Flash Memory Summit

Architected for Performance

NVMe™ over Fabrics: What's new in 1.1

Sponsored by NVM Express™ organization, the owner of NVMe™, NVMe-oF™ and NVMe-MI™ standards

Speakers

TCP Transport	Multi Pathing	Discovery, Flow Control
Sagi Grimberg	Fred Knight	David Black
lightbits	NetApp®	DELL EMC

NVMe-oF™ 1.1: TCP Transport for NVMe-oF

Sagi Grimberg

Chief Software Architect, Lightbits

Why Do We Need Another NVMe™ Transport?

- PCIe®
  - Great for direct attached NVMe™ SSDs
  - Does not scale well to large topologies
- FC and RDMA (Infiniband, RoCE, iWARP)
  - Provides a high degree of scalability, but requires special networks and hardware
- TCP
  - Ubiquitous (does not require special networks or hardware)
  - Scalable allowing large scale deployments and operation over long distances
  - Can provide performance (throughput and latency) that is comparable to direct attached NVMe SSDs

NVMe-oF™ Transport Taxonomy

NVMe Transports

Memory
Commands/Responses & Data use Shared Memory

Example
PCI Express

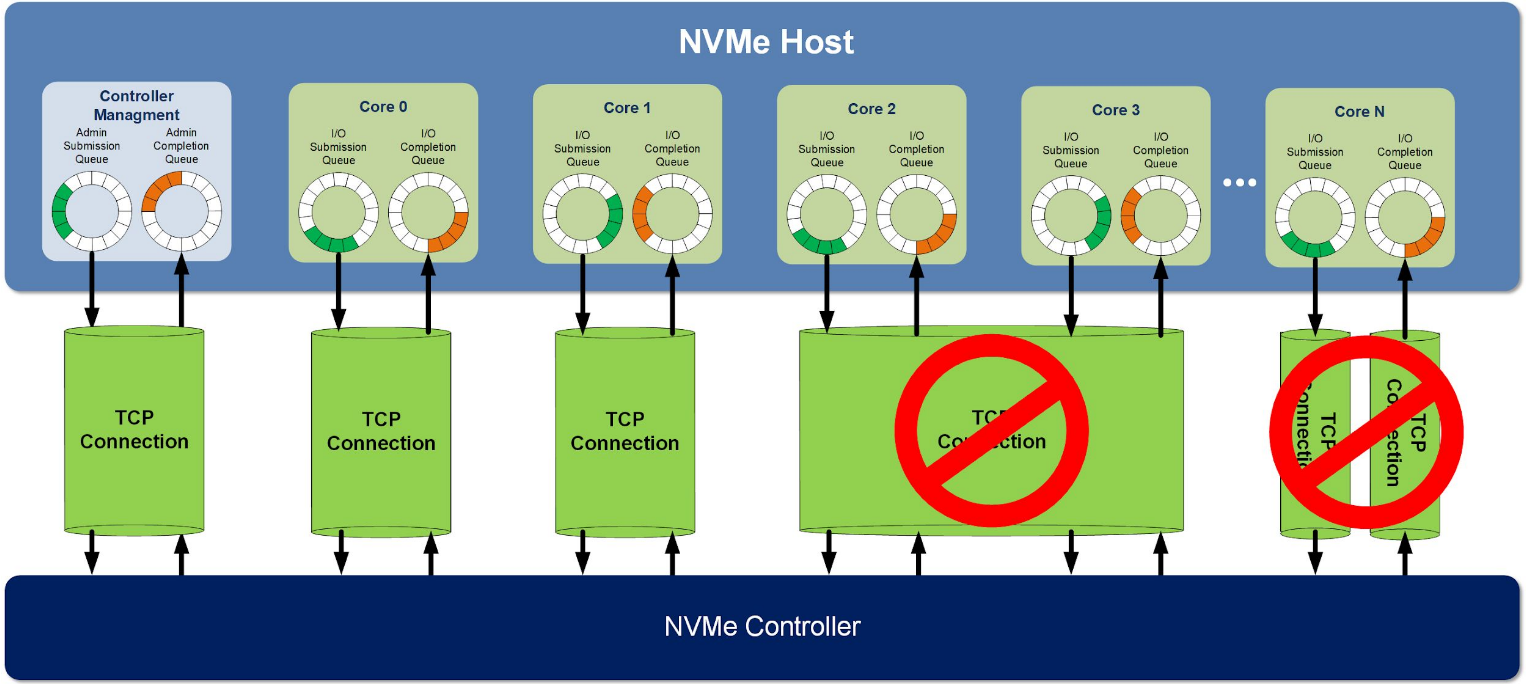
Message
Commands/Responses use Capsules
Data may use Capsules or Messages

Examples
Fibre Channel
TCP

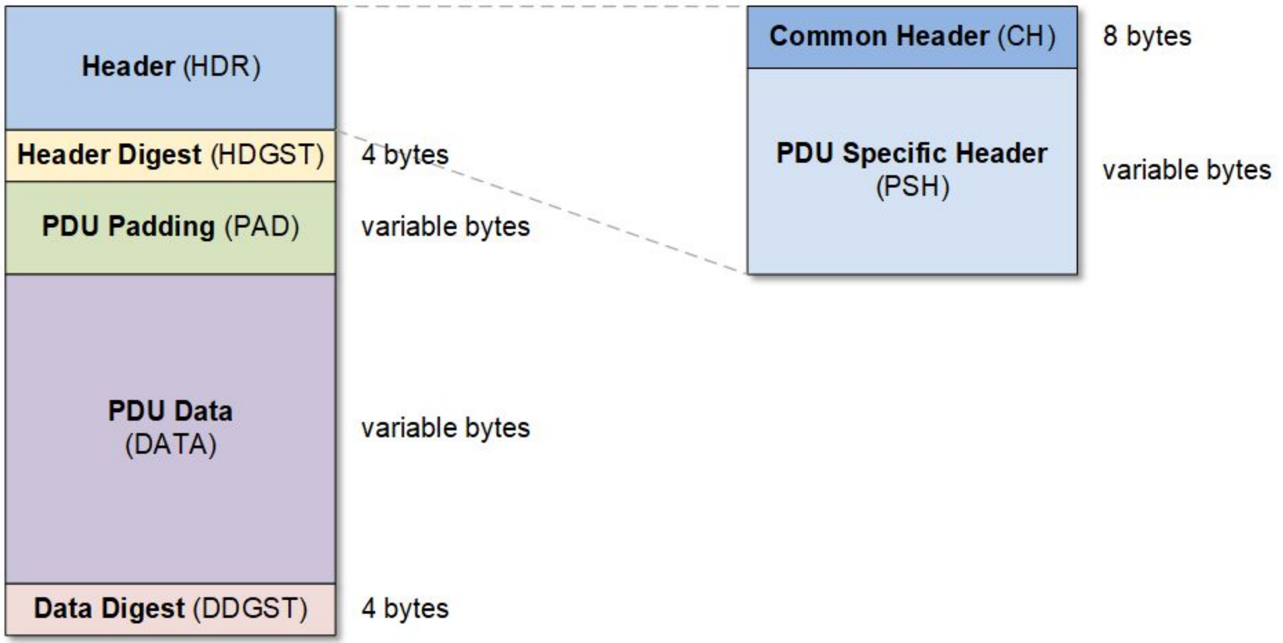
Message / Memory
Commands/Responses use Capsules
Data may use
Capsules or Shared Memory

Examples
RDMA
(InfiniBand, RoCE, iWARP)

NVMe^{TM}/TCP Queue Mapping



NVMe™/TCP PDU Structure

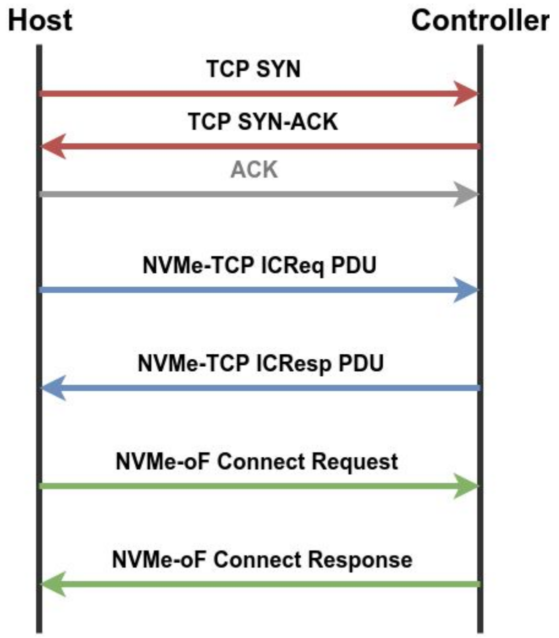


NVMe™/TCP Protocol Data Units (PDUs)

PDU Group	PDU Name	PDU Direction	Description
Initialize Connection	**ICReq**	Host to Controller	**Initialize Connection Request:** A PDU sent from a host to a controller to communicate NVMe™/TCP connection parameters and establish an NVMe/TCP connection
	ICResp	Controller to Host	**Initialize Connection Response:** A PDU sent from a controller to a host to accept a connection request and communicate NVMe/TCP connection parameters
Terminate Connection	**H2CTermReq**	Host to Controller	**Host to Controller Terminate Connection Request:** A PDU sent from a host to a controller in response to a fatal transport error
	C2HTermReq	Controller to Host	**Controller to Host Terminate Connection Request:** A PDU sent from a controller to a host in response to a fatal transport error
Capsule Transfer	**CapsuleCmd**	Host to Controller	**Command Capsule:** A PDU sent from a host to a controller to transfer an NVMe over fabrics command capsule
	CapsuleResp	Controller to Host	**Response Capsule:** A PDU sent from a controller to a host to transfer an NVMe over fabrics response capsule
Data Transfer	**H2CData**	Host to Controller	**Host to Controller Data:** A PDU sent from a host to a controller to transfer data to the controller
	C2HData	Controller to Host	**Controller to Host Data:** A PDU sent from a controller to a host to transfer data to the host
	R2T	Controller to Host	**Ready to Transfer:** A PDU sent from a controller to a host to indicate that it is ready to accept data

Connection Establishment

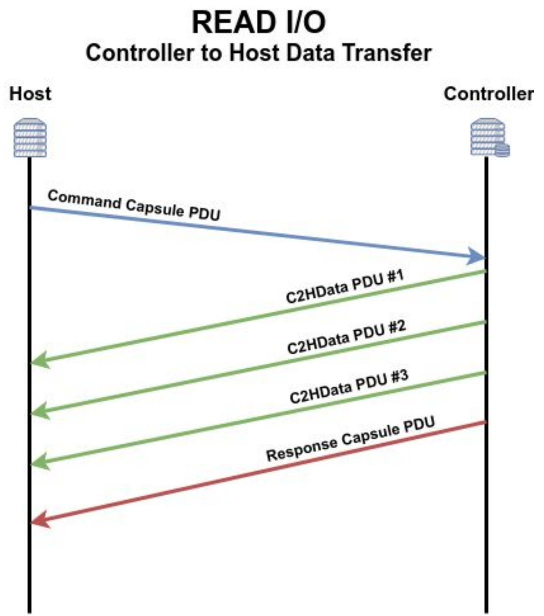
- Stage 1: TCP Connection Establishment
  - General TCP parameters
- Stage 2: NVMe™/TCP Connection Establishment
  - Parameter Negotiation
  - Features Support
- Stage 3: NVMe-oF™ Connection Establishment
  - Controller Binding
  - Queue Sizing



Flash Memory Summit

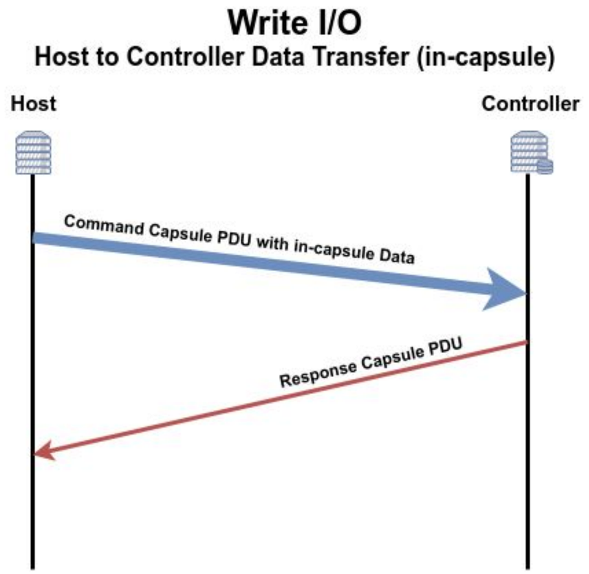
Data Transfer – Controller to Host

- Host issues a Command Capsule PDU
  - Contains the NVMe™ command
- Controller sends the Data payload to the host
  - Using one or more C2HData PDUs
- Controller sends a Response Casule PDU
  - Usually the NVMe completion entry



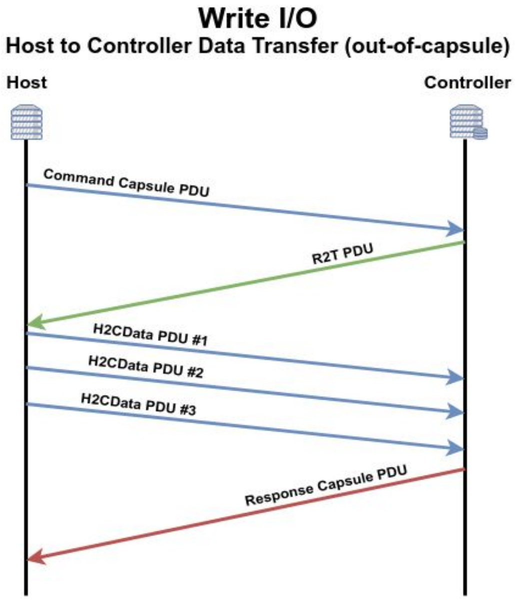
Data Transfer – Host to Controller (in-capsule)

- Host issues a Command Capsule PDU
  - Contains the NVMe™ command
  - Contains in-capsule Data
    - As supported by the Controller
- Controller sends a Response Casule PDU
  - Usually the NVMe completion entry



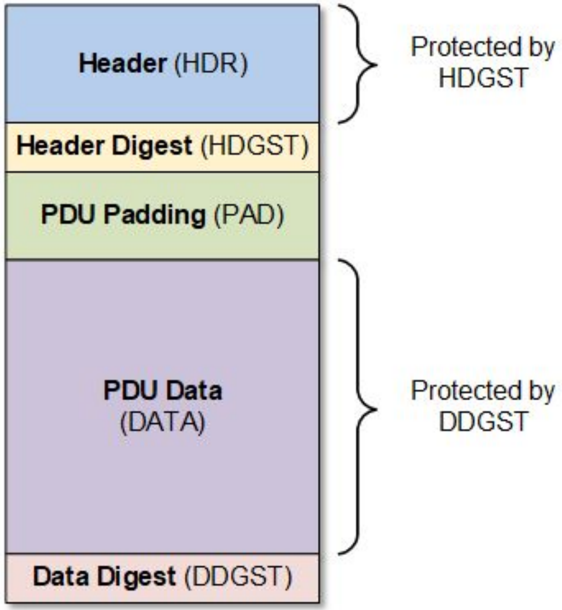
Data Transfer – Host to Controller (out-of-capsule)

- Host issues a Command Capsule PDU
  - Contains the NVMe™ command
- Controller sends a "Ready to Transfer" (R2T) solicitation
  - Host must support at least oneR2T per Command Capsule
- Host sends Data payload for that R2T using one or more H2CData PDUs
- Controller sends a Response Casule PDU
  - Usually the NVMe completion entry



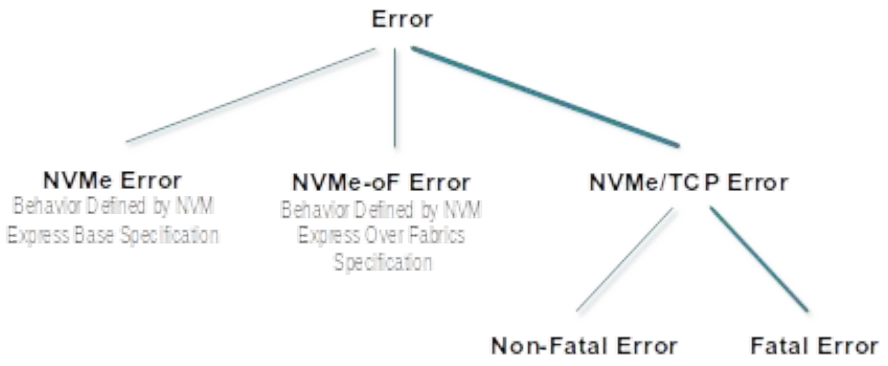
Header and Data Digest

- PDU Data integrity for both header and PDU Data
- Both Header and Data Digests are calculated using CRC32C (http://www.rfc-editor.org/rfc/rfc3385.txt)
- Generated by the sender and verified by the receiver
- Header Digest protects the PDU header it trails
  - Common Header (8 bytes)
  - Type-Specific Header (Variable Size)
- Data Digest protects the PDU Data payload it trails
  - Exists only for PDUs that contain Data payload



NVMe™/TCP Errors

- NVMe™/TCP Non-Fatal Error
  - An error that may affect one or more commands, but from which the transport is able to recover and continue normal operation
  - Commands affected by a non-fatal error are completed with a "Transient Transport Error" status code



- NVMe/TCP Fatal Error
  - An error from which the transport is not able to recover and continue normal operation
  - Fatal errors are handled by terminating the NVMe/TCP connection

NVMe-oF™ 1.1: Multi Pathing Improvements

Frederick Knight

Principal Engineer, NetApp

Multi-Pathing Improvements

NVMe™ Multi-Pathing improvements (2 TPs)

- Primary use case is in NVMe-oF™ Fabrics
- Basic commands in the NVMe Base spec (not fabric only commands)
- Result: Enable additional NVMe-oF implementations

2. Asymmetric Namespace Access (TP 4004)
    - Inform hosts about access characteristics of namespaces
    - Already included in Rev 1.4
3. Domains and Divisions (TP 4009)
    - Large NVM Subsystems
    - In 30-day member review

Single-Pathing

Original design of NVMe™ 1.0 had only single pathing

- Everything worked, or nothing worked

Namespaces are accessed through one controller

- Multiple controllers cannot be used to access the namespace

Single-Pathing

Original design of NVMe™ 1.0 had only single pathing

- Everything worked, or nothing worked

Namespaces are accessed through one controller

- Multiple controllers cannot be used to access the namespace
- Any problem on the path stops access

Examples

Most SSDs are single port, and therefore single path; some recent devices have added multiple ports

Multi-Pathing: Symmetric Access

Revision 1.1 added multiple access capability

- Multi-pathing
  - Single host with multiple access paths
  - Requires multiple controllers
- Shared namespace
  - Multiple hosts with one or more access paths each
  - Requires multiple controllers

First controller/path redundancy; allows partial failures

Assumption that access characteristics through each controller to the NVM are the same.

- It doesn't matter which controller is used by the host
- Discovered via CMIC + NMIC fields

Multi-Pathing: Redundancy and Performance

Multi-Pathing: Asymmetric Access

TP4004 added Asymmetric Namespace Access (in Rev 1.4)

Removes assumption that access characteristics through controllers to the NVM are the same; controllers provide:

- Optimized access
- Non-Optimized access
- Inaccessible access

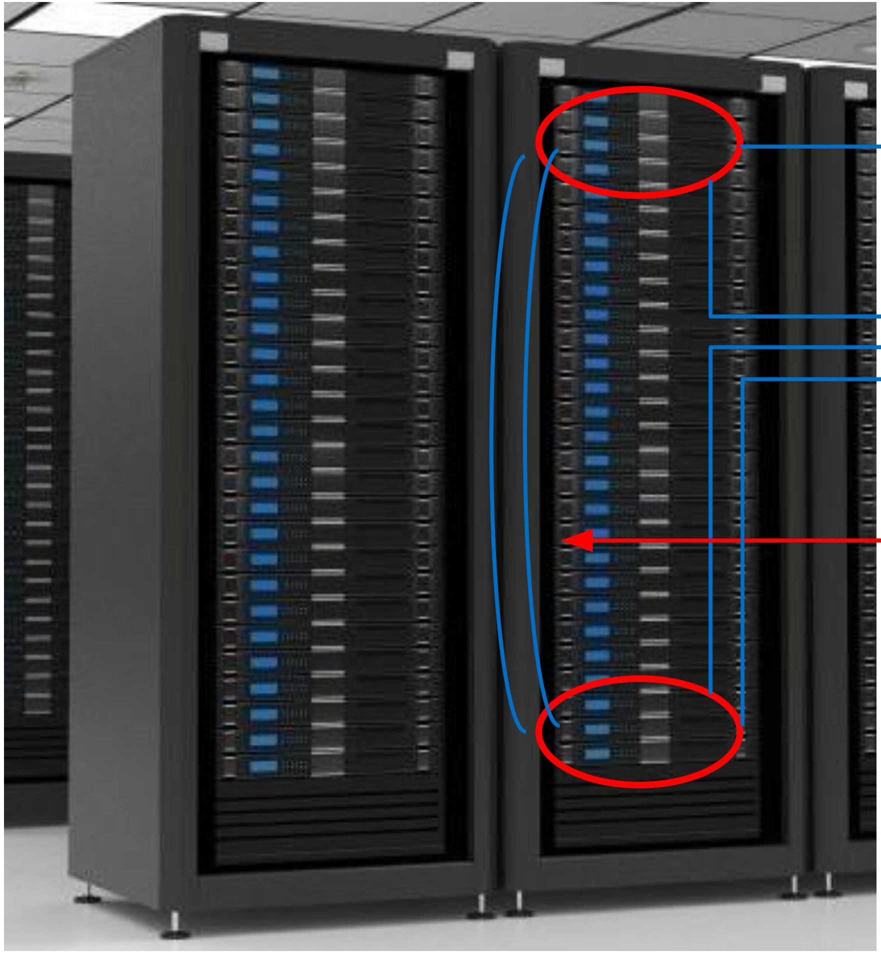
Hosts use these characteristics to provide redundancy and best access. Now, the host cares which controller is used to access a namespace.

Also allows partial failures

Discovered via CMIC field

Multi-Pathing: Asymmetric Access

Ex
a
m
pl
e



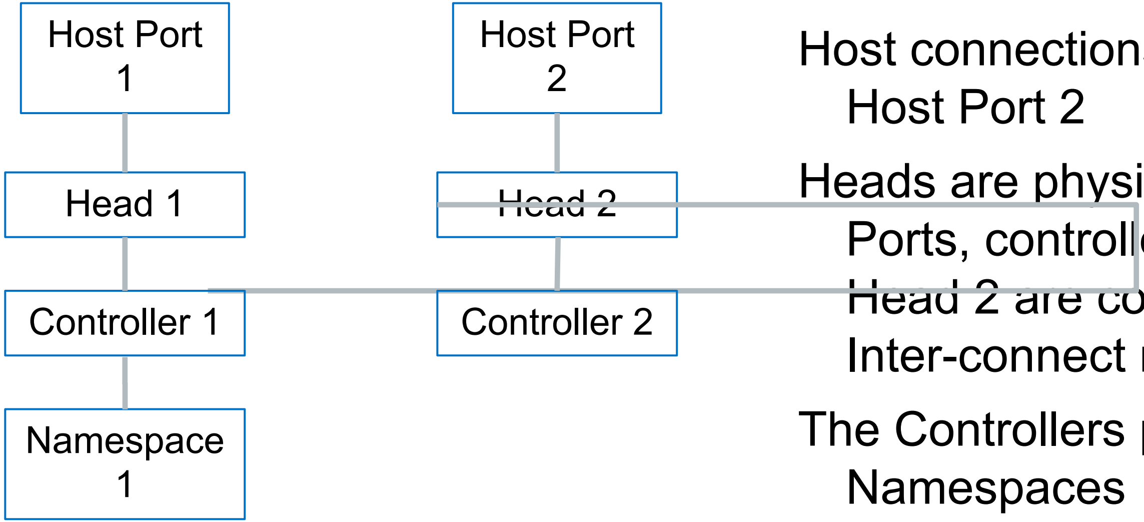
Head 1 contains:
Controller 1
NVM for Namespace 1

Fabric connections from Host

Inter-connect mechanism

Head 2 contains:
Controller 2

Asymmetric Storage System - Logical View



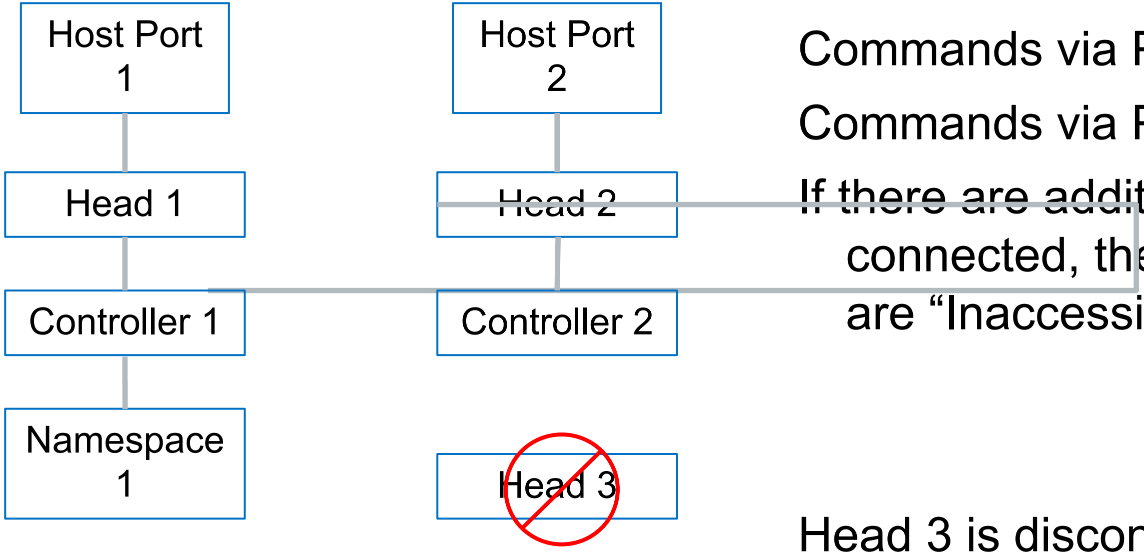
Host connections are via Host Port 1 and Host Port 2

Heads are physical boxes that contain Host Ports, controllers, and NSs – Head 1 and Head 2 are connected using an Inter-connect mechanism

The Controllers perform operations on the Namespaces

Operations from Host Port 2 must pass over the interconnect – acting like a toll booth

Asymmetric Storage System - Logical View



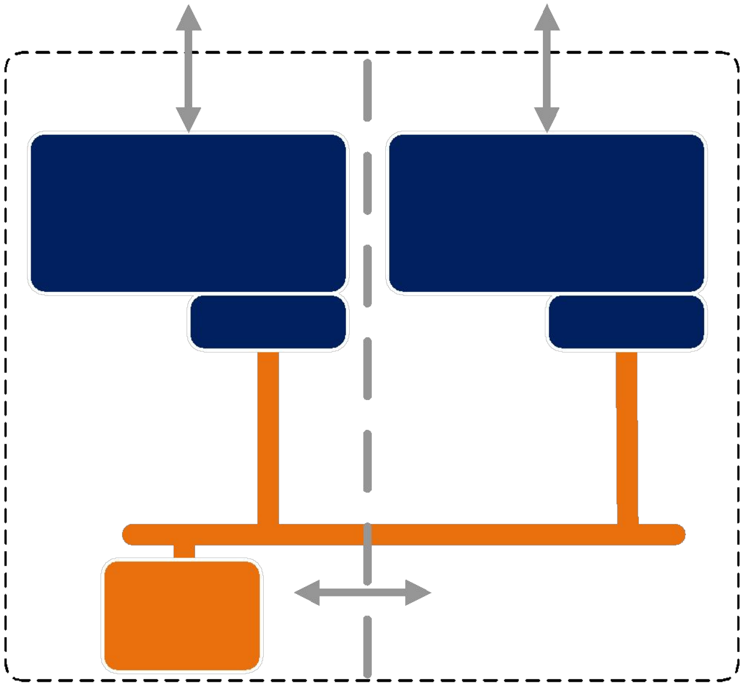
Commands via Port 1 are “Optimized”

Commands via Port 2 are “Non-Optimized”

If there are additional Heads that are not connected, the controllers in those Heads are “Inaccessible”

Head 3 is disconnected for maintenance

Asymmetric in the NVMe™ Specification



Multi-Pathing Improvements

NVMe™ Multi-Pathing improvements (2 TPs)

- Primary use case is in NVMe-oF™ Fabrics
- Basic commands in the NVMe Base spec (not fabric only commands)
- Result: Enable additional NVMe-oF implementations

2. Asymmetric Namespace Access (TP 4004)
   - Inform hosts about access characteristics
   - Already in Rev 1.4
3. Domains and Divisions (TP 4009)
   - Large NVM Subsystems
   - In 30-day member review

Domains and Divisions

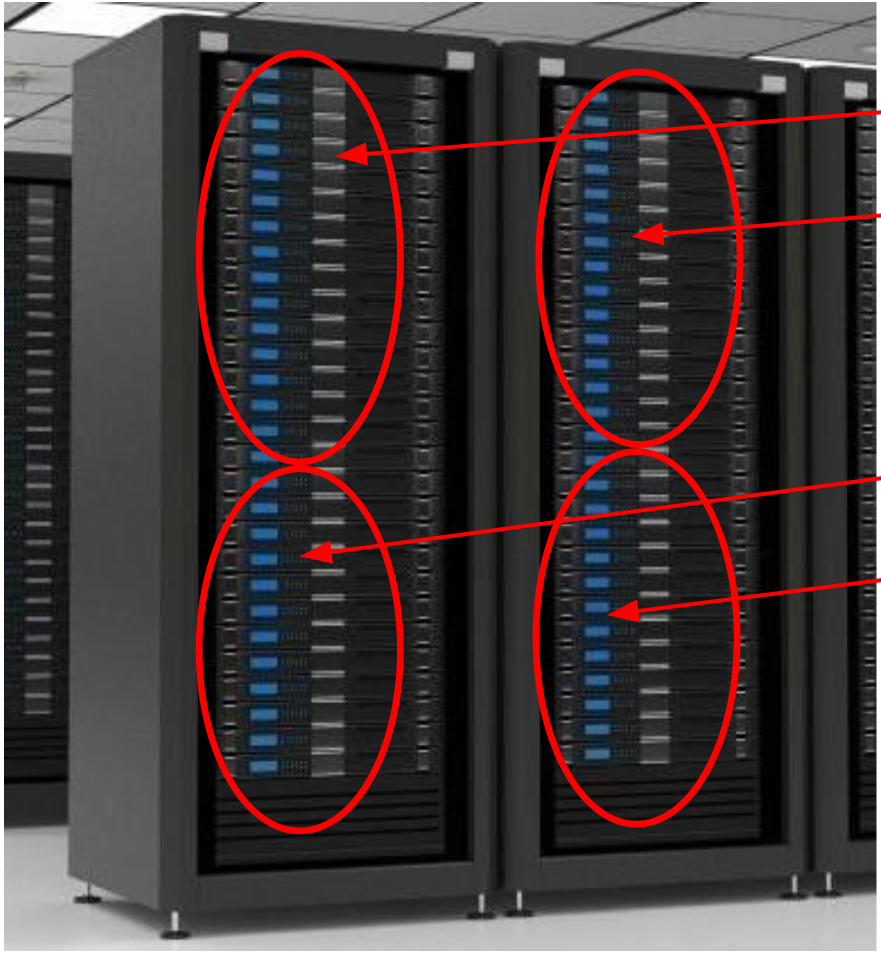
A Domain is the smallest indivisible unit within the NVM subsystem that shares state – For example:

- Power state – each domain may be powered independently
- Capacity information – each domain may have individual capacity
- Fault characteristics (there may be fault boundaries between domains)

Whole NVM subsystem state has been eliminated (scoped to domain)

Division is an event or action affecting communication between domains

- While present – global state may not be available
- Requires error codes from TP4004 for reporting



Domain 1

Domain 2

Domain 3

Domain 4

Domains and Divisions

Benefits for large NVM subsystems -

Scalability

Enables Non-Disruptive Rolling Maintenance

- Partial shutdown, perform maintenance, restore power

Enables Non-Disruptive Upgrades and Migration

- Add new domain (new hardware), shutdown and remove old domain (old hardware)

Enable Host detection of Domains

- Enhance host redundancy and performance

Summary

Asymmetric Namespace Access (TP 4004)

- Allow Host to determine access characteristics
- Notify host of access characteristic changes
- Already included in Rev 1.4

Domains and Divisions (TP 4009)

- Allow detection of Fault Boundaries
- Enhances Non-Disruptive operation
- In 30-day member review

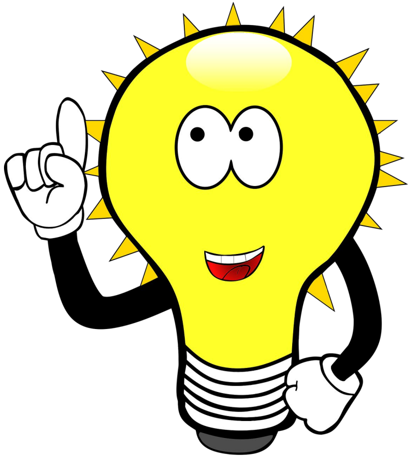


Image Credit: Conmongt
Needpix.com

NVMe-oF™ 1.1: Discovery & Transport Improvements

David L. Black, Ph.D.

Senior Distinguished Engineer, Dell EMC

Discovery & Transport Improvements

NVMe-oF^{TM} framework improvements (3 TPs)

- Result: Improved NVMe-oF implementations

2. Discovery: Persistent Controller (TP 8002)
   - Notify hosts when fabric configuration changes
3. Transport: Fabric I/O Queue Deletion (TP 8001)
   - Without terminating host-controller association
4. Transport: End-to-End Flow control (TP 8005)
   - Alternative to Submission Queue Flow Control

Image Credit: Nick Youngson, Alpha Stock Images


Image Credit: NOAA Wikimedia Commons

Discovery: Persistent Controller

Fabrics discovery: Discovery Controller

1) Host obtains NVM subsystem and fabric port info
2) Host contacts those NVM subsystems via those ports

Original design (NVMe-oF™) 1.0: One-shot

- Host disconnects after obtaining initial info

Design motivation: What if that info changes?

Solution: Persistent Discovery Controller

- Host retains connection to Discovery Controller
- Async event sent to host if discovery info changes

Host response to async event: Repeat discovery

- Important scenario: Fabric port added
- Host sees new Discovery Log Page entry for new port

Flash Memory Summit

nvm EXPRESS®

Transport: Fabric I/O Queue Deletion

NVMe™ Controller interface: Admin and I/O Queues

- I/O Controller: 1 Admin Queue & 1+ I/O Queues
- NVMe/PCIe: Create and Delete I/O Queue commands

NVMe over Fabrics: I/O Queue Management

- NVMe-oF™ 1.0: Create I/O Queues (Connect command)
  - Can't delete individual I/O Queues
- NVMe-oF 1.1: Delete I/O Queues (new Disconnect command)
  - Command sent on I/O Queue to be deleted (like Connect)
  - Disconnect: I/O Queue only, not Admin Queue
    - Only 1 Admin Queue, host loses contact with controller if deleted

Enables dynamic I/O Queue resource management

Image Credit: Nikin
Needpix.com

Transport: Fabric I/O Queue Termination - Bonus

Additional Functionality: Transport I/O Error Containment

- While we were in there …

Motivation: Unrecoverable I/O error, e.g., data corruption

- NVMe-oF™ 1.0: Terminate entire host-controller association
  - All fall down, start over with Connect command for Admin Queue
- NVMe-oF 1.1: Limit error impact to specific I/O Queue
  - Terminate that I/O Queue (involuntary disconnect)
  - Not always possible, simplifies host recovery when possible

Transport I/O Error Containment: Negotiated when Admin Queue created

- Usable only when both host and controller support

Image Credit: CERT
Wikimedia Commons

Discovery & Transport Improvements

NVMe-oF™ framework improvements (3 TPs)

- Result: Improved NVMe-oF implementations

2. Discovery: Persistent Controller (TP 8002)
   - Notify hosts when fabric configuration changes
3. Transport: Fabric I/O Queue Deletion (TP 8001)
   - Without terminating host-controller association
4. Transport: End-to-End Flow control (TP 8005)
   - Alternative to Submission Queue Flow Control

Image Credit: NOAA
Wikimedia Commons

Background: NVMe™ Command Queues

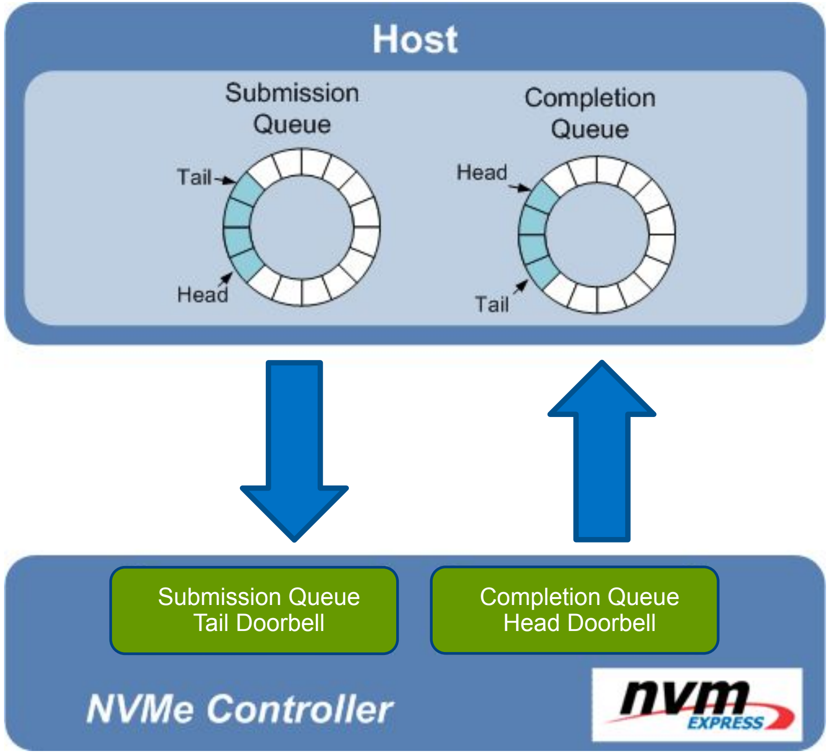


Image credit: Intel and NVM Express

Decouple host and controller

- Submission Queue (SQ): Submit commands to controller
- Completion Queue (CQ): Return completions to host

Queue occupancy:

- Enqueue at Tail
- Dequeue at Head
- Return entries for reuse

NVMe™ Flow Control: Overview

NVMe™/PCIe: Recipient manages queue occupancy

- Advance Queue Head to allow more commands to be submitted
- Queue Head not automatically advanced by command processing
- Queue Head advancement mechanism: Direction-specific
  - Submission Queue: Head pointer update in each command completion
  - Completion Queue: Host rings PCIe doorbell with updated Head pointer

NVMe-oF™ flow control: Submission only, no completion flow control

- Host has to be able to handle completions for all outstanding commands
- If host can't handle more completions, host pauses submitting commands

NVMe-oF Transports use lower level flow control mechanisms (e.g., TCP)

NVMe-oF™: End-to-End Flow control

Submission Queue: Head pointer update in each command completion

- Head pointer state maintained at host & controller
- Fabric required to deliver completions in order
- Prevents optimization for successful reads (common case)
  - From iSCSI: Set "It worked!" bit on last read data transfer

NVMe-oF™: End-to-End flow control: Omit submission flow control

- Size both SQ and CQ to maximum # of outstanding commands
  - Tradeoff: Static transport queue resources (not dynamic)
- Flow control mechanism negotiated by Connect command
  - Resource management may vary across implementations
- Enables transport and implementation optimizations

Summary

Persistent Discovery Controller (TP 8002)
- Notify host of fabric changes after initial discovery

Fabric I/O Queue Deletion (TP 8001)
- Dynamically manage I/O Queue resources
- Bonus: Transport I/O Error Containment

End-to-End Flow Control (TP 8005)
- Transport and implementation optimizations
- Applies to all three NVMe-oF™ Transports (RDMA, FC, TCP)

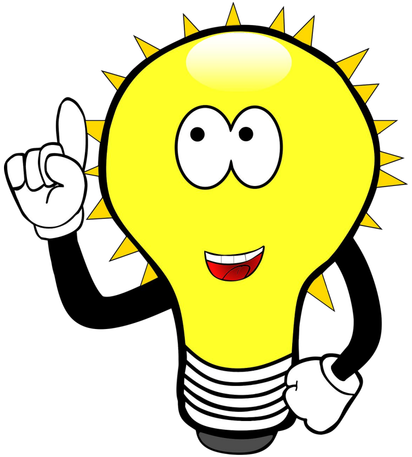


Image Credit: Conmongt
Needpix.com

Architected for Performance