Native Booting using NVMe over Ethernet Fabrics

Doug Farley (Dell Technologies)
Douglas.Farley@dell.com

Lenny Szubowicz (Red Hat)
lszubowi@redhat.com

Team:

Tony Rodriguez (Dell Technologies)
Curtis Ballard (HPE)
Stephanie Yuan (HPE)

John Leung (Intel)
Maciej Rabeda (Intel)
Phil Cayton (Intel)

Vincent Zimmer (Intel)
Rob Davis (NVIDIA Networking)

Murali Rajagopal (VMware)
Wenhua Liu (VMware)

Abstract

NVMe over Fabrics™ (NVMe-oF™) lacks a native capability for boot from Ethernet. We will Introduce a joint model to address boot from NVMe-oF/TCP, its impact to the kernel and the entire ecosystem, and collect feedback from the Linux community. This architectural model is being designed for standardization by the appropriate committees (e.g., NVM Express™ or UEFI™ Forum).

Summary / Agenda

- NVMe over Fabrics (NVMe-oF) lacks a native capability for boot from Ethernet.
  - To address: multi-vendor group including Dell Technologies, RedHat, Nvidia Networking, Intel, VMware, and HPE investigating possible methods to enable boot from NVMe-oF in a manner designed for standardization and maintainability.
  - Underlying design intended to be implementation independent.
  - However, focused on UEFI and NVMe/TCP as first implementation.
- Short overview of NVMe-oF
- Design Direction
  - ACPI™ Table
  - UEFI Device Path
  - Pre-Boot and OS Handoff
  - Multipath

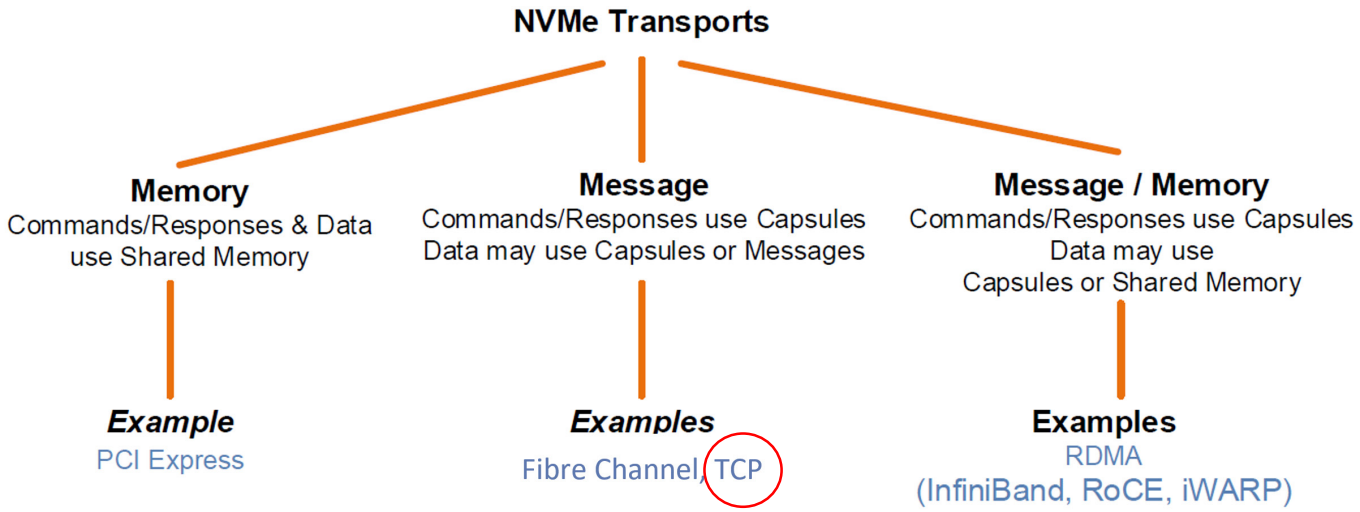
Terminology and Acronyms, Further Reading

- DC – NVM Discovery Controller, hosts NVM Subsystem Indicies
- DLP – NVMe-oF Discovery Log Page
- ESP – EFI System Partition; i.e. (/boot/efi)
- Namespace - NVMe flavor of SCSI LUN, block device representation
- NID – Namespace Identification Descriptor; a globally unique ID (def. NVMe CNS 03h)
- PDU – Protocol Data Unit
- SQE – Submission Queue Entry
- SGL – Scatter Gather List

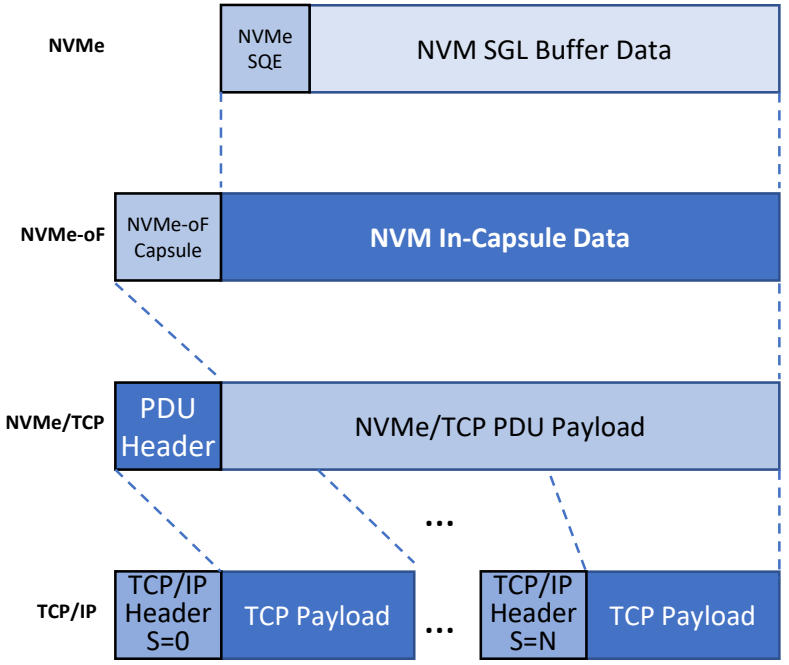
NVMe-oF/TCP Overview (tl;dr version)

- NVMe/TCP uses NVMe-oF capsule messaging on TCP/IP to provide remote NVMe Access across a network "Fabric“, similar to what iSCSI does today for SCSI.
- NVMe-oF replaces PCIe bus scanning with a discovery service mechanism, similar to iSCSI sendtargets.
- NVMe-oF was merged officially in v4.8, and NVMe/TCP in v5.0 with both Initiator/Host (nvme) and Subsystem/Target (nvmet) code in tree.

NVMe-oF/TCP Overview (proto stack)



Memory to Messaging Encapsulation



Boot from NVMe-oF Solution

- Starting with a general UEFI solution for easiest broad adoption.
  - UEFI + NVMe/TCP as easiest route, no HW needed.
- Learn from iSCSI; create a new ACPI Table (“nBFT”) coupled with a new UEFI Device Path.
  - Known NVMe-oF DLP carried in Discovery Sub-table to recreate programmed topology.
- Security (Keys/Creds) can be exchanged via in-band redfish+tls via DMTF (WIP Mechanism*) because EFIvar or ACPI inappropriate for Creds.
- Support for logical grouping of paths for multipath.

* DMTF DSP-IS0013, ‘DRAFT’ Redfish Host Credential Bootstrapping, https://www.dmtf.org/standards/wip

ACPI Table and Features Overview

- Define NVMe-oF Boot Table “nBFT” – like iSCSI iBFT; found via the XSDT.
- Contains entries for unique Adapter, Namespace, Security and Discovery elements.
- All required and optional parameters present.
- Supports boot (i.e. ESP) and non-boot namespaces.
- Features like:
  - (CNS 03h) NID, Discovery Controllers, IP/Port, NVM Subsystem Name, Logical Multipath Group ID, Transport Security Settings.

UEFI Device Path

PciRoot(0)/Pci(19|0)/Mac(001320F5FA77,0x01)/IPv4(192.168.0.100,TCP,Static,192.168.0.1)/NVMEoF(nqn.1991-05.org.nvme:nvmeoftarget-nvmeofdisk-target,4eff7f8e-d353-4e9b-a4ec-deea8eab84d7,<tbd>)/HD(1,GPT,15E39A00-1DD2-1000-8D7F-00A0C92408FC,0x22,0x2710000)

- SubNQN – Maximum 224 byte Unique Subsystem Identifier including null termination in compliance with the NVM Express Base Spec.
- NID – Globally Unique Identifier defined in NVM Express Base Spec
  - Adapter, Subsystem, Namespace, and Security Parameters – All found in the nBFT

Booting: Pre-OS EFI Bootloader Flow

- Before EFI ExitBootServices()
- EFI boot variable(s)
  - Device path, with new NVMe-oF Device Path Nodes
  - File specification of bootloader, or null
- nBFT via ACPI XSDT
  - Grub2 bootloader (for example)
    - Reads config file, etc.
  - Relies on its ImageHandle parameter
    - EFI_LOAD_FILE_PROTOCOL
    - Loads kernel and initramfs/initrd, from different partition on same device, e.g. /boot
    - EFI_BLOCK_IO_PROTOCOL
- The bootloader can be unaware that it was loaded from an NVMe namespace over TCP using a novel device path

Booting: Linux OS Flow

- After EFI ExitBootServices()
- Kernel, kernel command line, initramfs/initrd
- Needs to mount the run-time root file system
- Device discovery is going on in parallel
  - Normal kernel nvme, nvme-tcp run-time drivers
  - Boot-time nvme discovery daemon
    - Kernel provides nBFT via sysfs (new)
    - Device path from EFI BootCurrent boot variable (new)
- Switch to target root file file system and continue boot

Multipath

- Pre-Boot, Multipath is just fall-through loading; i.e try next UEFI Boot variable – inc. UEFI Device Path
  - Some devices may not have a EFI System Partition; i.e. ‘/’ and ‘/usr’ on namespace.
  - Optimizations possible.
- OS Runtime (Post-Boot)
  - Assuming something reads nBFT via sysfs, multipathd will then have all discovered namespaces, no new changes needed.
- Discovery Services
  - nBFT contains a Discovery Sub-table, consisting of NVMe-oF DLP Contents
  - Provides the breadcrumb content to reformulate Ethernet L2, L3 and NVM Subsystem connectivity for the OS

Future areas of investigation and work

- Enhance multipathd to better leverage nBFT
- Support for NVMe-oF Security in Kernel Driver still missing today...

OS Runtime/Installer drive add flow

- **OS or OS Installer Needs to Add or Modify NVMe-oF Boot Targets specified in UEFI Config:**
- Assumptions:
  - UEFI NVMe-oF Implementation.
  - A local BMC with a RedFish endpoint who has secure access to UEFI Secure Attributes exists.
  - DMTF-DSP-IS0013 Proposal for bootstrapping and establishing a OS to BMC secure channel exists and OS has Support/Tools.
  - OS/Installer has 'curl' like equivalent or more advanced Redfish Client.
- Flow:
  1. OS determines it needs UEFI/ACPI NVMe-oF Boot enabled and is supported by local UEFI instance.
  2. OS sets up in-band mechanism to get to redfish and creds via DMTF-DSP-IS0013
  3. OS makes redfish call to see if populating NBFT is supported.
  4. OS, via redfish, adds new Host/NIC/Initiator Parameters.
  5. OS, via redfish, adds new NVM Subsystem and Namespace Target Parameters, including the NVMe NID.
  6. OS, via redfish, adds new NVME Subsystem Security parameters if required.
  7. OR {
     1. OS, via EFI Boot Variable mechanic, tells BIOS it wants a new boot parameter with the NID and SubNQN as specified above, at a specific Boot Order.
     2. OS, via redfish, might set the Boot order via existing BiosAttribute mechanism(s).

     }
  8. OS must update any local config that mirrors this nBFT data, since ACPI nBFT would only be expected to update at boot time.

* DMTF DSP-IS0013, 'DRAFT' Redfish Host Credential Bootstrapping, https://www.dmtf.org/standards/wip

Wrap-up

- This architectural model is being designed for standardization in a sustainable manner by the appropriate committees (e.g., NVM Express™ or UEFI™ Forum).

- Questions, Feedback?

- Contacts:
  - Douglas.Farley@dell.com, lszubowi@redhat.com

Thank you!!

Backup

Multi-Pathing: One Sub-NS ESP, Two Host IP, Two Rail