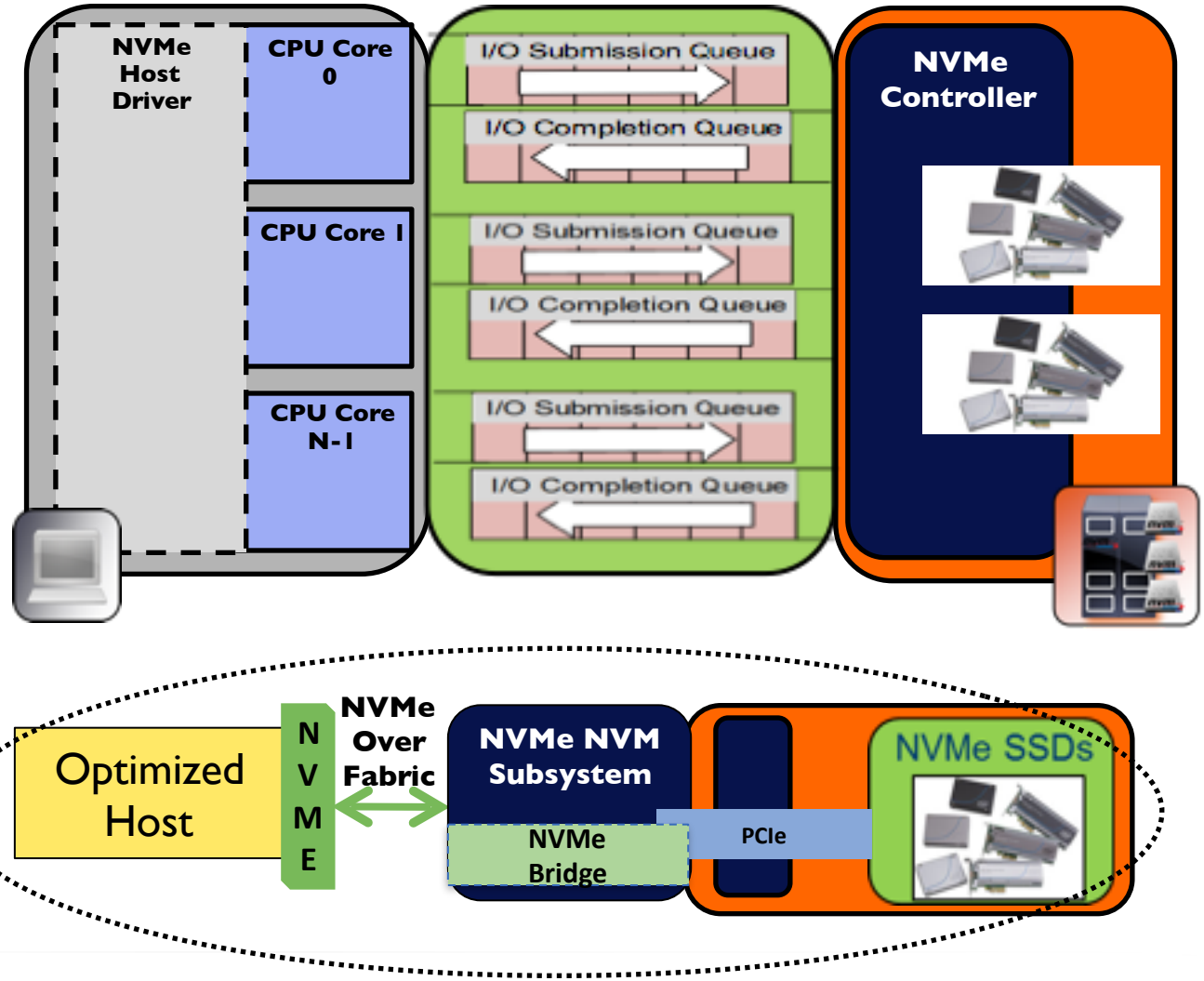
End-to-End NVMe over Fabrics

- High Performance All Flash Storage Systems With:
  - Scaled out Fabric-Attached NVMe JBODs with NVMe PCIe SSDs
  - NVMe hosts interfaces on low-latency, high bandwidth fabrics
  - Optimized NVMe Fabric host driver stacks



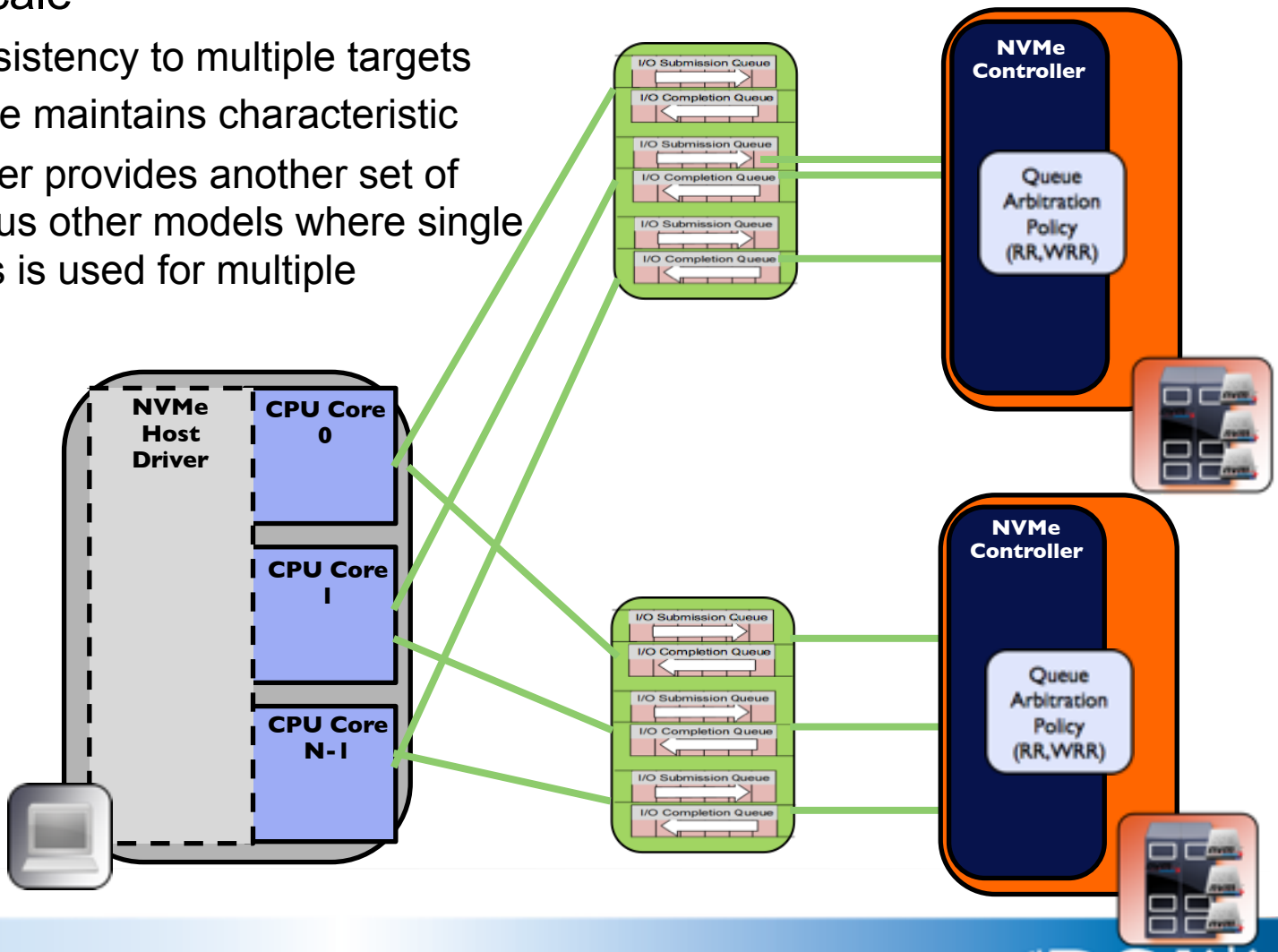
Maintaining Consistency

- Recall:
  - Multi-queue model
  - Multipathing capabilities built-in
- Optimized NVMe System
  - Architecture is the same, regardless of transport
  - Extends efficiencies across fabric



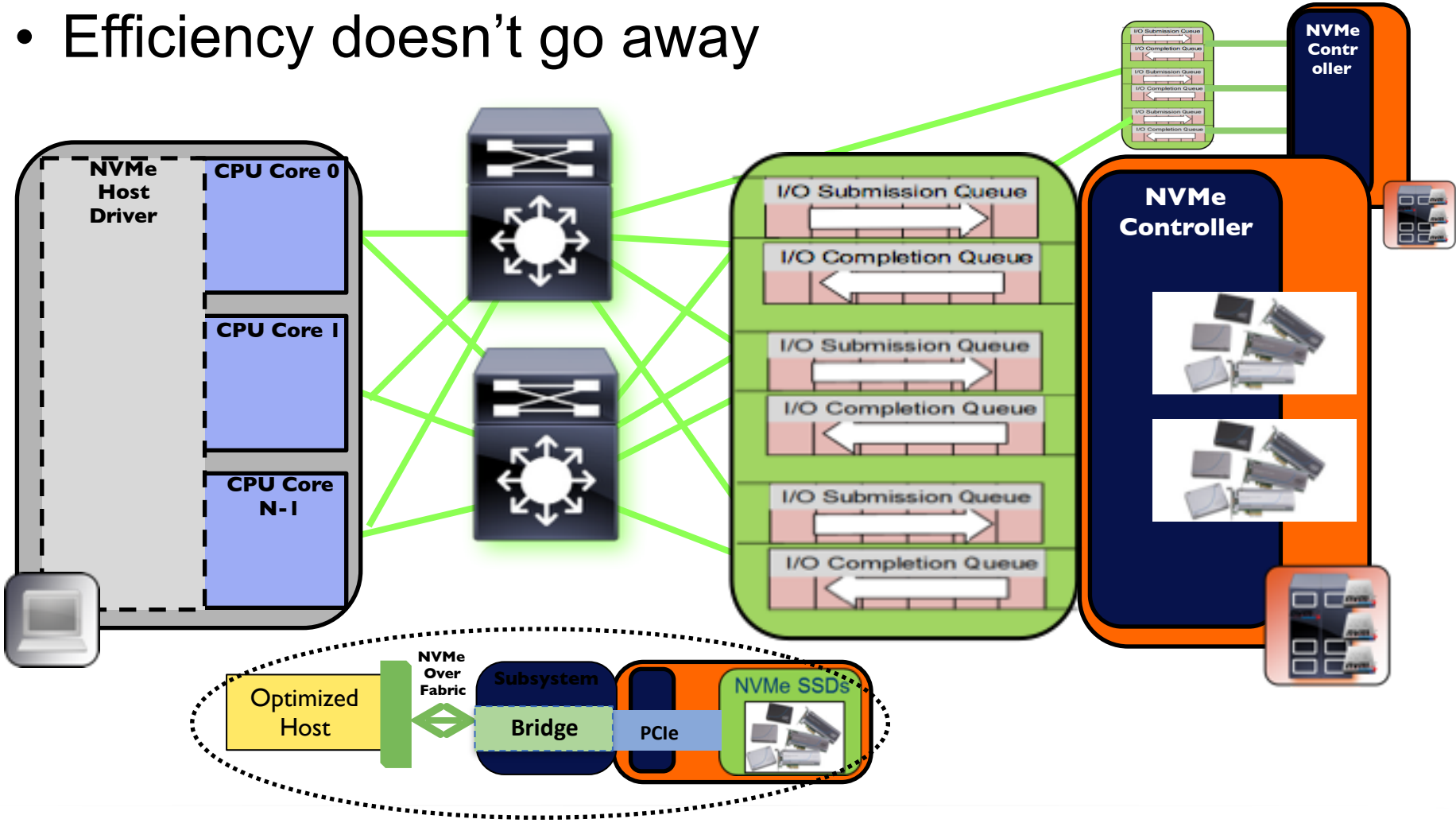
NVMe Multi-Queue Scaling

- Queue pairs scale
  - Maintain consistency to multiple targets
  - Each interface maintains characteristic
  - Each controller provides another set of queues, versus other models where single set of queues is used for multiple controllers



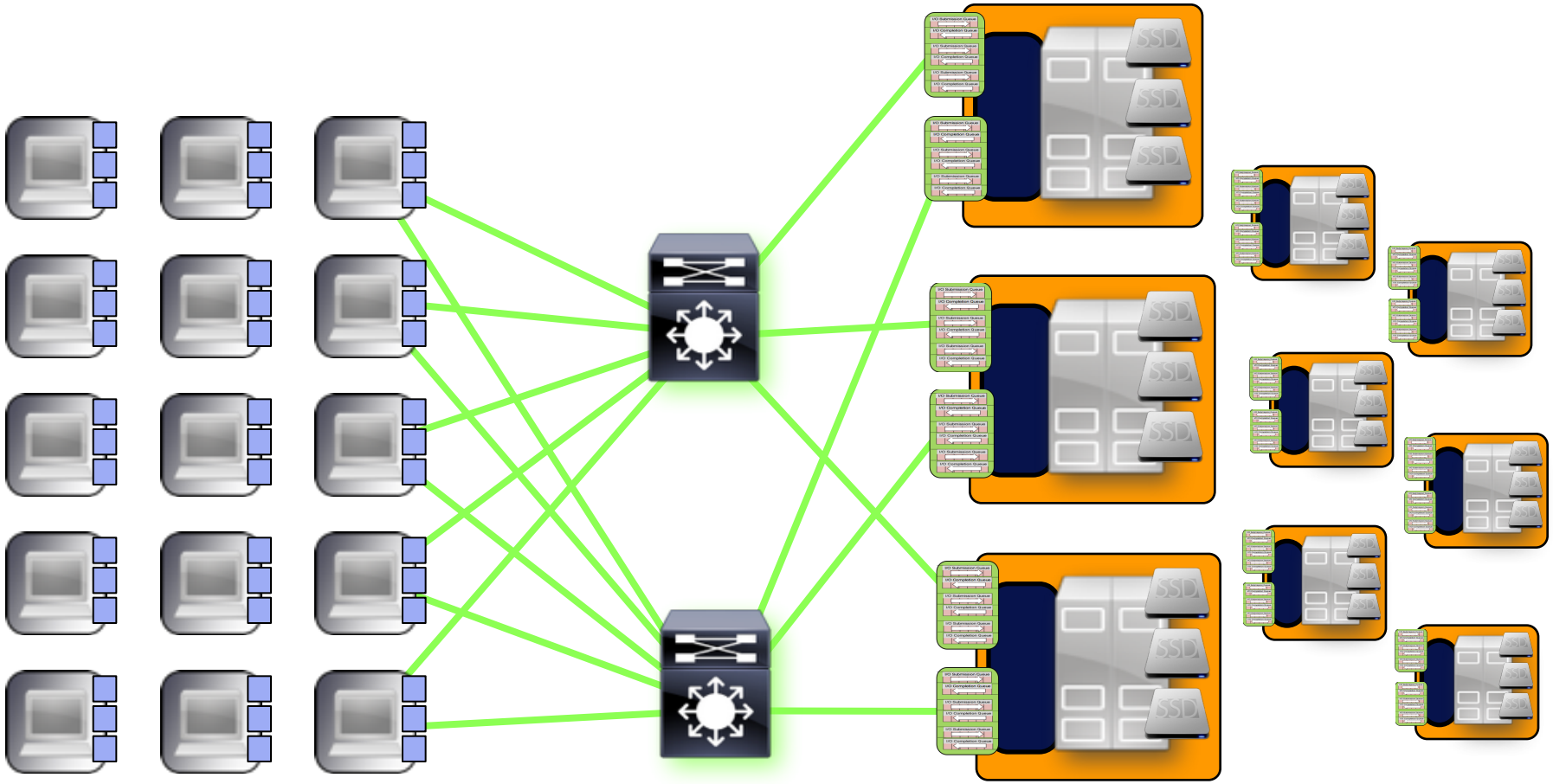
Connect Across Agnostic Transports

- Efficiency doesn't go away



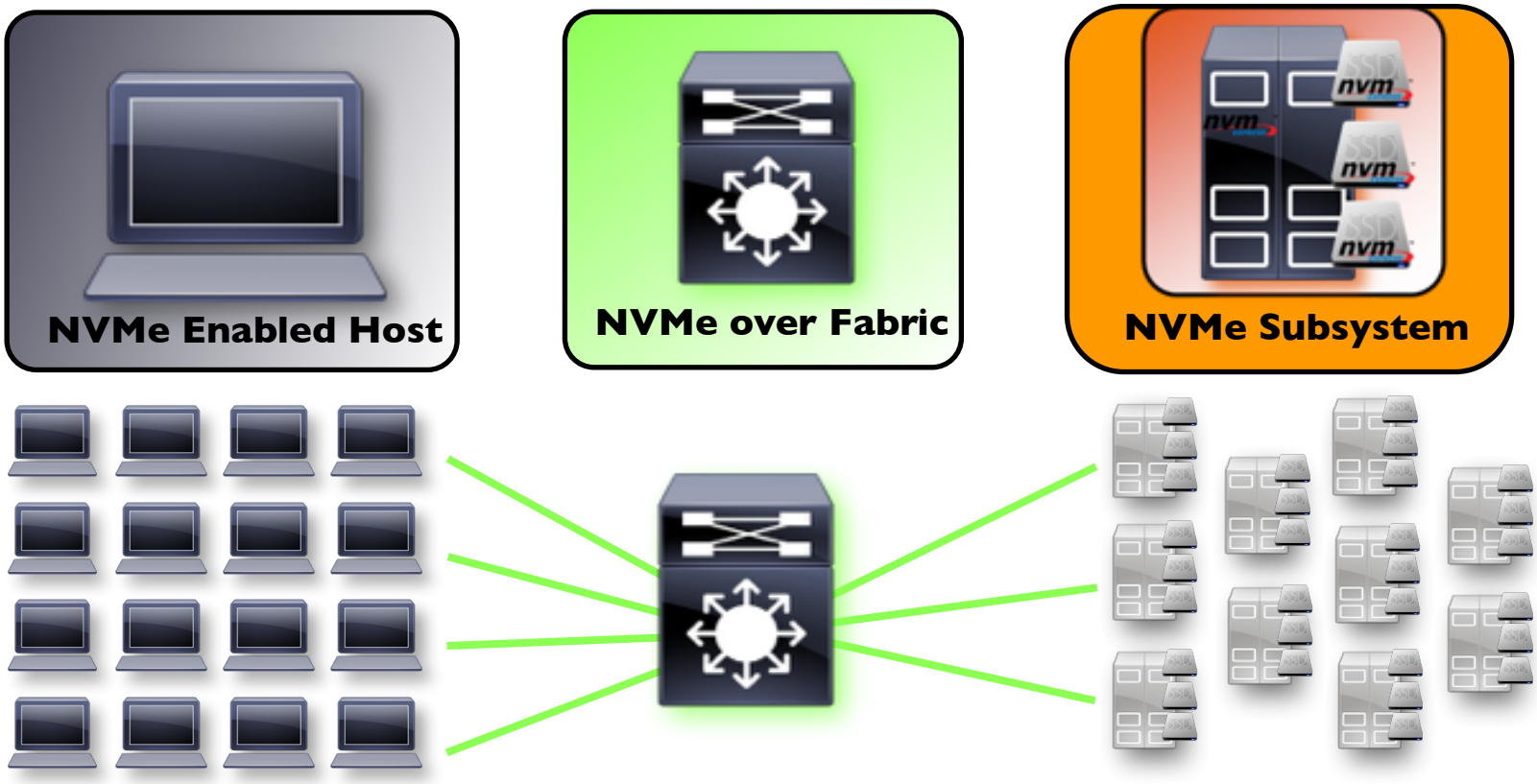
End-to-End NVMe Model

- NVMe efficiencies scaled across entire fabric



Full Circle

- NVMe over Fabrics Deployments



Summary

It's Worth The "Trouble"

- Why do we need another standard block storage networking protocol?
  - Low SSD latency puts pressure on delivering an efficient low-latency networking architecture
    - Emerging NextGen NVM based SSD are likely going to have 10x+ improvement in latency compared to NAND based SSDs
  - For cluster-based compute and storage, this brings new requirements on network latency and efficiency
    - Network becomes the bottleneck
- NVMe for PCIe proved the value of standardization
  - Enabled a storage eco-system with interoperability between vendors
  - NVMe over Fabrics standard is extending this model to fabrics
    - Defines a common abstraction and encapsulation
    - Maintains NVMe architecture and software consistency between fabric types
- We Move to Accept!

Summary

- NVMe was built from the ground up to support a consistent model for NVM interfaces, even across network fabrics
- Simplicity of protocol enables hardware automated I/O Queues – NVMe transport bridge
- No translation to or from another protocol like SCSI (in firmware/software)
- Inherent parallelism of NVMe multiple I/O Queues is exposed to the host
- NVMe commands and structures are transferred end-to-end
- Maintains the NVMe architecture across a range of fabric types

Thank you!

- Special thanks to Dave Minturn from Intel, co-author of this presentation