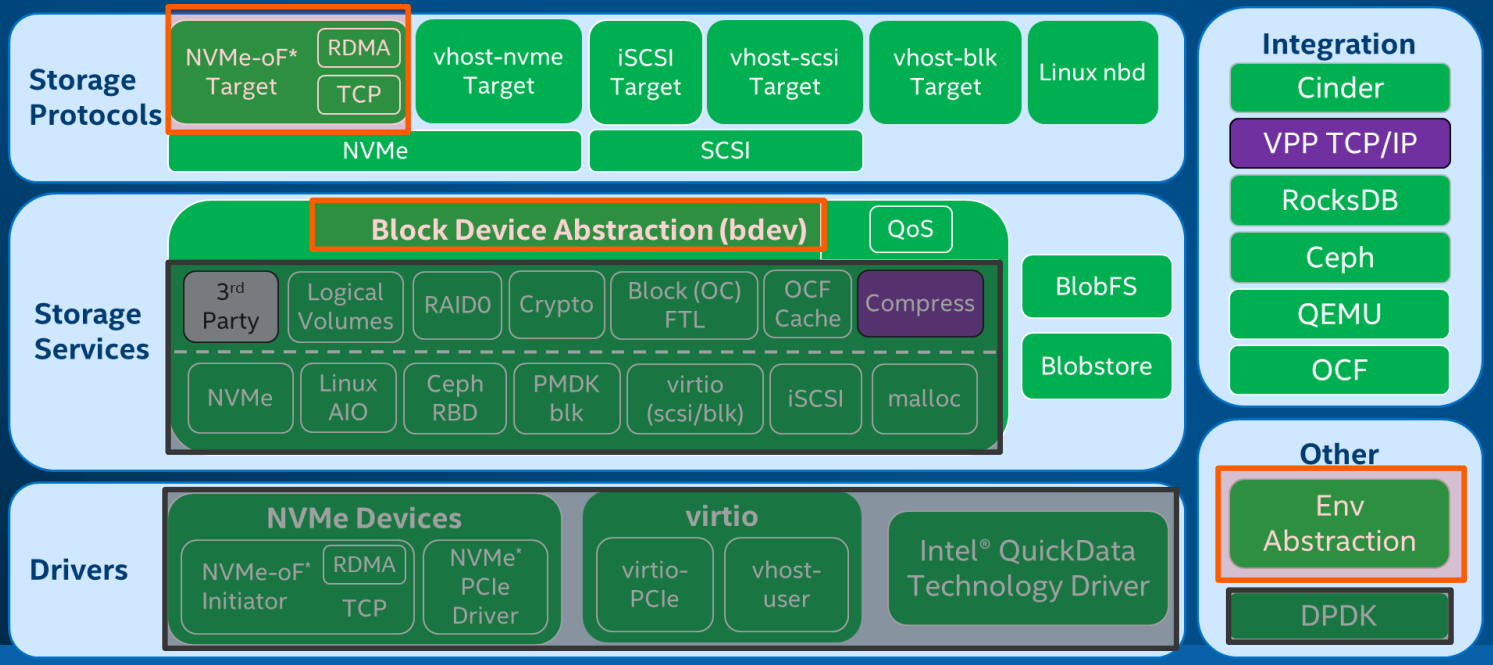
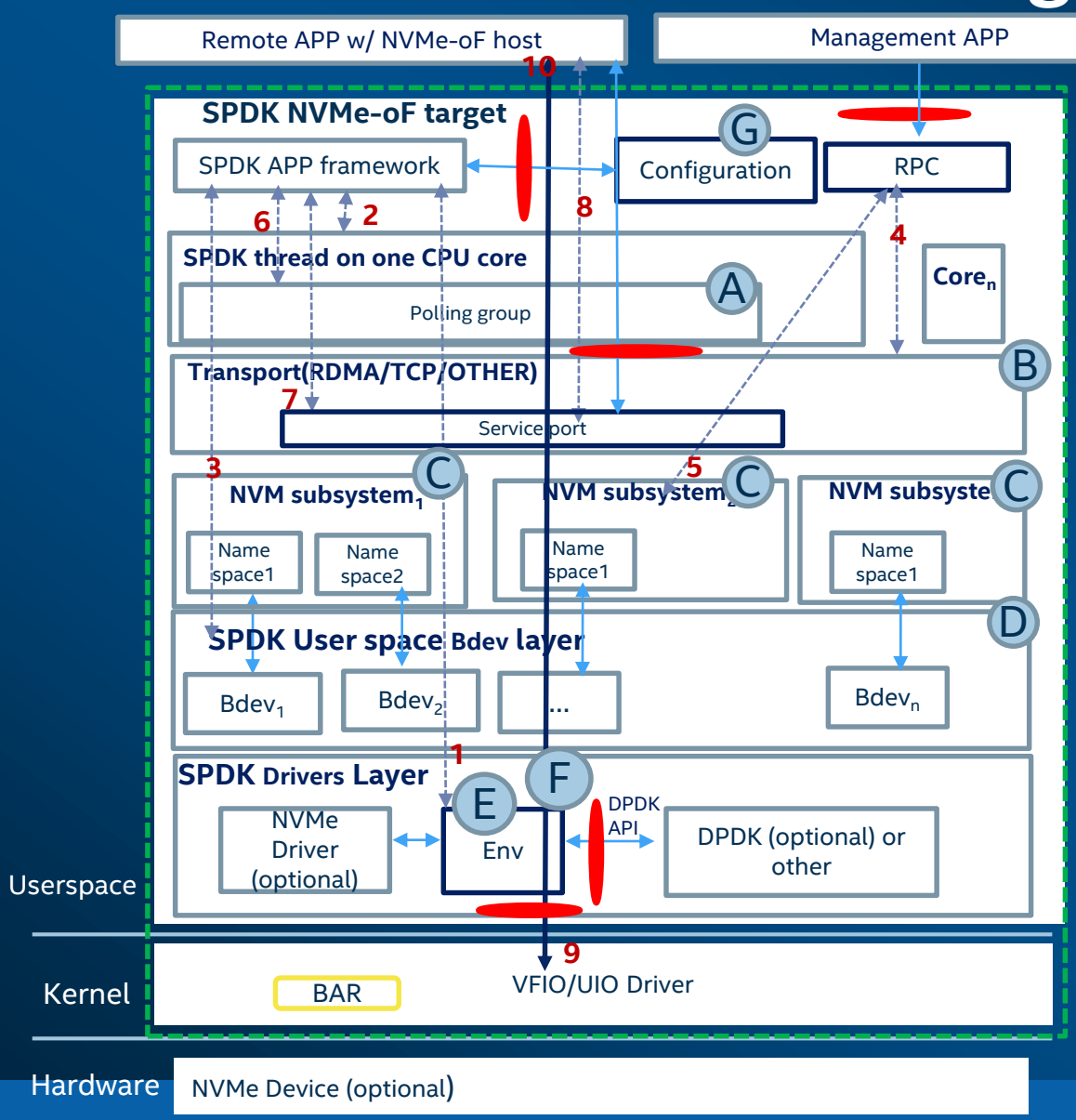
Use Case: NVMe-oF target Integration

A community member has decided to use the SPDK NVMe-oF target or conduct the further development on SPDK NVMe-oF target to inform a modified version and use it to serve their application with NVMe-oF host/initiator. It could leverage the SPDK components, i.e., such as the environment abstraction layer which can be used w/DPDK or their own equivalent, such as NVMe driver, different bdev components, NVMe-oF target and etc. These components which are highlighted below in **RED** are must have, and the components marked as black are optional.



Use Case: NVMe-oF target Integration



Assets:

A. NVMe-oF polling group
B. NVMe-oF transport
C. NVM subsystem
D. Bdev
E. Env layer
F. User Data
G. Configuration info

High Level Flow:

1) SPDK NVMe-oF target app starts and leverages the SPDK application framework to initialize env.
2) SPDK application frame work initializes the reactors in each core.
3) SPDK NVMe-oF tgt app initializes the different SPDK module subsystems (including at least bdev, net, NVMf) with the order.
4) For the NVMf subsystem initialization, it initializes g_spdk_nvmf_tgt structure with general NVMF configuration and configured transports
5) Initialize the NVM subsystems through RPC or configuration file.
6) Then the SPDK framework leverages the NVMf subsystem to initialize the poling groups on each thread for different transports.
7) The SPDK framework leverages the NVMf subsystem to listen on the different listeners configured in each NVMf subsystem.
8) When there is connection from the initiator/host side, SPDK NVMe-oF App allocates transport connection pairs for each connection according to the transport type. And dispatches the connection qpair handing to a dedicated polling group.
9) The NVMe-oF command on each connection from the initiator can be parsed and converted to NVMe command and be transferred to bdev io request, and be submitted to the owned bdev for handling.
10) When the bdev io is completed, the call back will be triggered, and the bdev io result can be converted to NVMe response, and each transport can wrap the NVMe response to its own format according to the transport type and write to the provided fabric, and the initiator can finally get the response.

Use Case: NVMe-oF target

Attack Surfaces

System Element	Compromise Type(s)	Assets exposed	Attack Method
App/Transport service Interface	Invalid NVMe-oF command on the connection or large NVMe-oF commands(DDOS)	NVMe-oF polling group, NVMe-oF transport, NVM subsystem, bdev, User data	Invalid NVMe-oF command or large NVMe-oF command sets
App/RPC Interface	Invalid RPC command or large RPC command sets (DDOS)	NVMe-oF polling group, NVMe-oF transport, NVM subsystem, bdev, User data, Configuration info	Invalid RPC command or large RPC command sets
App/configuration file Interface	Invalid configuration file.	NVMe-oF transport, NVMe-oF polling group NVM subsystem, bdev, User data, Configuration info	Invalid section in configuration file to get the user data on the backend device.
App/Env Interface	Invalid memory allocation, device enumeration	User data	Corrupt the application with bad control information
App/Driver Layer	Invalid initialization, bad SQ/CQ entries	User Data	Invalid/intercept data buffers in SQ/CQ entries

Use Case: NVMe-oF target

Threat Matrix

Assets / Surface	Global NVMe-oF target	NVMe-oF polling group	NVMe-oF transport	NVM subsystem	BDEV	ENV	User Data	Configuration info
APP/Fabric Transport (e.g., Ethernet, InfiniBand, Fibre channel) service interface	Yes	Yes	Yes	Yes	Yes	No	Yes	No
APP/RPC Interface	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes
APP/Configuration file interface	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes
APP/ENV Interface	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
App/Driver Layer	No	No	No	No	No	Yes	Yes	No