STORAGE DEVELOPER CONFERENCE
SDC 21
BY Developers FOR Developers

Virtual Conference
September 28-29, 2021

A SNIA Event

NVMe-oF: Protocols & Transports Deep Dive

Not quite so

Presented by Dr. Joseph L White, Dell Fellow

Original Abstract

Block storage access across Storage Area Networks (SANs) has an interesting protocol and transport history. The NVMe-oF transport family provides storage administrators with the most efficient and streamlined protocols so far leading to more efficient data transfers and better SAN deployments. In this session we will explore some of the protocol history to set the context for a deep dive into NVMe/TCP, NVMe/RoCE, and NVMe/FC. We will then examine network configurations, network topology, QoS settings, and offload processing considerations. This knowledge is critical when deciding how to build, deploy, operate, and evaluate the performance of a SAN as well as understanding end to end hardware and software implementation tradeoffs.

Revised Abstract

We will briefly explore some of the history of block storage access across Storage Area Networks (SANs) looking at what the protocols have in common.

We will then look at the transport properties and behavior tradeoffs between of NVMe/RoCE and NVMe/TCP using simplified examples and discuss offloading in a Data Processing Unit.

A SAN (Storage Area Networking) Protocols Timeline

▪ Parallel SCSI:	1980s – 1990s
▪ FCP/Fibre Channel, FCIP:	1990s – 2000s – 2010s – 2020s- Today
▪ Early IP/Ethernet based transports:	Late 1990s – Early 2000s
▪ several switched & networked parallel SCSI approaches	
▪ mFCP(UDP), iFCP(TCP)	
▪ iSCSI(TCP):	2000s – 2010s – 2020s – Today
▪ FCoE(Ethernet):	Late 2000s – 2010s – Today
▪ NVMe-oF(RoCE, TCP, FC, Infiniband, iWARP):	2010s – 2020s – Today

- Common SAN protocol properties
  - Core purpose: transfer data from data from one system to another across a network or fabric or issue commands to a system.
  - Commands and Transfers are broken into packets with an end-to-end protocol on top of a transport protocol
  - A method for sequencing commands and allowing multiple commands
  - Within the systems the communicating endpoints are logical or virtualized on top of the hardware
  - A set of services, controllers, and orchestrators are used to manage the systems and their connection
  - Sets of Admin or control commands to assist with discovery, connection setup/management, and operations

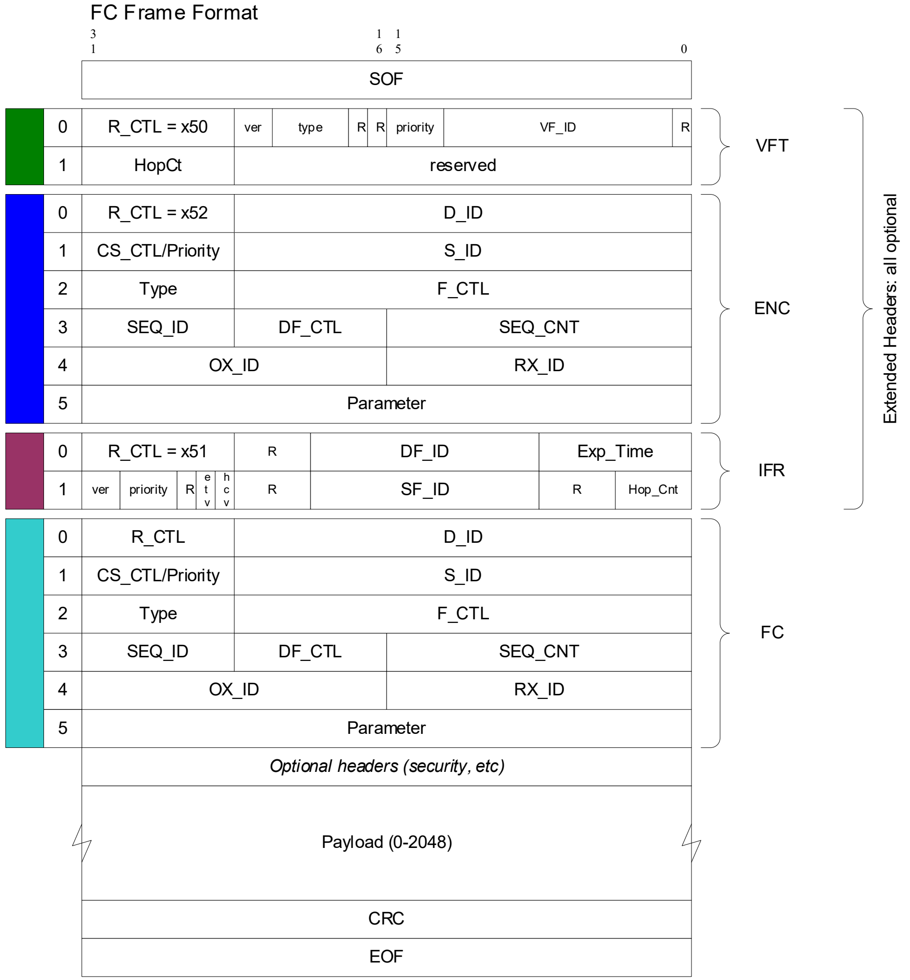
STORAGE DEVELOPER CONFERENCE SDC 21

SAN Packet Examples

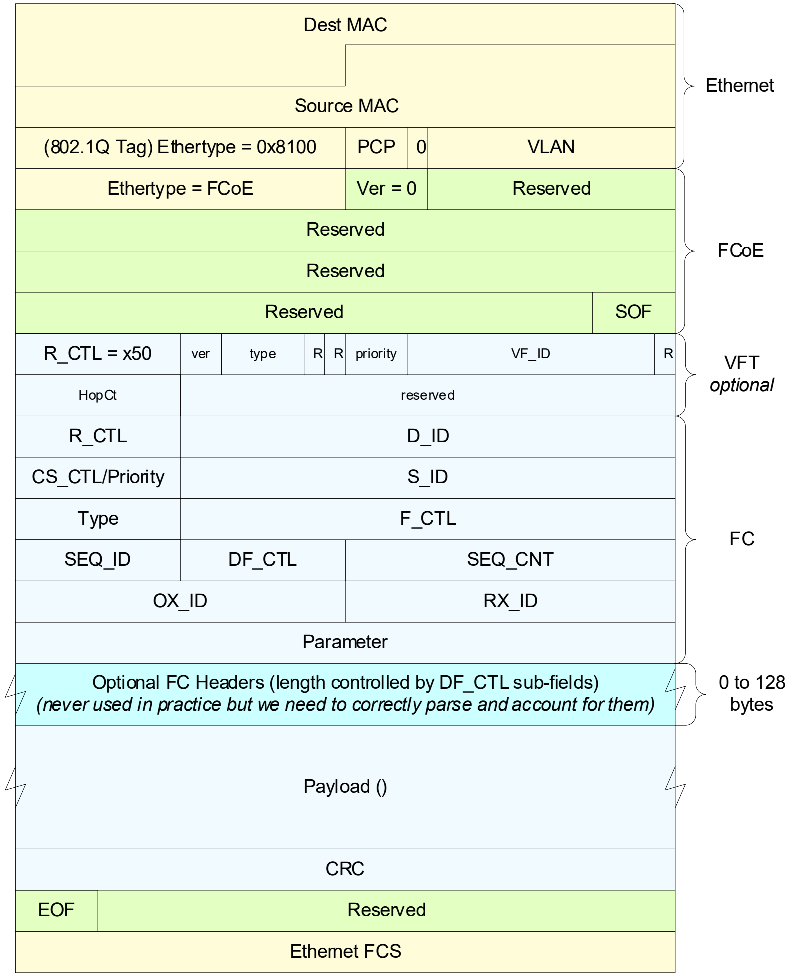
Using Fibre Channel as an example but all protocols follow the same core patterns

- Packet Properties
  - Addressing
  - Sequencing
  - Control flags, bits, words
  - Quality of Service
  - Tunneling or Encapsulation
  - Packet Type Identification Codes
  - Payload
    - Command
    - Data
    - Status
  - Data integrity
  - Data Confidentiality

FC:

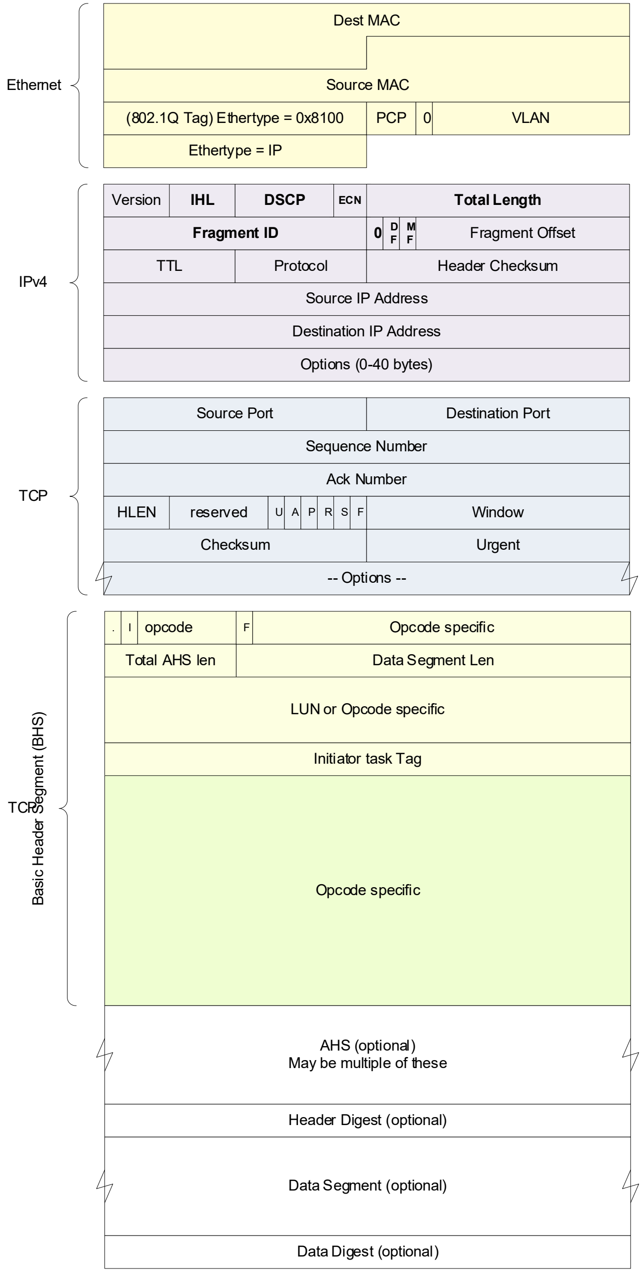


FCoE:



Dest MAC
Source MAC
(802.1Q Tag) Ethertype = 0x8100
PCP
0
VLAN
Ethernet
Ethertype = FCoE
Ver = 0
Reserved
Reserved
Reserved
Reserved
SOF
FCoE
R_CTL = x50
ver
type
R
R
priority
VF_ID
R
HopCt
reserved
VFT
optional
R_CTL
D_ID
CS_CTL/Priority
S_ID
Type
F_CTL
SEQ_ID
DF_CTL
SEQ_CNT
OX_ID
RX_ID
Parameter
FC
Optional FC Headers (length controlled by DF_CTL sub-fields)
(never used in practice but we need to correctly parse and account for them)
0 to 128 bytes
Payload ()
CRC
EOF
Reserved
Ethernet FCS

iSCSI:



Ethernet
Dest MAC
Source MAC
(802.1Q Tag) Ethertype = 0x8100
PCP
0
VLAN
Ethertype = IP
IPv4
Version
IHL
DSCP
ECN
Total Length
Fragment ID
0
DF
MF
Fragment Offset
TTL
Protocol
Header Checksum
Source IP Address
Destination IP Address
Options (0-40 bytes)
TCP
Source Port
Destination Port
Sequence Number
Ack Number
HLEN
reserved
U
A
P
R
S
F
Window
Checksum
Urgent
-- Options --
TCP
Basic Header Segment (BHS)
I
opcode
F
Opcode specific
Total AHS len
Data Segment Len
LUN or Opcode specific
Initiator task Tag
Opcode specific
AHS (optional)
May be multiple of these
Header Digest (optional)
Data Segment (optional)
Data Digest (optional)

NVMe-oF

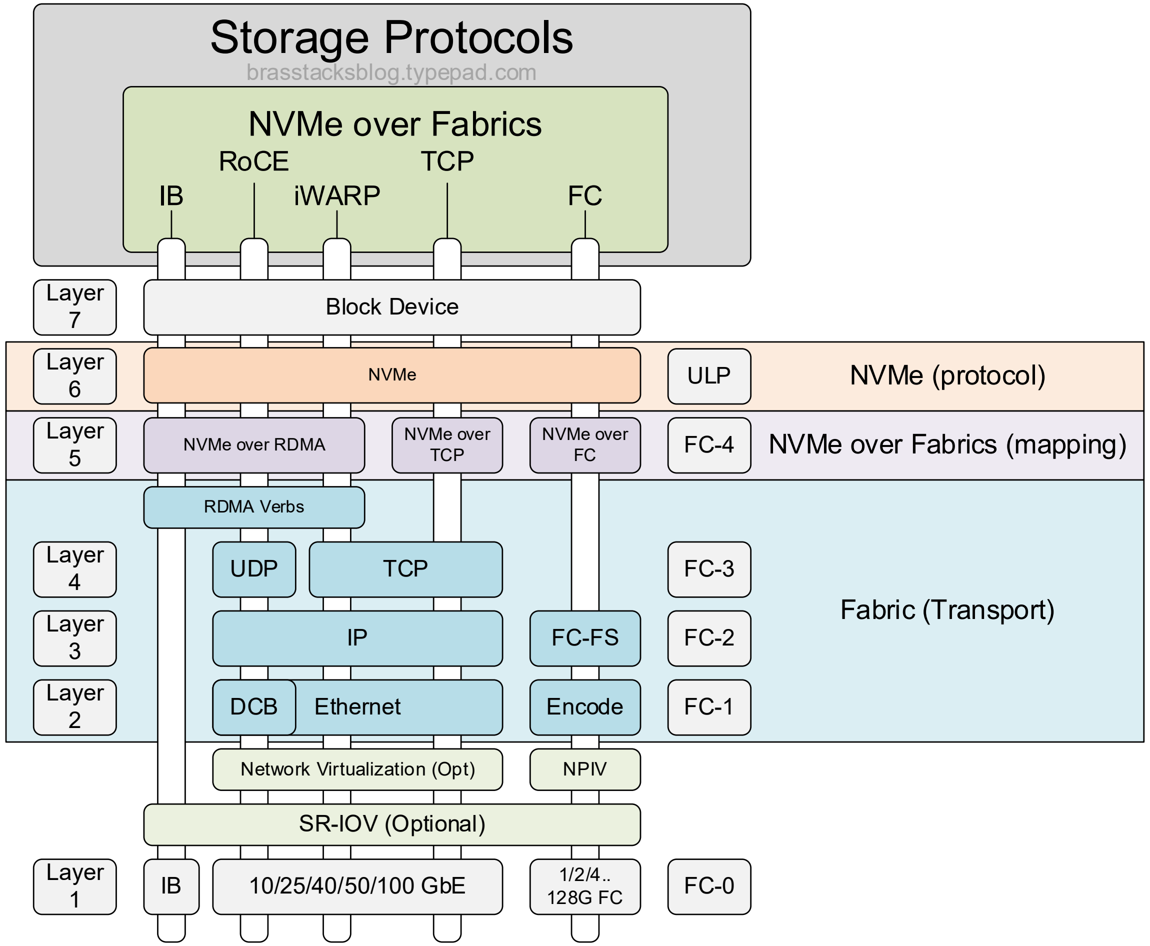
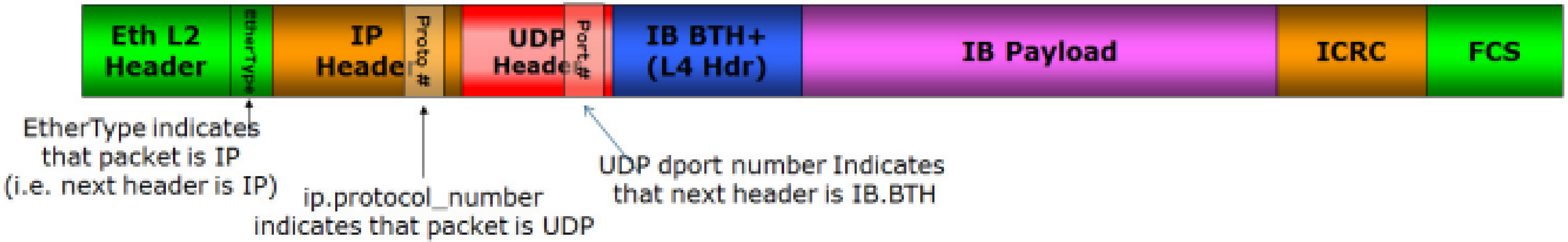


Diagram used with permission from brasstacksblog.typepad.com

- Core NVMe-oF Properties
  - Common Command Formats
  - Multiple command sets
  - Multiple transport Mappings
  - Rich discovery and connection setup
  - Efficient Read and Write Command formats
  - Queue pairs for concurrency, multiple commands

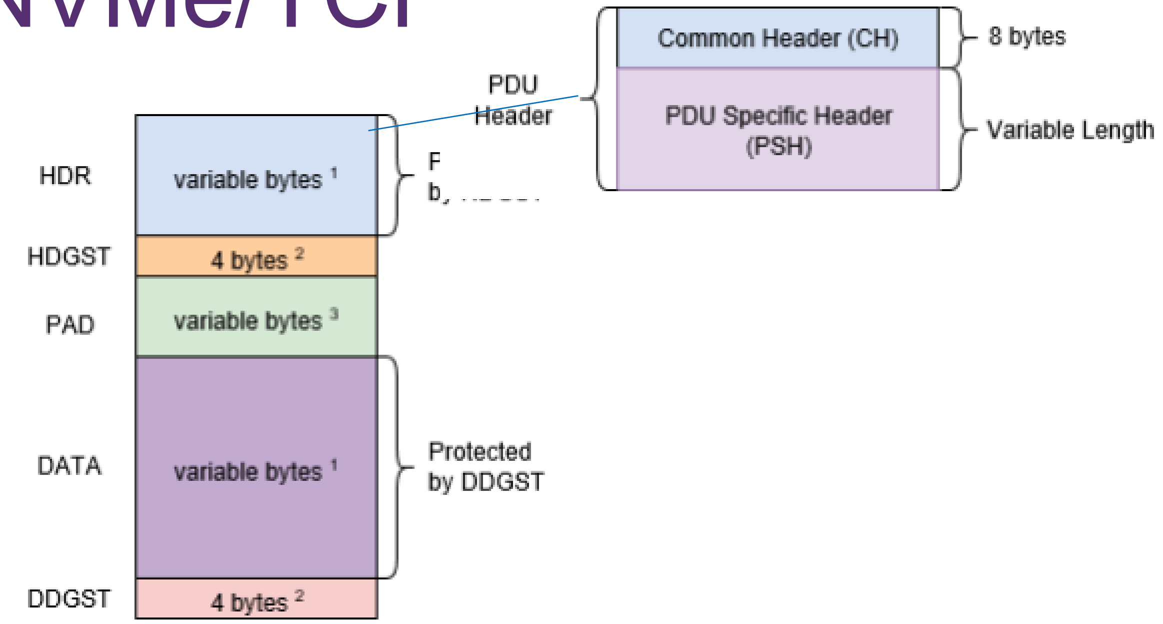
NVMe/RoCEv2

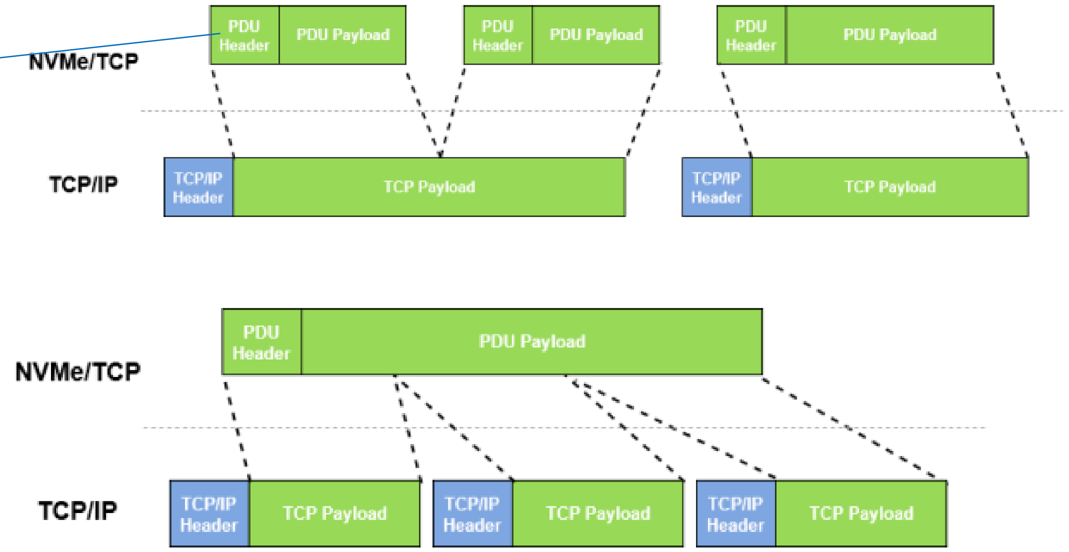
Framing diagram from https://cw.infinibandta.org/document/dl/7781



- Uses UDP as a transport
- Requires lossless Ethernet via link level flow control or Priority Flow Control
- Each UDP datagram payload starts with IB header followed by IB payload
- NVMe commands and data are carried within the IB payload
- The IB layer uses RDMA (RDMA_SEND, RDMA_READ, RDMA_WRITE, etc) for command and data transfer

NVMe/TCP

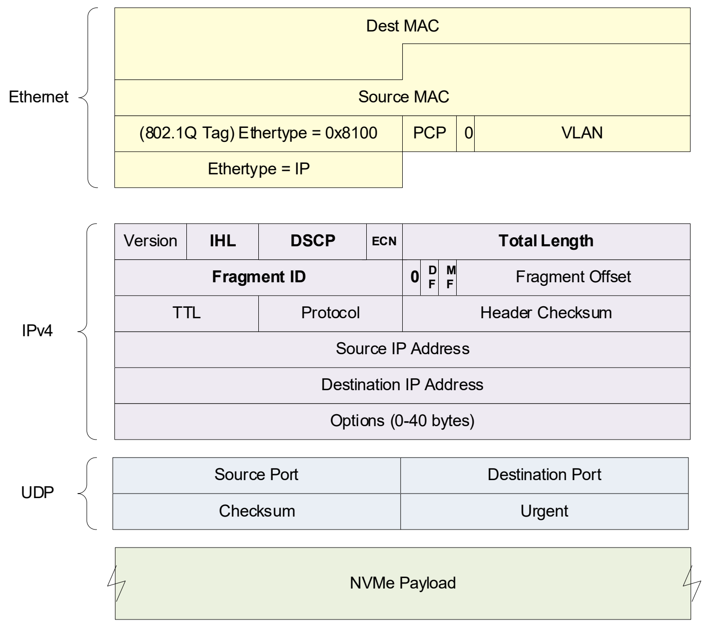




- Treats a stream of NVMe Protocol Data Units as a byte stream segmented into packets for transport
- No alignment of underlying NVMe protocols to the TCP/IP packets on the wire
- Runs over lossy or lossless networks

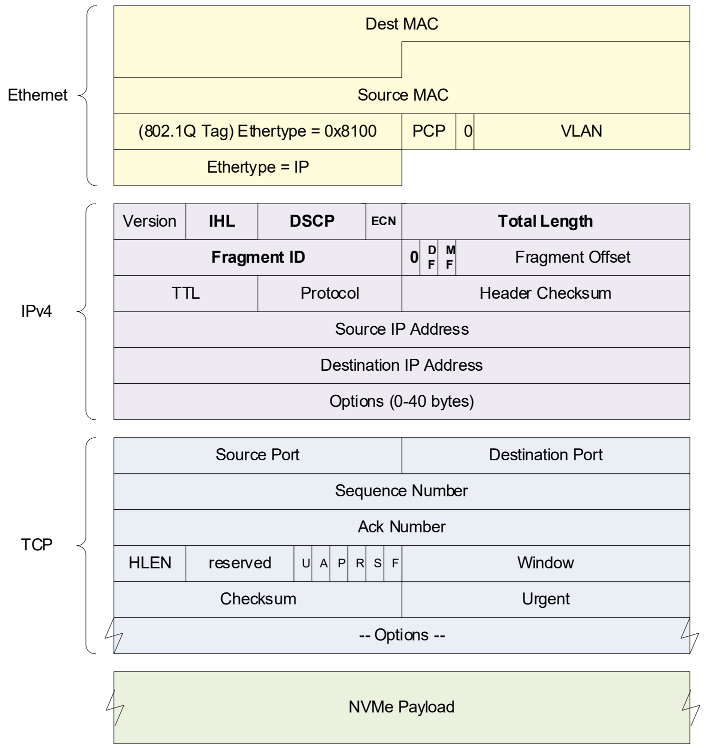
Framing diagrams from https://nvmexpress.org/wp-content/uploads/NVMe-TCP-Transport-Specification-1.0a-2021.07.26a-Ratified-1.pdf

UDP Transport



- Straightforward datagram protocol
- Connectionless
- No guaranteed delivery
- No flow control mechanisms

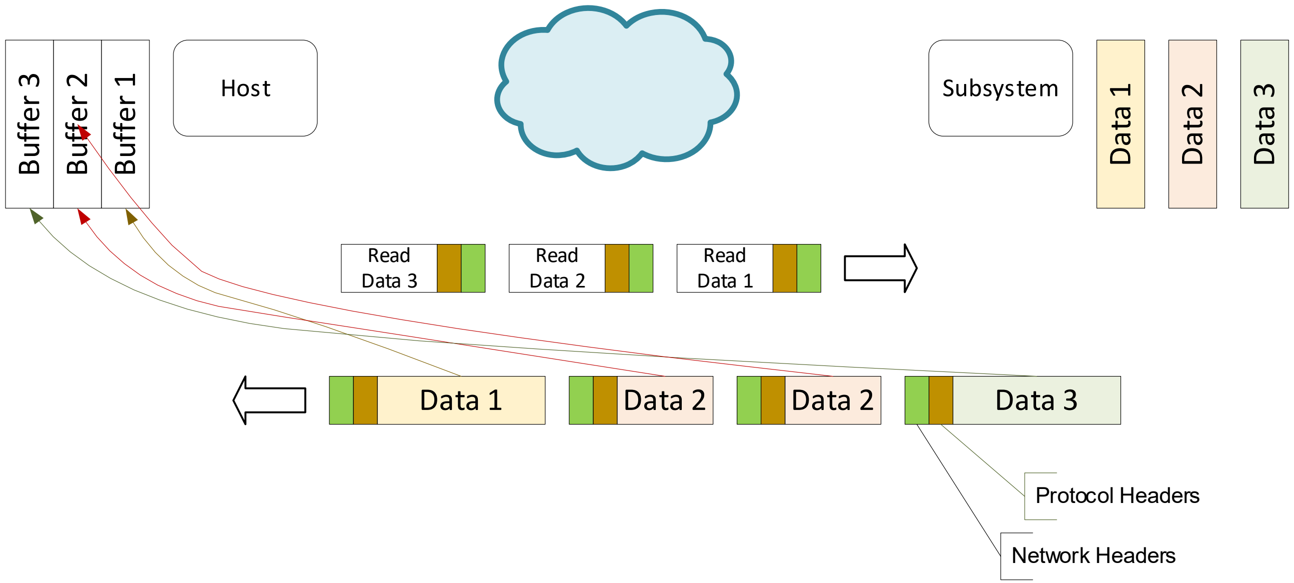
TCP Transport



- Byte stream protocol
- Connection oriented
- Guaranteed delivery
- Flow control built into the protocol
- Windows, sequence numbers and acknowledgements must be handled

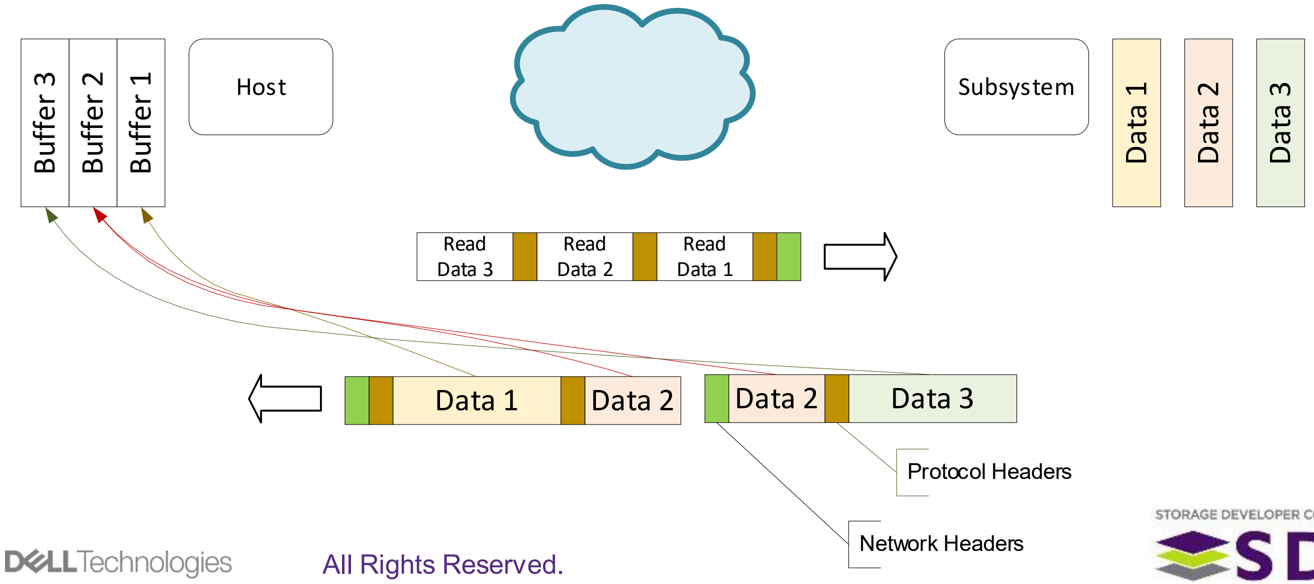
Let’s Discuss these approaches with a simplified example:

- Datagram Example
  - Each command is in a separate packet
  - Each data transfer can be handled directly based on the headers
  - Any drops are recovered at the command layer

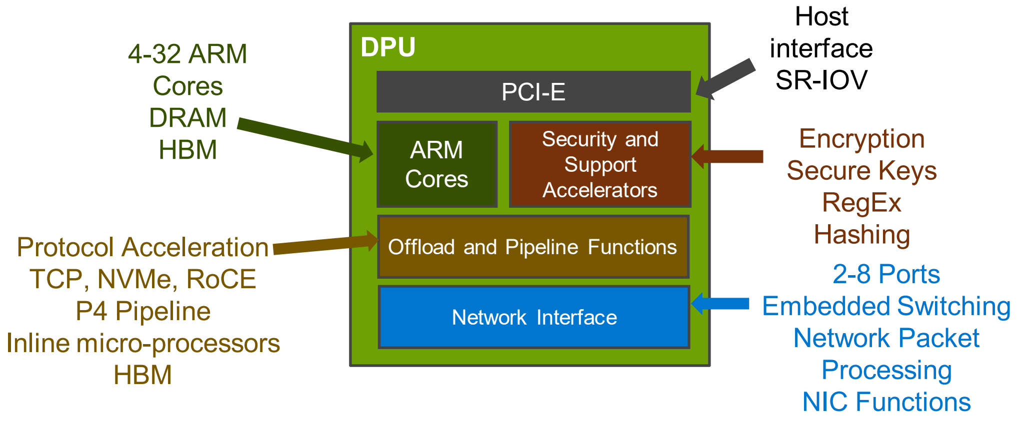


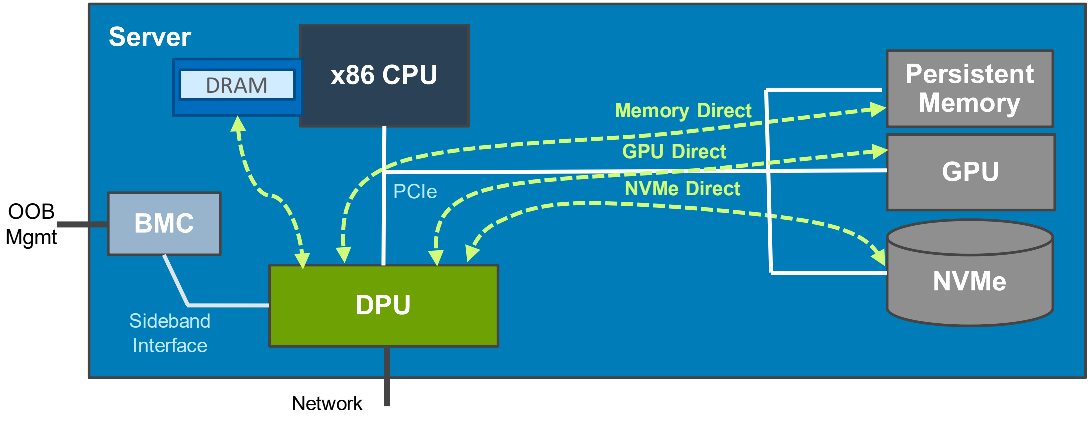
- *In each case the commands and data transferred are the same*

- Byte Stream Example:
  - The subsystem must walk through the packet to extract the protocol headers and commands
  - Zero Copy TCP data handling is typically needed for performance
  - The host must wait for the second response packet to complete Data 2
  - If packet 1 were dropped and retransmitted, the host would not know how to handle the second piece of Data 2 or how to find the header for Data 3 until packet 1 was retransmitted



Data Processing Units and Protocol Acceleration





- Supports dedicated NVMe-oF processing and acceleration in Hardware pipelines
- DPU can directly place or send data from Host DRAM via DMA or across PCIe to local devices.
- **Both** TCP (byte stream) and RDMA/RoCE (datagram) transports **can be offloaded**
  - The Accelerators and pipelines must be constructed correctly for transport
  - Reference the previous slide on the core differences on WAY the transports move the commands and data

Thanks!

Please take a moment to rate this session.

Your feedback is important to us.

Please take a moment to rate this session.

Your feedback is important to us.