SNIA NSF | NETWORKING STORAGE

What NVMe™/TCP Means for Networked Storage

Live Webcast
January 22, 2019

Today's Presenters

Sagi Grimberg
Lightbits

J Metz
Cisco

Tom Reu
Chelsio Communications

SNIA-At-A-Glance

170
industry leading organizations

2,500
active contributing members

50,000
IT end users & storage pros worldwide

Learn more: **snia.org/technical** **@SNIA**

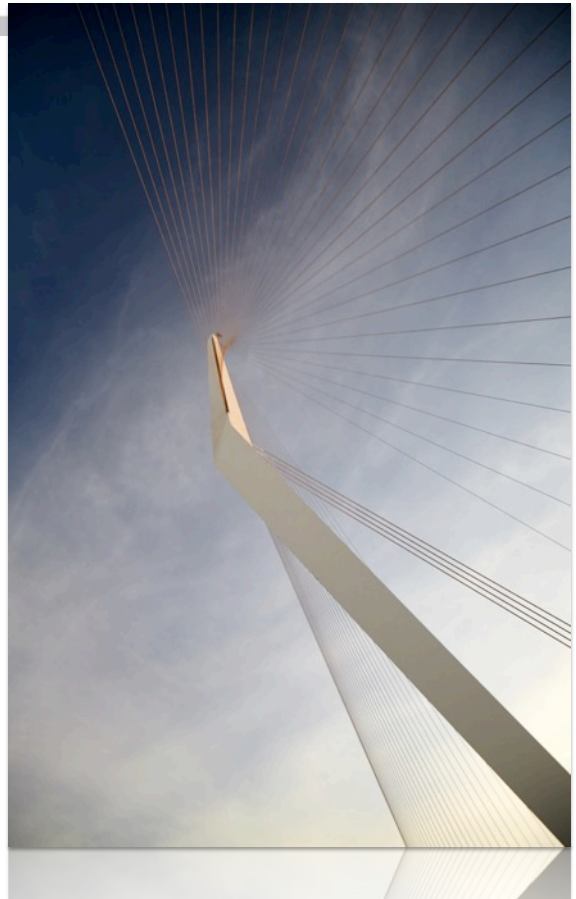
SNIA Legal Notice

- The material contained in this presentation is copyrighted by the SNIA unless otherwise noted.
- Member companies and individual members may use this material in presentations and literature under the following conditions:
  - Any slide or slides used must be reproduced in their entirety without modification
  - The SNIA must be acknowledged as the source of any material used in the body of any document containing material from these presentations.
- This presentation is a project of the SNIA.
- Neither the author nor the presenter is an attorney and nothing in this presentation is intended to be, or should be construed as legal advice or an opinion of counsel. If you need legal advice or a legal opinion please contact your attorney.
- The information presented herein represents the author's personal opinion and current understanding of the relevant issues involved. The author, the presenter, and the SNIA do not assume any responsibility or liability for damages arising out of any reliance on or use of this information.

NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.

Agenda

- NVMe over Fabrics (NVMe-oF™) Primer
- NVMe™/TCP
- Building a Better TCP
- Conclusions

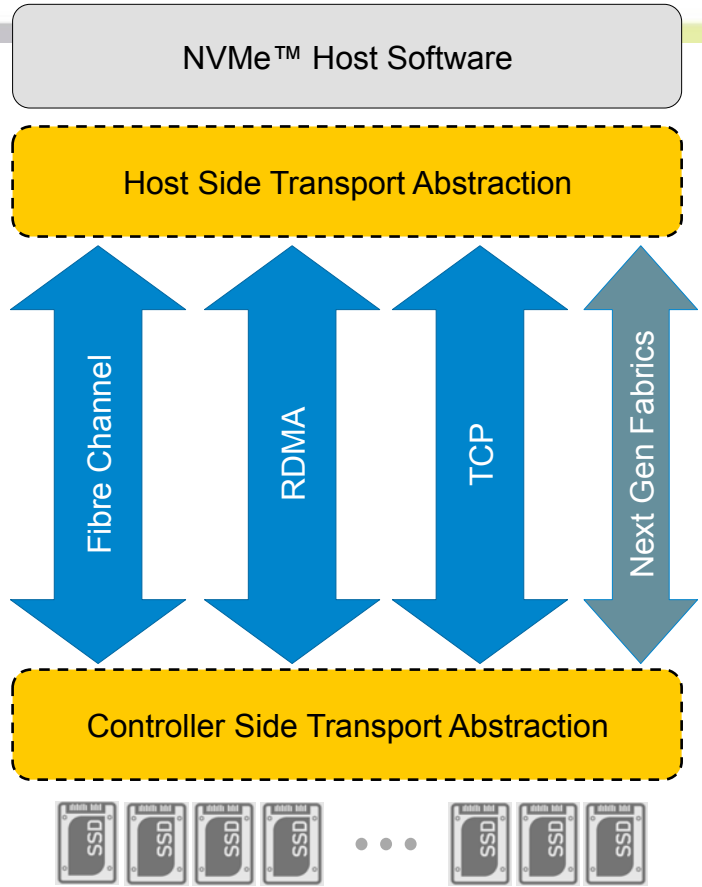


SNIA NSF | NETWORKING STORAGE

NVMe™ over Fabrics (NVMe-oF™) Primer

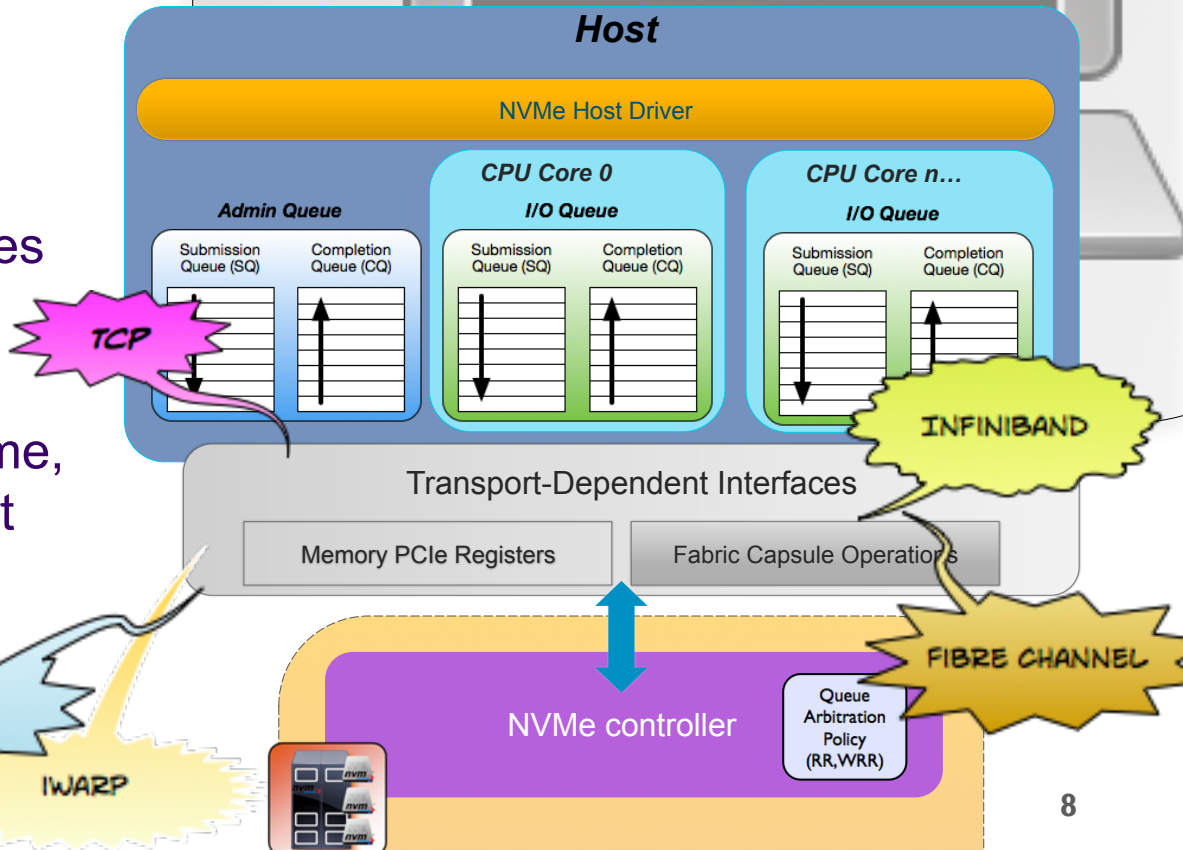
NVMe-oF Choices

- Enables disaggregation of NVMe™ SSDs without compromising latency and without requiring changes to networking infrastructure
- Independently scale storage & compute to maximize resource utilization and optimize for specific workload requirements
- Extends NVMe™ model: sub-systems, controllers namespaces, admin queues, data queues



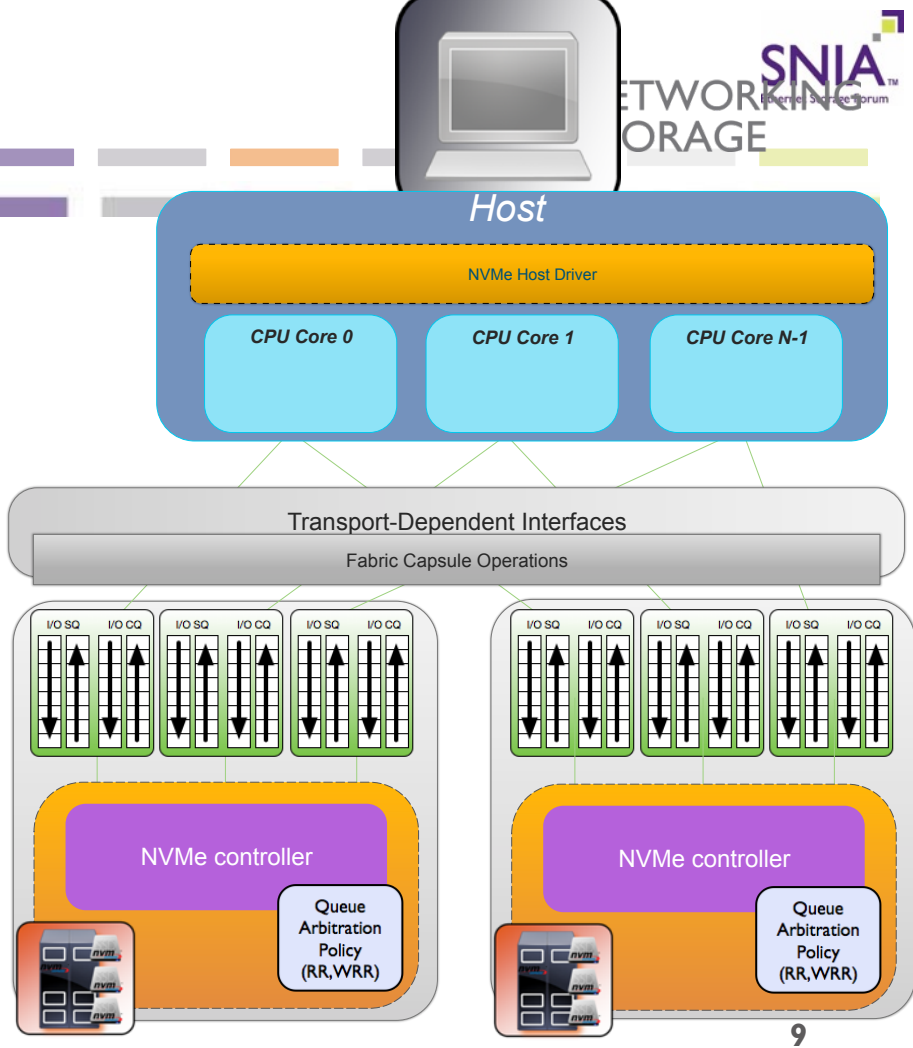
What's Special About NVMe over Fabrics?

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

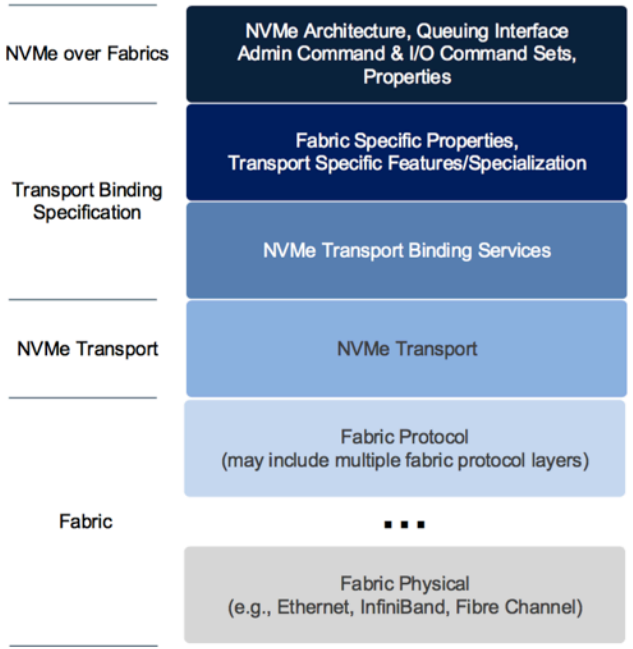


NVMe Multi-Queue Scaling

- Queue pairs scale
  - Maintain consistency to multiple Subsystems
  - Each controller provides a separate set of queues, versus other models where single set of queues is used for multiple controllers
- Efficiency retained



What's Special About NVMe-oF: Bindings

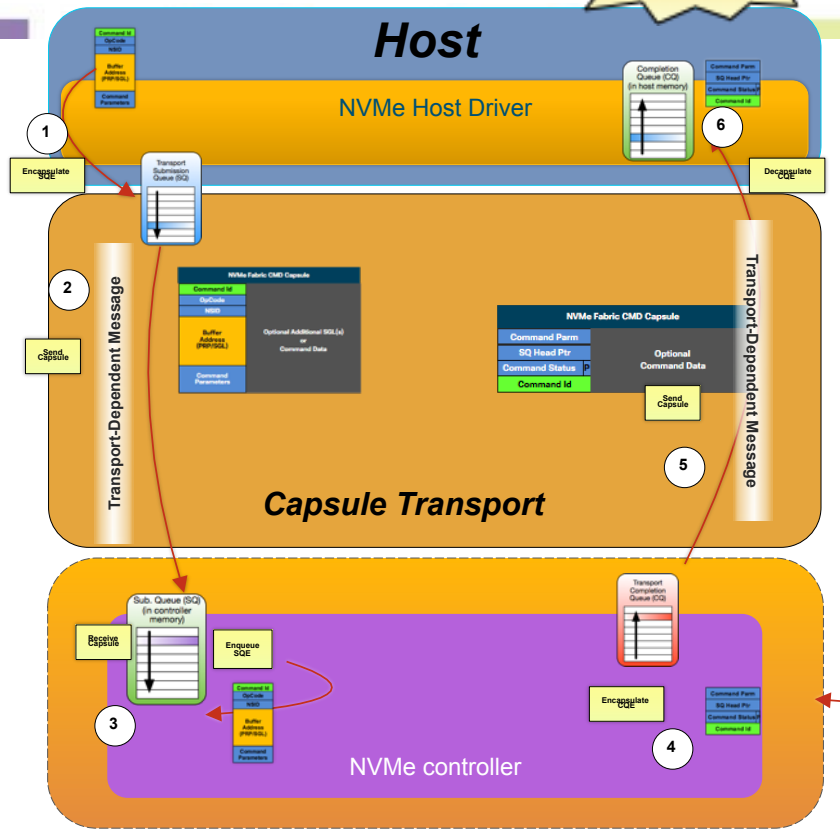


- What is a Binding?
  - "A specification of reliable delivery of data, commands, and responses between a host and an NVM subsystem for an NVMe Transport. The binding may exclude or restrict functionality based on the NVMe Transport's capabilities"
- I.e., it's the "glue" that links all the pieces above and below (examples):
  - SGL Descriptions
  - Data placement restrictions
  - Data transport capabilities
  - Authentication capabilities

NVMe-oF Queuing Interface to Transports

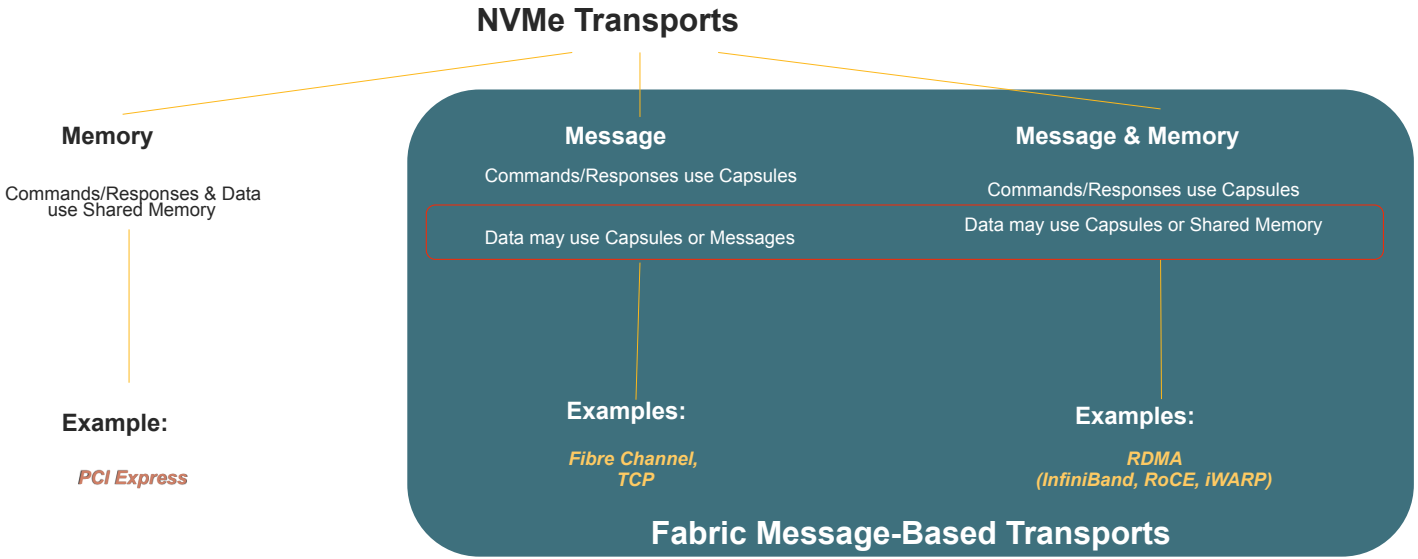
Queues, Queues, and MORE Queues!

- Host Driver encapsulates SQE into an NVMe-oF Command Capsule
- NVMe-oF capsule is sent to the network/ Fabric
- Fabric enqueues the SQE into the remote NVMe SQ
- Controller encapsulates CQE into an NVMe-oF Response Capsule
- NVMe-oF Response capsule is sent to the network/Fabric
- Fabric enqueues the CQE into the host CQ



NVMe Transport Models

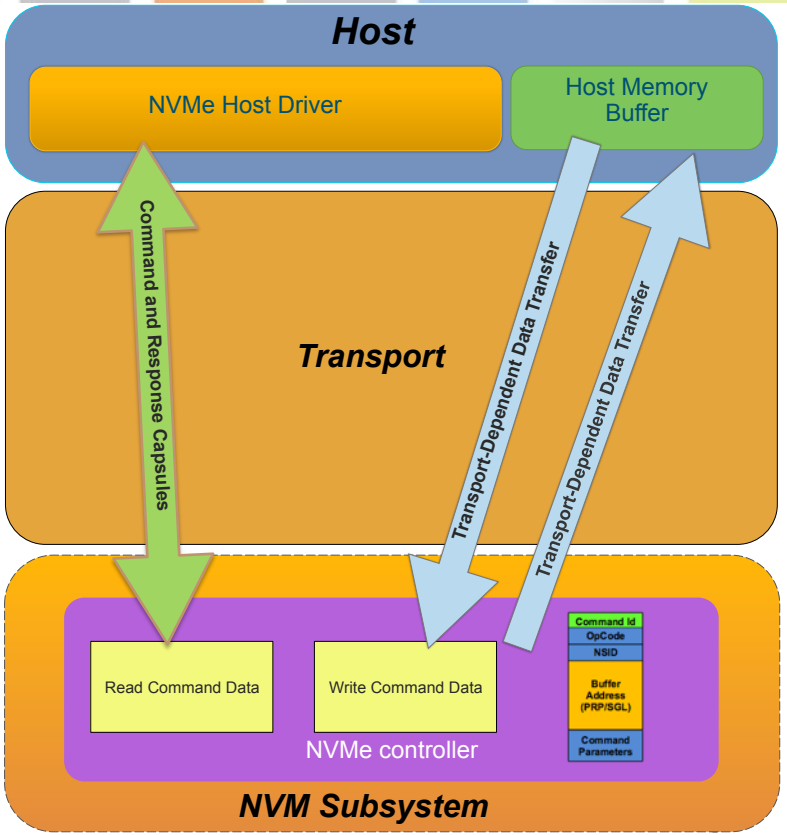
- NVMe is a Memory-Mapped, PCIe Model
- Fabrics is message-based, shared memory is optional
- In-capsule data transfer is *always* message-based



Capsule = Encapsulated NVMe Command/Completion within a transport Message
Data = Transport data exchange mechanism (if any)

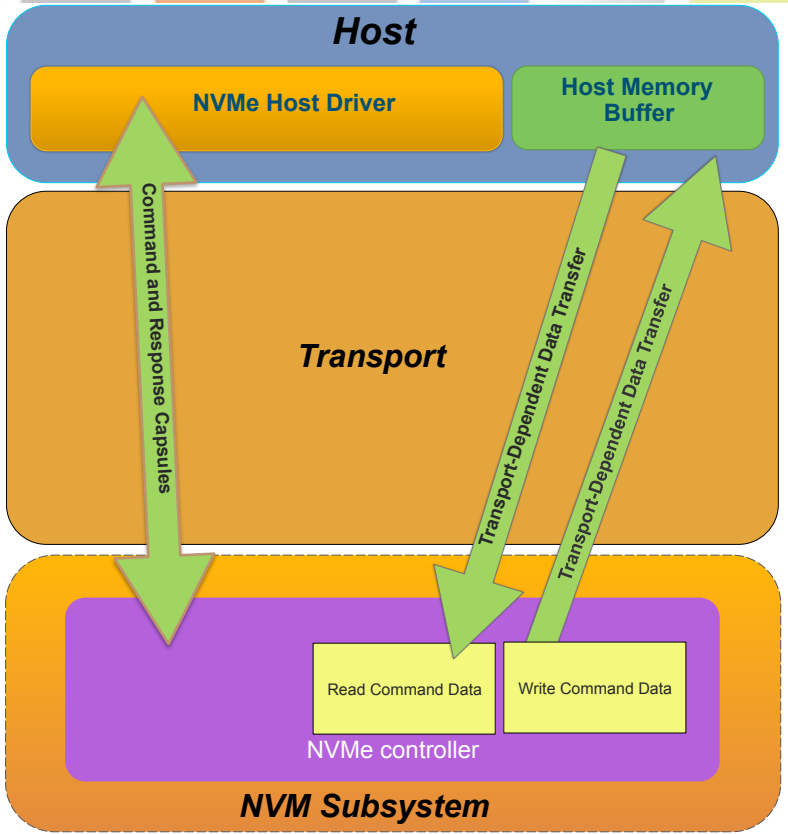
NVMe-oF Data Transfers (Memory + Messages)

- **Command** and **Response Capsules** are transferred using messages
- **Data** is transferred using memory semantics
- Data transfer operations are transport specific; examples
  - RDMA: RDMA_READ/ RDMA_WRITE Operations
  - Similar to PCIe (PCIe Memory Read/Write Requests)



NVMe-oF Data Transfers (Messages Only)

- **Command Capsules, Response Capsules** transferred using messages
- **Data** is transferred using messages
- Data transfer operations are transport specific; examples
  - Fibre Channel: FCP Exchanges
  - TCP: H2C and C2H PDUs



SNIA NSF | NETWORKING STORAGE

Introducing NVMe™/TCP

Why NVMe/TCP?

- Ubiquitous - runs on everything everywhere…
- Well understood - TCP is probably the most common transport
- High performance - TCP delivers excellent performance scalability
- Well suited for large scale deployments and longer distances
- Actively developed - maintenance and enhancements are developed by major players
- Inherently supports in-transit encryption

- The Network Infrastructure may not be well designed to meet Latency requirements
- Some Application may have strict performance thresholds than vanilla TCP can provide

NVMe-oF Capsules and NVMe/TCP PDUs

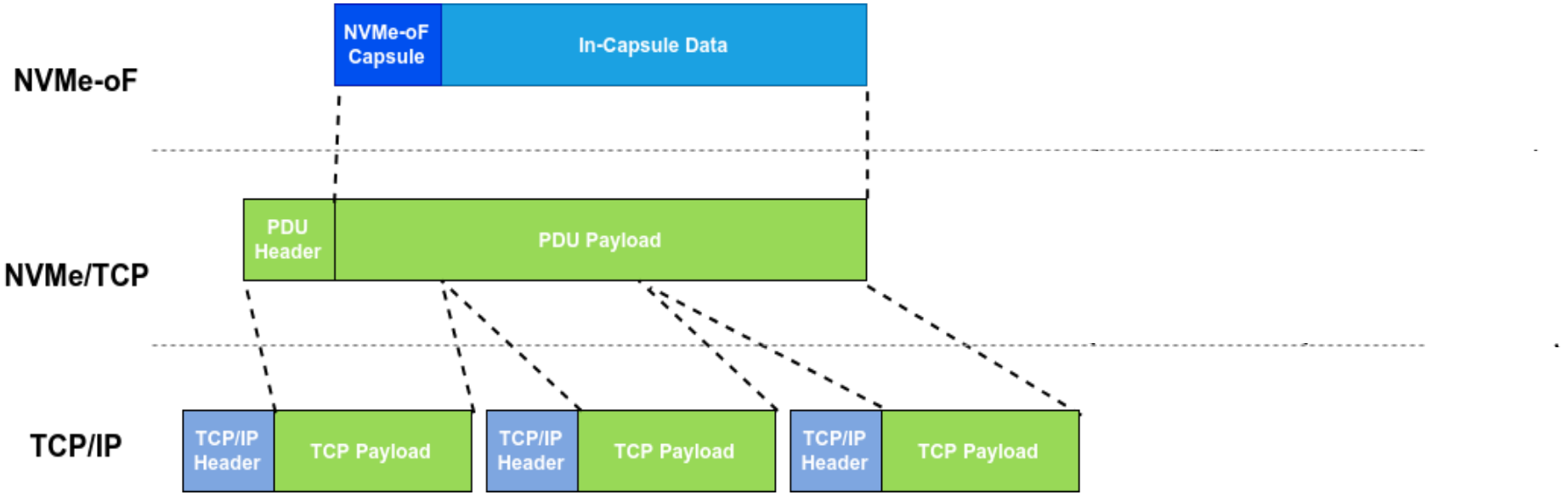
NVMe-oF Capsule

- Represents an NVMe Command and/or Completion
- In practice Command Capsules look like NVMe Commands with different Data buffer SGL and optionally In-Capsule Data
- Response Capsules are NVMe completions

NVMe/TCP PDU

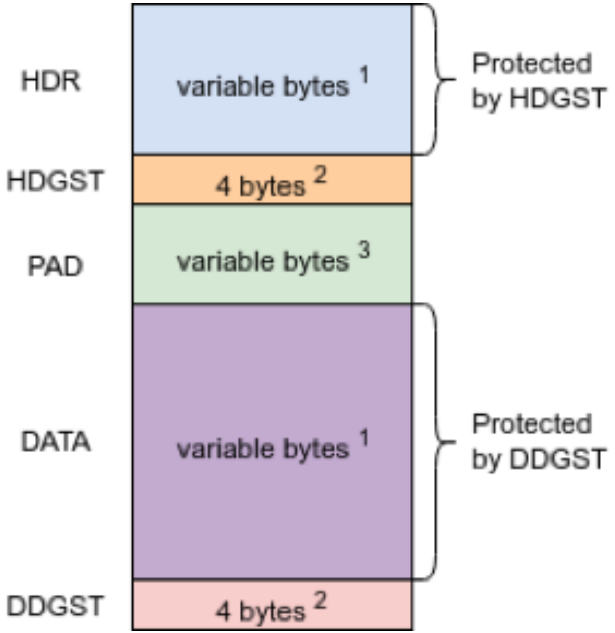
- Encapsulates every protocol message (NVMe-oF Capsules, Data, Ready-To-Transfer, Connection Initialization, Connection Termination)
- In the most generic form will include: Header, Digests, Padding and Data

NVMe/TCP PDUs and Capsules Segmentation

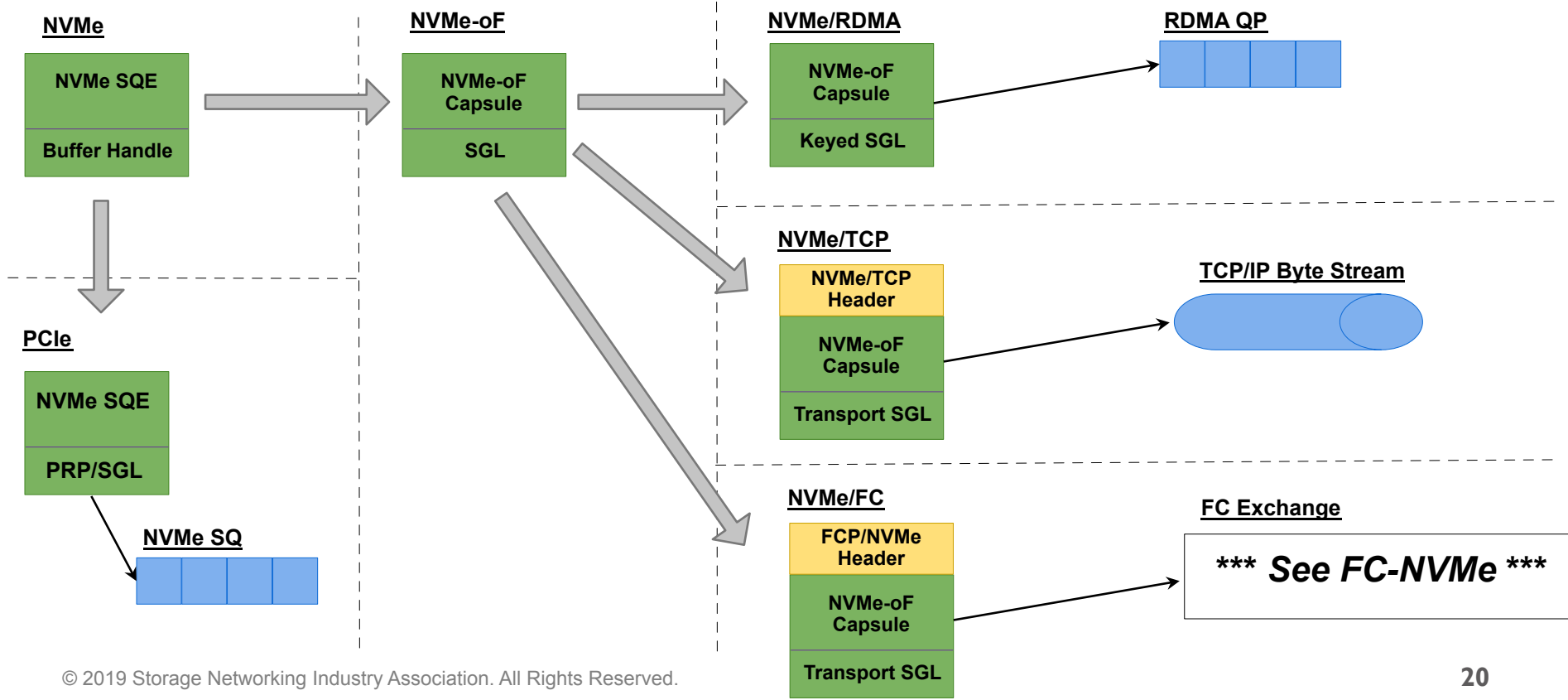


NVMe/TCP Protocol Data Unit (PDU)

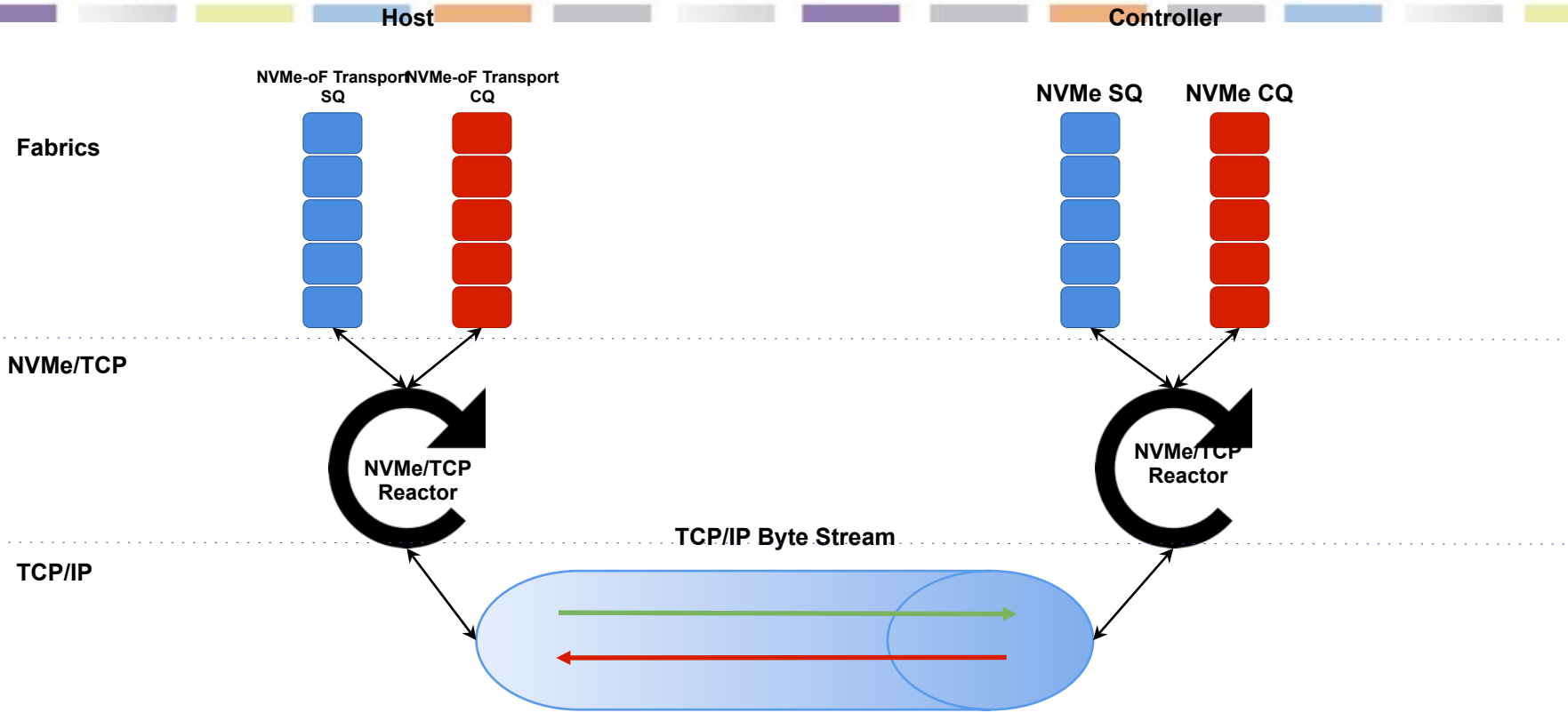
- NVMe-oF Capsules and Data are encapsulated in PDUs
  - PDU structure varies per PDU type
  - 8-byte Common Header
- Variable length PDU specific header
- PDUs optionally contain Header and/or Data digest protection
- PDUs contain optional PAD used for alignment enhancements



NVMe-oF Encapsulation Process

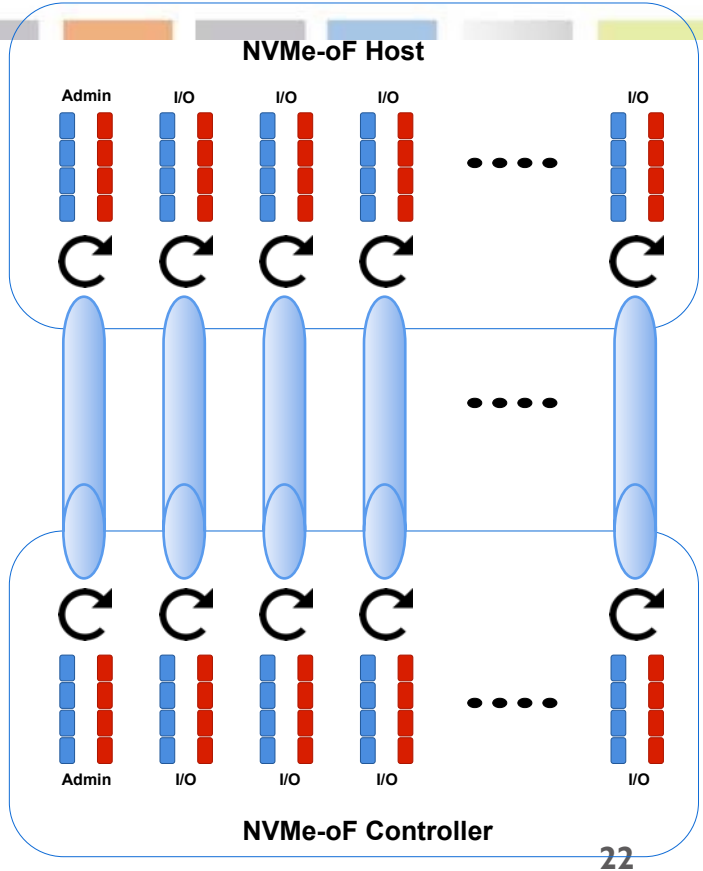


NVMe/TCP Queue Command Processing



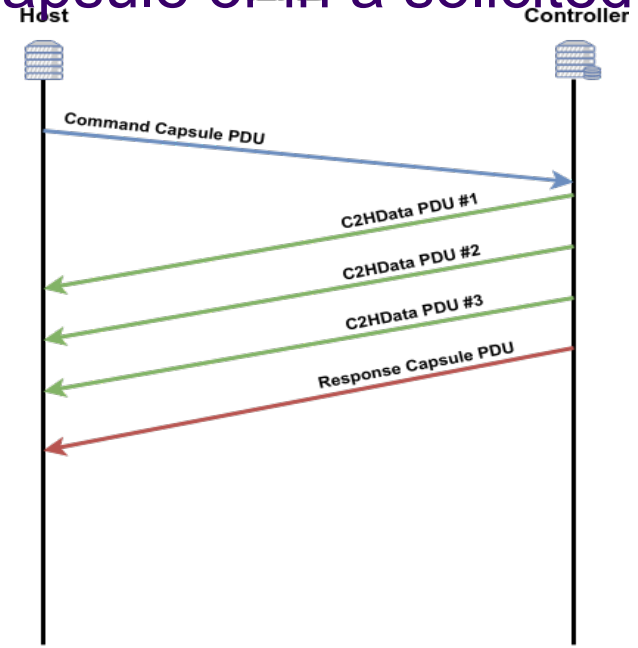
NVMe/TCP Controller Association

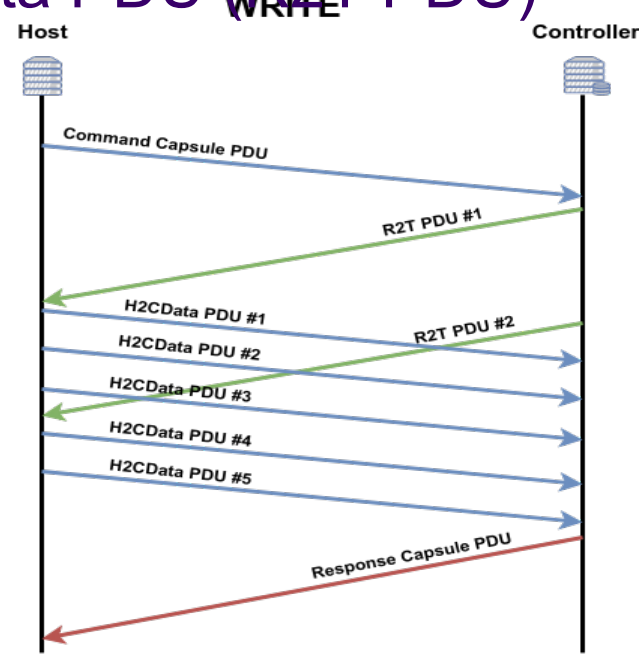
- Controller association maps 1x1 NVMe queue to a TCP connection
  - No controller-wide sequencing
  - No controller-wide reassembly constraints
  - No shared state across NVMe queues and TCP connections accessed in the “hot” path
  - Each NVMe queue (and its backing TCP connection) can be assigned to a separate CPU core.
- Connection binding is performed in NVMe-oF connect time (binding



NVMe/TCP I/O Flow

- Host to Controller Data (H2CData) can come in-capsule or in a solicited H2CData PDU (R2T PDU)





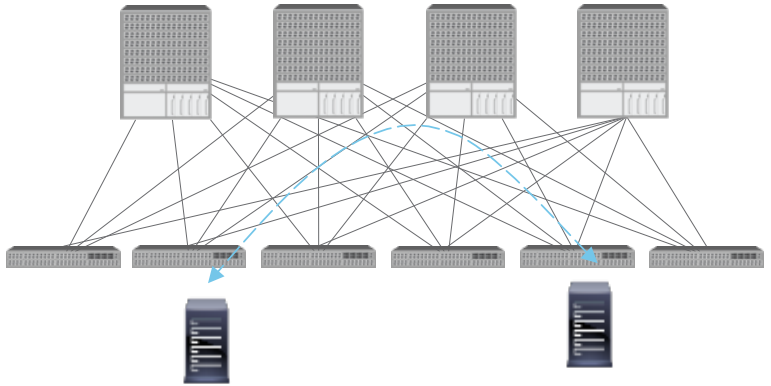
Potential Issues with NVMe/TCP

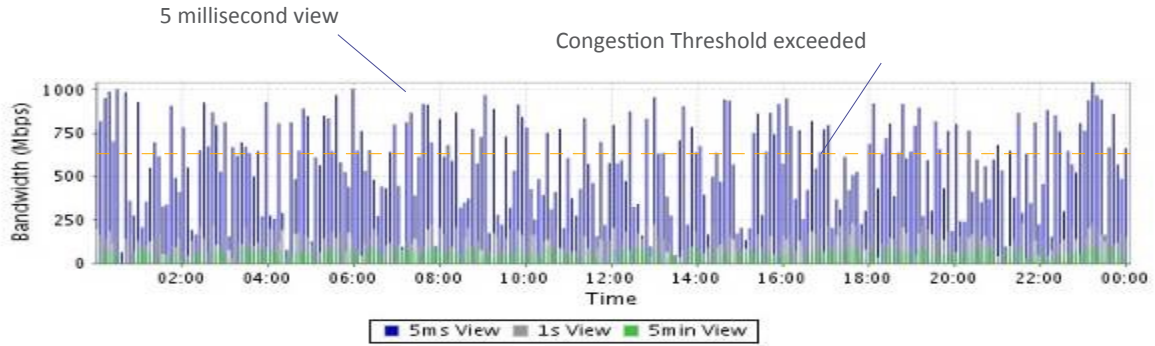
Absolute Latency is Higher than RDMA?	**Yes, by several microseconds. This only matters if the application is sensitive to such latency differences**
Head-of-Line blocking can cause higher latencies?	**- Protocol breaks up large transfers - Read/Write queue separation helps - NVMe priority-based queue arbitration can help as well**
Incast could be an issue?	**Common potential issue with TCP/IP. A lot of attention is being put by both switch vendors, NIC vendors and TCP/IP OS developers.**
Lack of HW acceleration	**NVMe/TCP is designed to be efficient also when running in SW. Offload devices are coming as well.**

Some Advances in TCP

Build Networks To Optimize for the Application

- You do not need to and should not be designing a network that requires a lot of buffering
- Capacity and over-subscription is not a function of the protocol (NVMe, NAS, FC, iSCSI, CEPH) but of the application I/O requirements

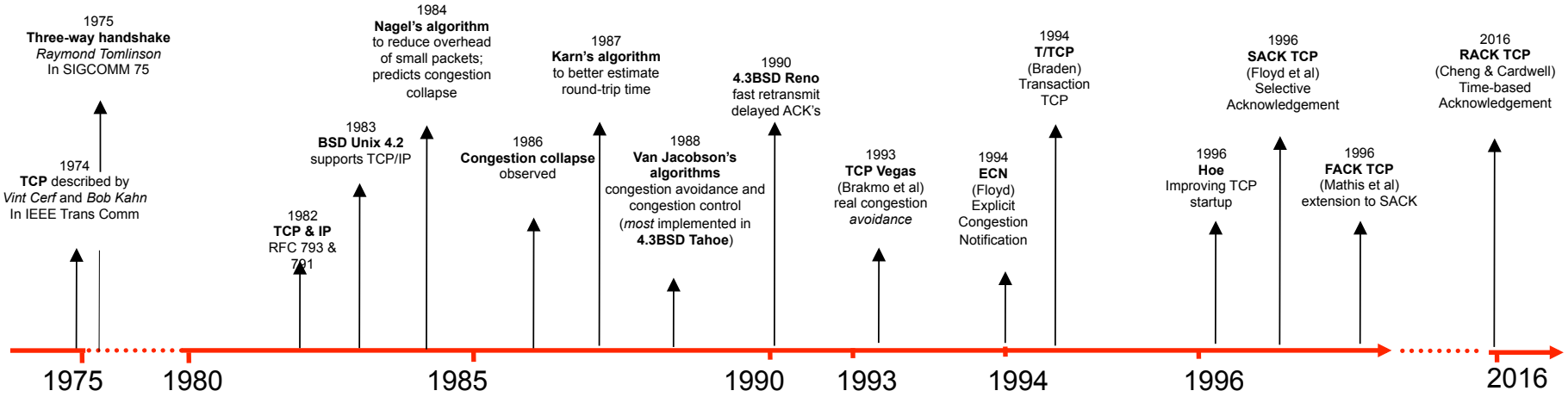




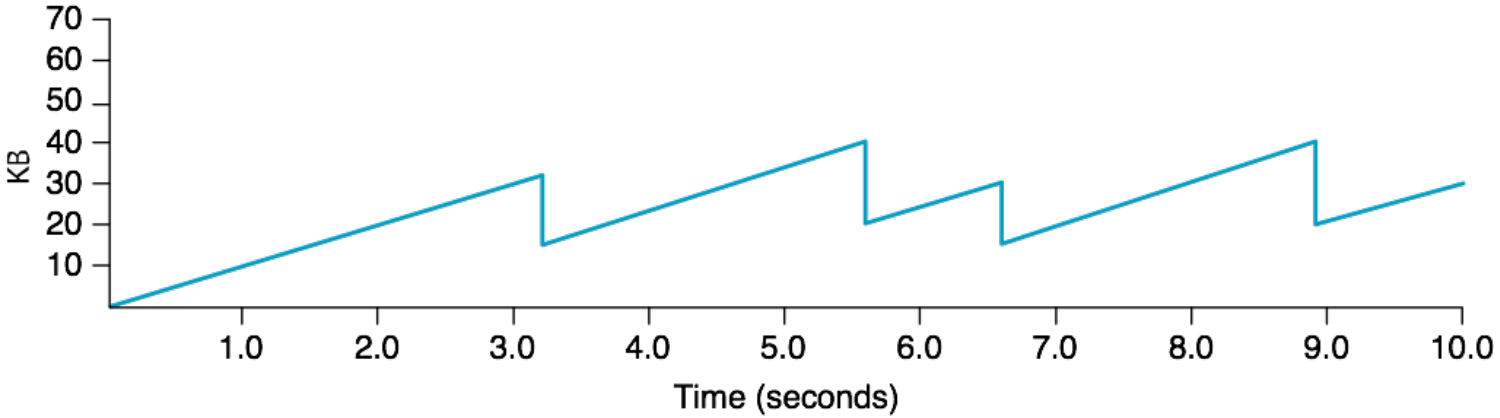
Data Center Design Goal: Optimizing the balance of end to end fabric latency with the ability to absorb traffic peaks and prevent any associated traffic loss

Not All TCP Stacks Are Created Equal

- TCP stacks that rely on drops (most common stacks) are the ones that require proper network buffering
- Newer stacks looking at RTT or other feedback loops to monitor throughput are optimizing for 'zero buffer' networks
  - Importance of accurate RTT estimators:
    - Low RTT - unneeded retransmissions
    - High RTT - poor throughput
- It helps to know which stacks you are using



Typical TCP Sawtooth Pattern



- TCP flows have the tendency to grab as much bandwidth as available
- TCP commonly uses retransmission as a signal for network congestion
- A healthy dose of retransmission helps TCP Congestion Control

The Network Buffer Paradox

Not enough buffer – poor utilization of link BW

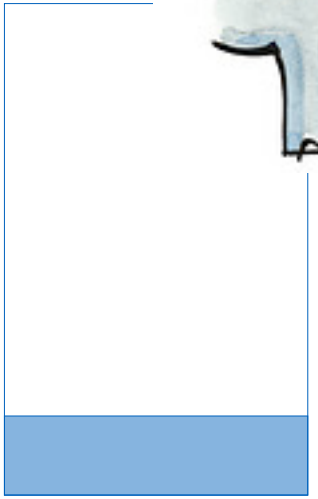
Too much buffer – increased latency

Just enough buffer – best possible link utilization and latency

What's the "Goldilocks" amount?

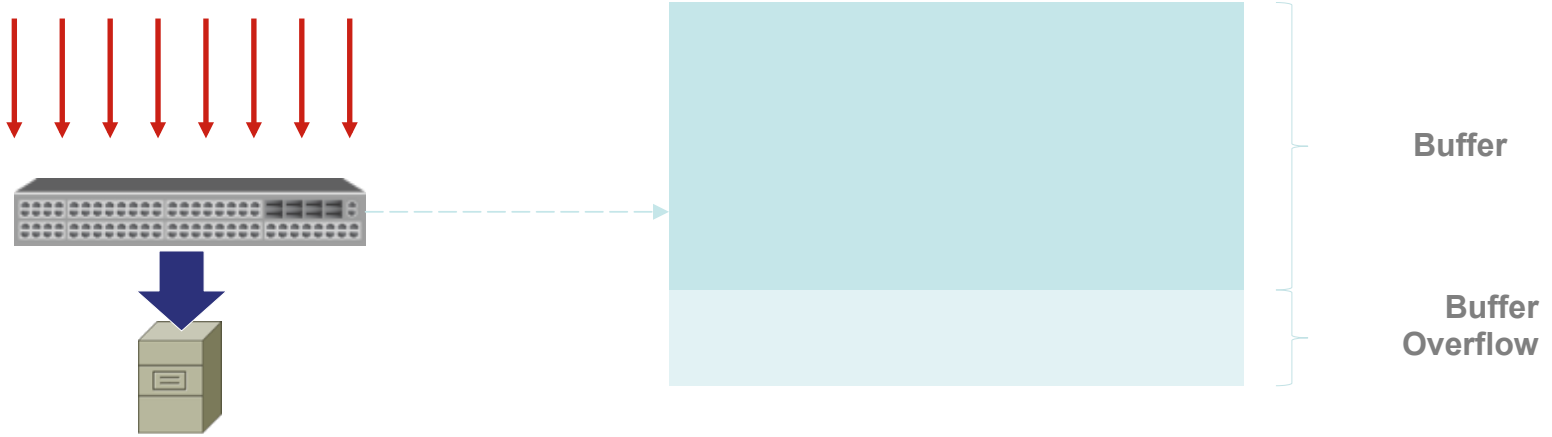
Buffering the Data Center

- Large, “elephant flows” can overrun available buffers
- 2 methods of solving this problem:
  - Increase buffer sizes in the switches
  - Notify the sender to slow down before TCP packets get dropped

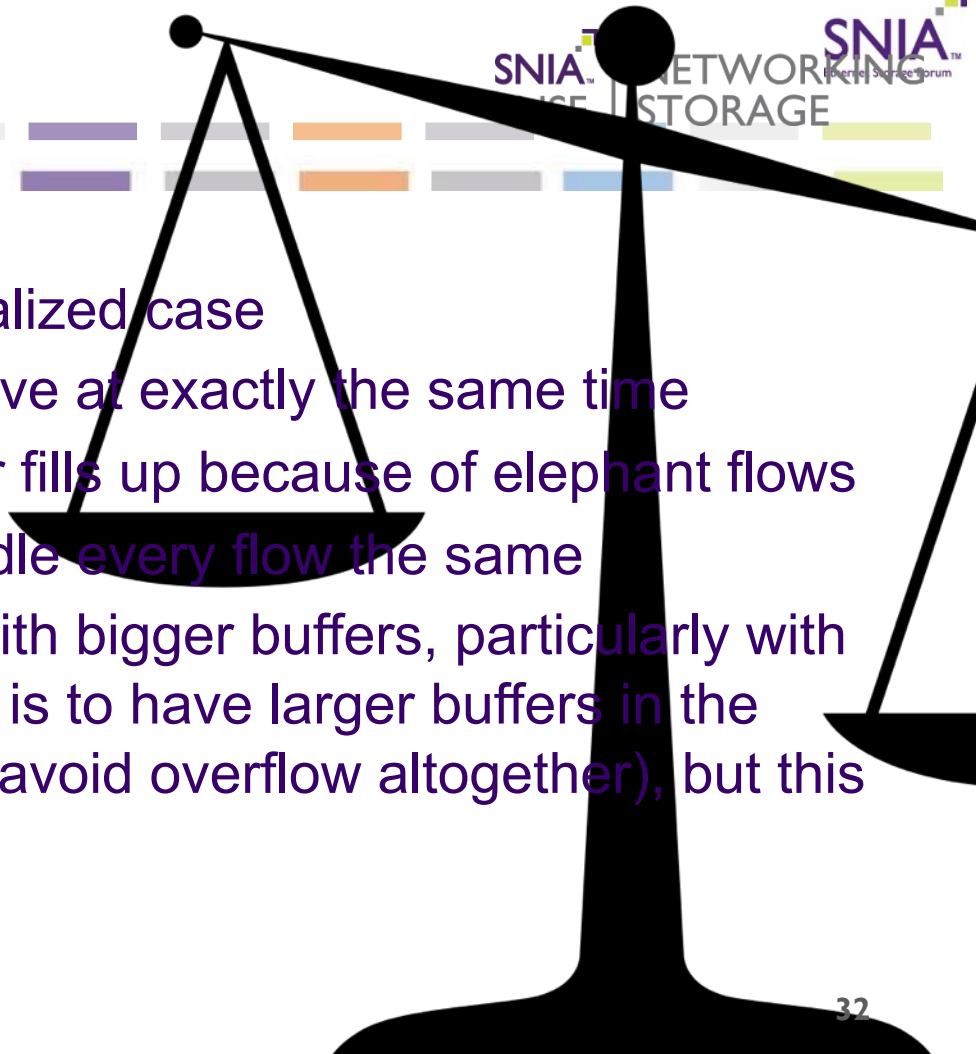


Understanding TCP Incast

- Synchronized TCP sessions arriving at common congestion point (all sessions starting at the same time)
- Each TCP session will grow window until it detects indication of congestion (packet loss in normal TCP configuration)
- All TCP sessions back off at the same time



Incast Collapse



- Incast collapse is a very specialized case
- It would need every flow to arrive at exactly the same time
- The problem is more the buffer fills up because of elephant flows
  - Historically, buffers handle every flow the same
- It could potentially be solved with bigger buffers, particularly with short frames, and one solution is to have larger buffers in the switches than the TCP Incast (avoid overflow altogether), but this adds latency

Solution 2: Telling the Sender to Slow Down

- Instead of waiting for TCP to drop packets and then adjust flow rate, why not simply tell the sender to slow down before the packets get dropped?
- Technologies such as Data Center TCP (DCTCP) uses Explicit Congestion Notification, “ECN”) instruct the sender to do just this
- Dropped packets are the signal to TCP to modify the flow of packets being sent in a congested network

DCTCP

- Congestion indicated quantitatively (reduce load prior to packet loss)
- React in proportion to the extent of congestion, not its presence.
  - Reduces variance in sending rates, lowering queuing requirements.

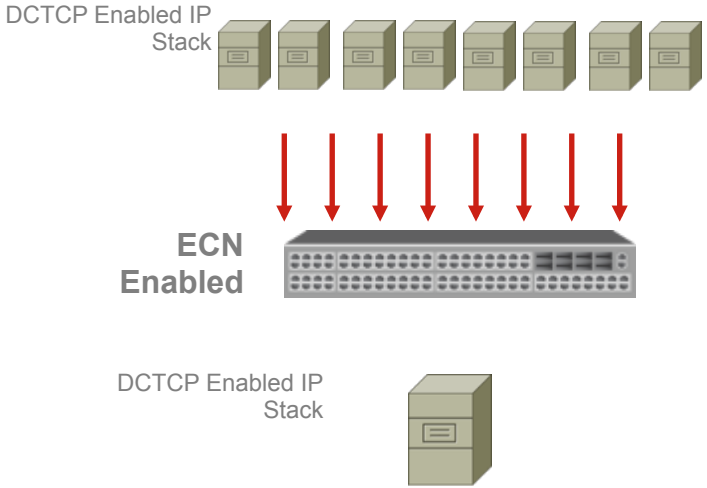
ECN Marks	TCP	DCTCP
1 0 1 1 1 1 0 1 1 1	Cut window by 50%	Cut window by 40%
0 0 0 0 0 0 0 0 0 1	Cut window by 50%	Cut window by 5%

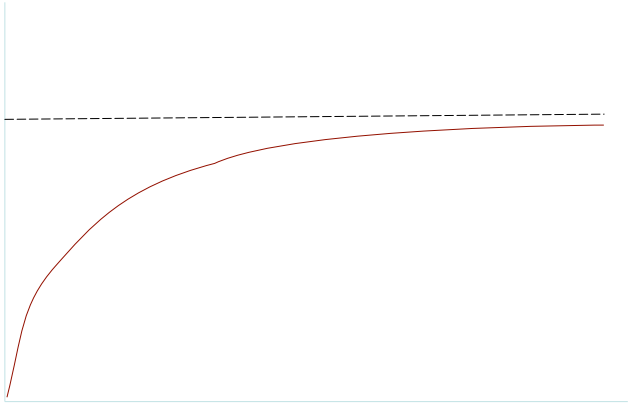
- Mark based on instantaneous queue length.
  - Fast feedback to better deal with bursts.

Source: Data Center TCP (DCTCP), SIGCOMM 2010, New Dehli, India, August 31, 2010.

DCTCP and Incast Collapse

- DCTCP will prevent Incast Collapse for long lived flows
- Notification of congestion via ECN prior to packet loss
  - Sender gets informed that congestion is happening and can slow down traffic
  - Without ECN, the packet would have been dropped due to congestions and sender will notice this via TCP timeout



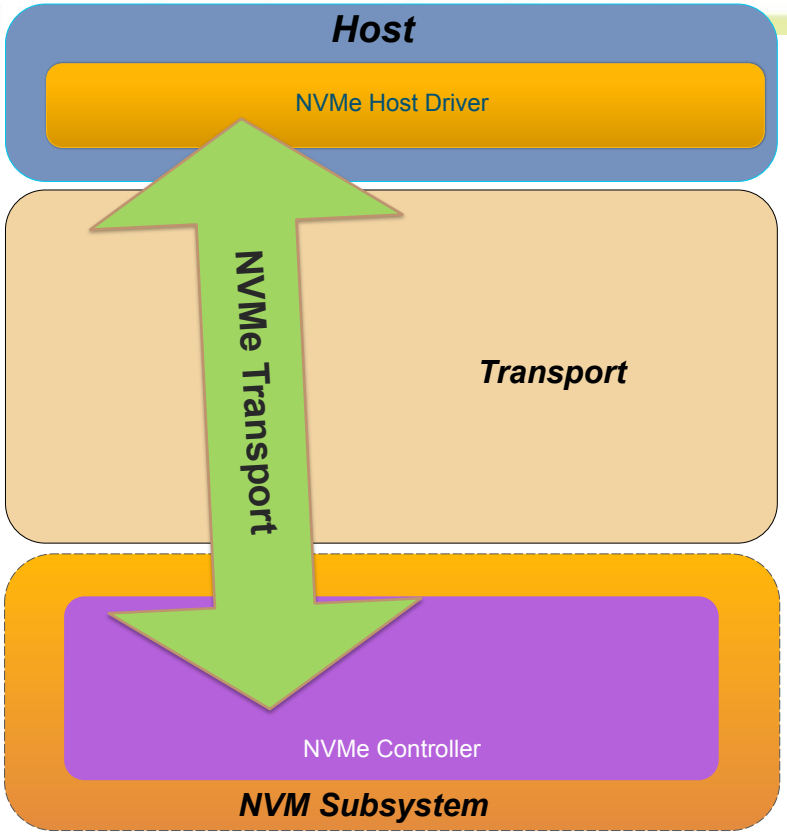


SNIA NSF | NETWORKING STORAGE

Bringing it All Together

Now.. Back To NVMe-oF

- NVMe™/TCP
  - Efficient NVMe transport using well-understood TCP networks
- Remember…
  - This **arrow**, innocuous as it may seem, is where all the network goodness and badness happens
  - We want this **arrow** to be as short and as reliable as possible
  - This is where modern TCP advancements such as DCTCP, ECN, and other technologies can make life easier

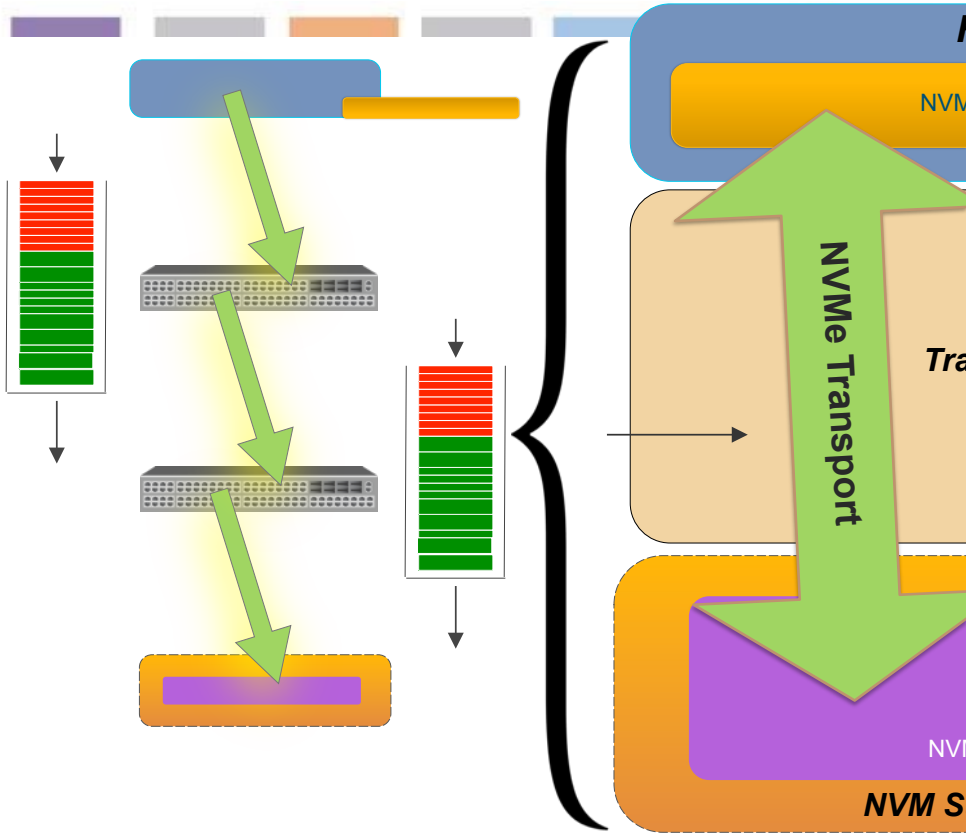


The Curse of Large Buffers

NVMe queuing is dependent upon the ongoing communication between the host and NVMe controller

Inserting large buffers in-between the host and the storage subsystem:

- Increases latency
- Adds potential points of failure/ delay
- Significantly reduces efficiency



Summary

- NVMe over Fabrics is not just “NVMe on a Stick”
  - Each major transport has it’s own methods of handling high-throughput in congested environments
- NVMe™/TCP
  - Well-understood, ubiquitous, high-performance and highly scalable NVMe transport
  - More similar to NVMe™/FC than NVMe™/RDMA (despite being Ethernet)
  - Available now!
- Understanding how storage reacts to the network will be the difference between a good storage solution and a nightmare one

For More Information

- NVM Express, Inc - **nvmexpress.org**
- NVMe for Absolute Beginners - **https://blogs.cisco.com/datacenter/nvme-for-absolute-beginners**
- NVMe-oF for Absolute Beginners - **https://jmetz.com/2018/08/nvme-over-fabrics-for-absolute-beginners/**
- Welcome NVMe™/TCP to the NVMe Family of Transports
  - **https://nvmexpress.org/welcome-nvme-tcp-to-the-nvme-of-family-of-transports/**
- Latest Developments in NVMe/TCP
  - **https://www.snia.org/sites/default/files/SDC/2018/presentations/NVMe/Grimberg_Sagi_Latest_Developments_with_NVMe_TCP.pdf**
  - **https://www.lightbitslabs.com/blog/**
- Data Center TCP: **https://people.csail.mit.edu/alizadeh/papers/dctcp-sigcomm10.pdf**
- Everything You Wanted To Know About Storage: Part Teal - The Buffering Pod
  - **https://www.brighttalk.com/webcast/663/241275**

- J's Twitter: **@drjmetz**
- Sagi's Twitter**: @sagigrim**

Upcoming NSF Webcasts

Networking Requirement for Hyperconvergence

February 5, 2019

Register at: https://www.brighttalk.com/webcast/663/341209

The Scale-Out File System Architecture Overview

February 28, 2019

https://www.brighttalk.com/webcast/663/346111

After This Webcast

- Please rate this webcast and provide us with feedback
- This webcast and a PDF of the slides will be posted to the SNIA Networking Storage Forum (NSF) website and available on-demand at www.snia.org/library
- A full Q&A from this webcast, including answers to questions we couldn't get to today, will be posted to the SNIA-NSF blog: sniansfblog.org
- Follow us on Twitter @SNIANSF

SNIA NSF | NETWORKING STORAGE

Thank You