NVMe over Fabric Architecture & Functional Model

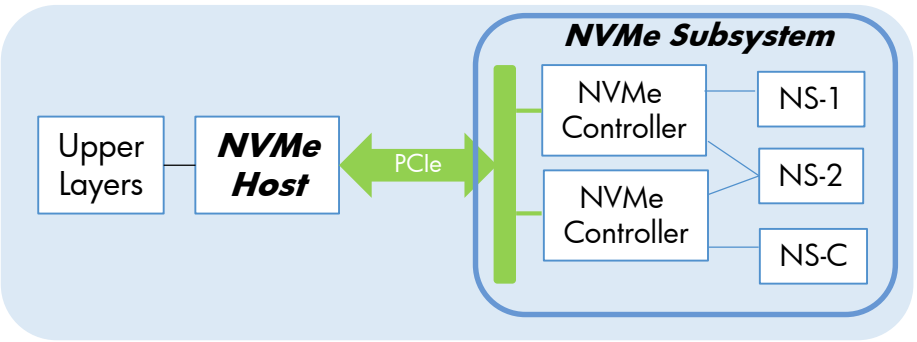
15-327v0

Siamack Ayandeh
Chief Architect
HP Networking
September, 2015
Siamack@HP.Com

NVMe over Fabric/FC -- Why

- Flash offers significant performance advantage for comparable economy vs. HDD
  - Measurements for local flash using NVMe/PCI-e report reducing latency by half vs. SCSI/SAS
- SSD having generating significant TCO reductions as local storage for servers has found its way into storage arrays
- Storage arrays may add benefits of software intelligence and dedicated hardware including:
  - Storage tiering,
  - Deduplication/real-time Compression/thin provisioning,
  - Encryption,
  - High availability of RAID and potential performance enhancements
- NVMe standard is expected to have a 10+ year life span, during which time memory developments will reduce NVM access latencies to less than 1 [us]
  - Therefore fabric latency needs to be kept in check
- In order to reduce latency for NVM workloads we need to:
  - Take advantage of higher port speed interconnects, &
  - Exploit parallelism offered by fabrics by definition

NVMe over PCI-e



- NVMe Subsystem is connected with one or more controllers via PCI-e ports – defined in NVMe Spec_1.x (figure-4)
  - PCI-e port, function, or VF are supported
  - Multi tenant extensions have been added
- NVMe Controller Register set is accessed by the ***NVMe Host***
  - A separate register-map is being defined for the capsule interface for fabrics
- For NVMe over Fabrics (NVMe), each controller offers multiple software abstractions each dedicated to a host, I call controller-abstraction (or instance or virtual-controller)
- ***In the rest of this presentation I shorten NVMe over fabrics to NVMe***

NVMe is developing an architecture model that is generic and supports multiple fabric I/O technologies

Scope of the first generation NVMe

- Continue to have a 1:1 temporal mapping of a host to a controller abstraction
  - A host can have sessions with a 2nd controller within the same subsystem or in another subsystem
- ***Discovery*** operation returns NVMe-subsystems names and addresses and not those of controllers
  - ***Hence all controller abstractions have access to the same set of namespaces (this statement needs to be fixed!)***
  - Discovery returns a single record at a time with a single address
  - Multiple records can be discovered
- There is a 1:1 mapping of NVMe Submission to Completion Queues
- Queues shall be sized to support the maximum number of outstanding commands
  - No allowance for flow control at the NVM Express layer
- The host may have multiple outstanding ***NVMe commands*** for IO
  - The host shall have a maximum of one **capsule-command** outstanding (e.g. discover, connect, etc.)

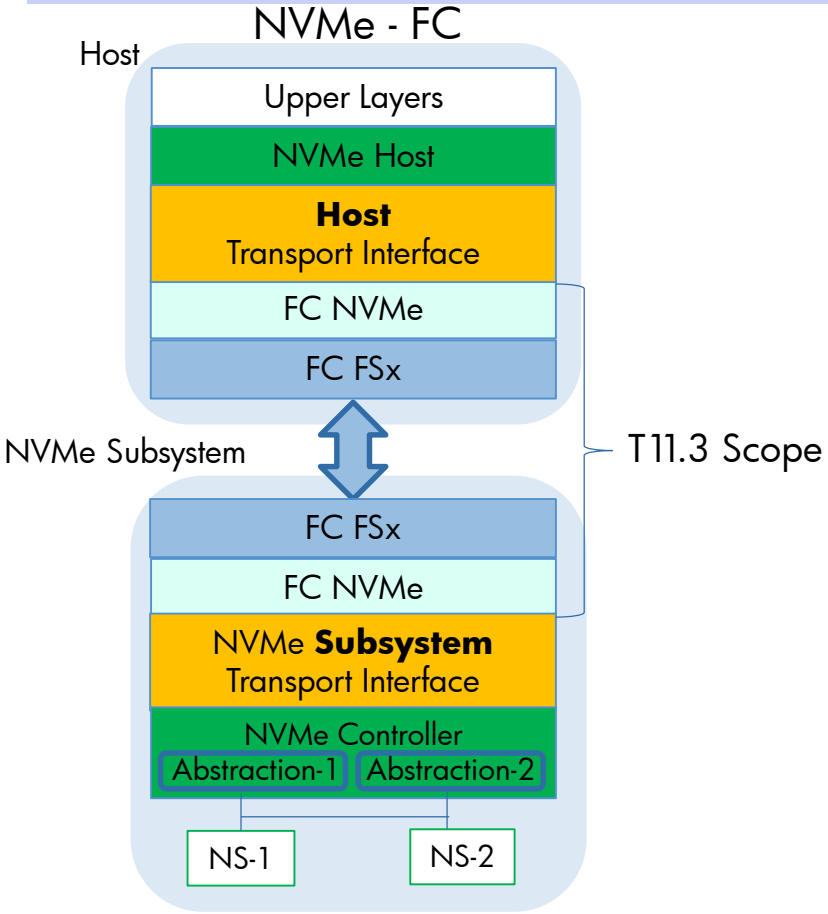
Capsule Ordering Requirement

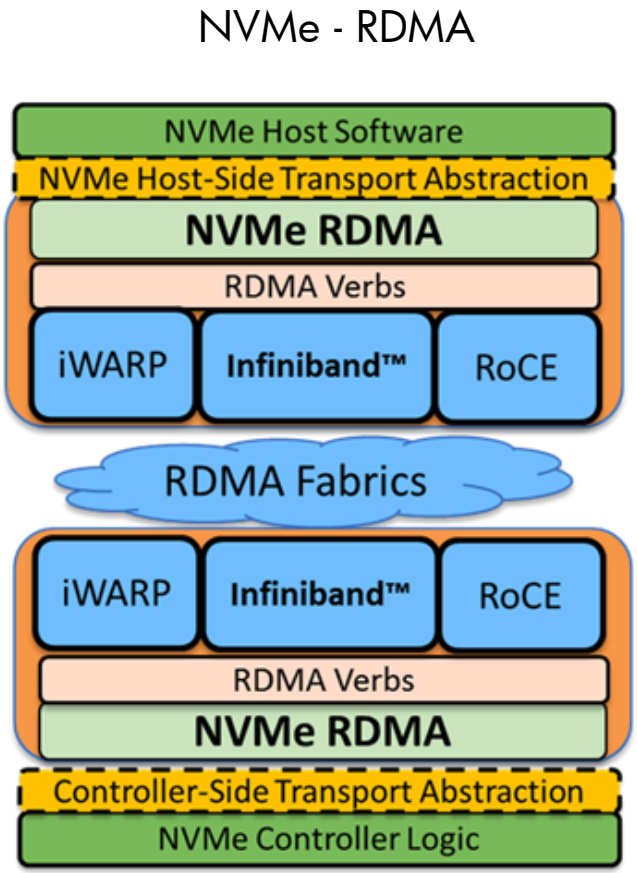
From fabric core specification:

"The transport shall provide *reliable* in-order delivery of capsules between a host and NVM subsystem over a *particular connection*. As one specific example, Submission Queue Entries (and Completion Queue Entries) shall be delivered in the order sent from the host to the NVM subsystem (or NVM subsystem to the host)."

- FC fabrics are connectionless on the wire!
- The FC-NVMe shall deliver capsules in order between the transport layer interfaces on a per host session basis (i.e. per controller abstraction)
- To enable this a capsule command sequence number is defined as part of the FC Command IU which can be used in conjunction with the Host Session ID
- The scope of term reliable should also be stated:
  - While FC offers reliable delivery the upper layer should be able to recover from loss of commands

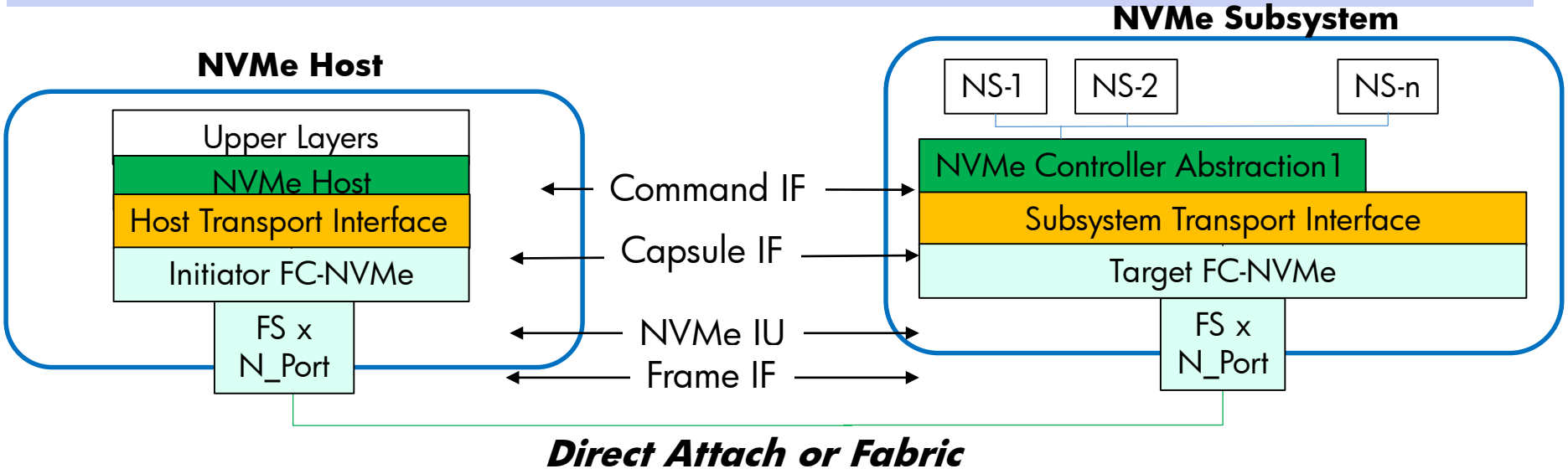
NVMe over Fabric (NVMe) – Protocol Layers and Abstractions





- Each subsystem (and host) terminates one or more fabric ports, but how?
- Remote access is via capsules and FC NVMe Information Units for FC
- Each host has a dedicated controller abstraction to access its storage resources organized as Name Spaces

NVMe– Functional Model – Host description



- NVMe Host Transport Interface description:
  - Is identified by an NVM-Qualified-Name
  - For FC this qualified name is associated with the Initiator FC node WWN, and associated port WWNs and addresses
  - Discovers the subsystem and requests access to a controller using a *Connect-capsule*, can configure a controller, and enable its operation using *Get/Set* capsules
  - May have to authenticate with a subsystem
- A distinguished host ***session-ID*** associates a host with a sub-system/controller and associated I_T_Nexus used for communications
- FC-NVMe maps capsules to NVMe Information Units (FCP4 equivalent)

Session-ID Proposal

Define the following three parameters:

- Host Session-ID (**HSID**) [2 bytes]: Identify the triple {host NQN, Subsystem NQN, Controller-ID} is set on the host (one HSID per controller)
- An Initiator_Target_Nexus-ID (**ITNID**) [x bytes]: Is a host-subsystem unique identifier for a given FC Initiator Target Nexus and is *discovered* via NVMe discovery service
  - e.g. the “Initiator port WWN + Target port WWN” can form the ITNID (please see next slide)
  - Distinguished Host Session-ID : (HSID, ITNID) is a local handle for host-subsystem-controller communications over an I_T_Nexus
- (HSID, QID) identifies a unique SQC-pair on that controller

Operations:

- ***Discover*** Capsule for FC will return all the ITNID(s) (instead of a port transport address for a subsystem)
- ***Connect*** Capsule will use the Host Session-ID as defined above
  - If a time-stamped session identifier (RFC 4122) is also required for purpose of collecting logs etc. it will be a different field
- A service request/indication or response/confirmation will provide the (HSID, ITNID) as parameters
- FC-NVMe Information Units (IU) will carry the HSID (to be added to t11.3 proposals) as well as the QID
- ITNID(s) can be added or deleted as status of ports change within the fabric –dynamics of discovery

Three Options considered for the Initiator-Target-Nexus Identifier

Option-1 (recommended)

ITNID = {host port WWN, subsystem port WWN} [16 bytes]

- Are known a priori for each device
- Remains stable while port FCIDs can change
- By assigning an alias can be associated with a readable name
- Note that this option does not change the potential loss of data in transit in case of a port down

Option-2 (not recommended)

ITNID = {host port FCID, Subsystem port FCID} [6 bytes]

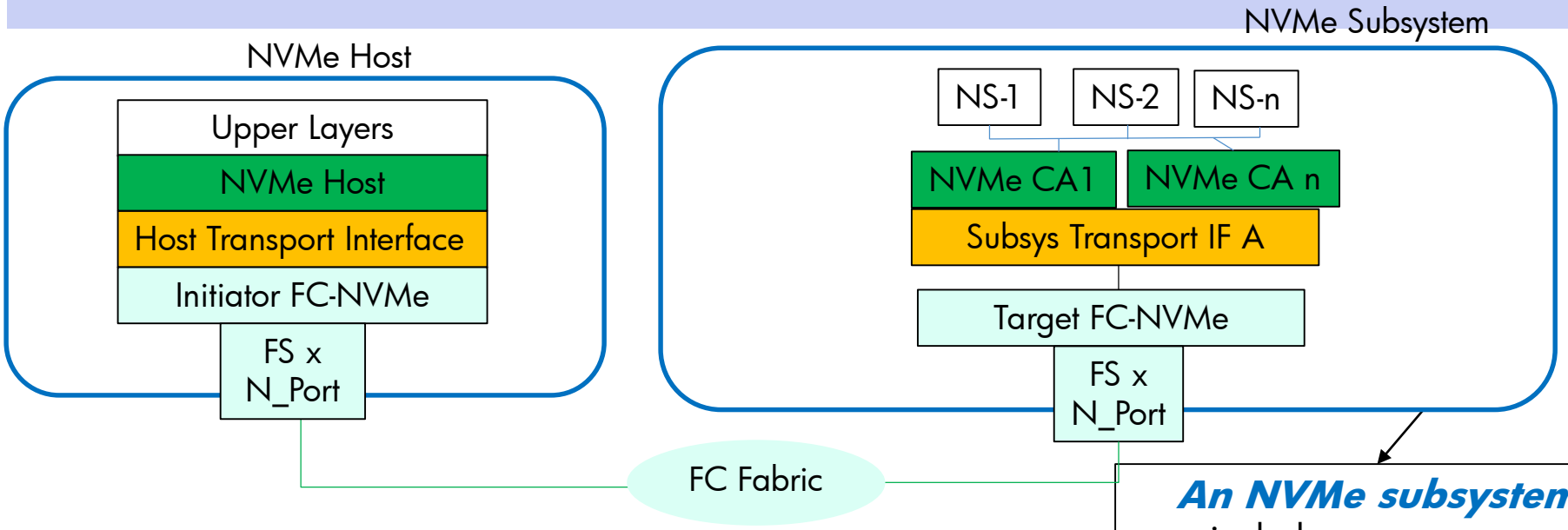
- The FC fabric assigns FCID(s), so they are not known a priori. Need to setup FC connectivity, find what addresses are assigned , and enter them into the NVMe Directory Server
- A port FCID may change after a port up/down (FLOGI) event or switch reboot and the change may have to be propagated up the layers past the FC-NVMe layer

Option-3

ITNID = {host NQN, subsystem NQN, Nexus-index} [34 bytes]

- Remains stable while port FCIDs change
- Maybe cumbersome to handle

NVMe – Subsystem description



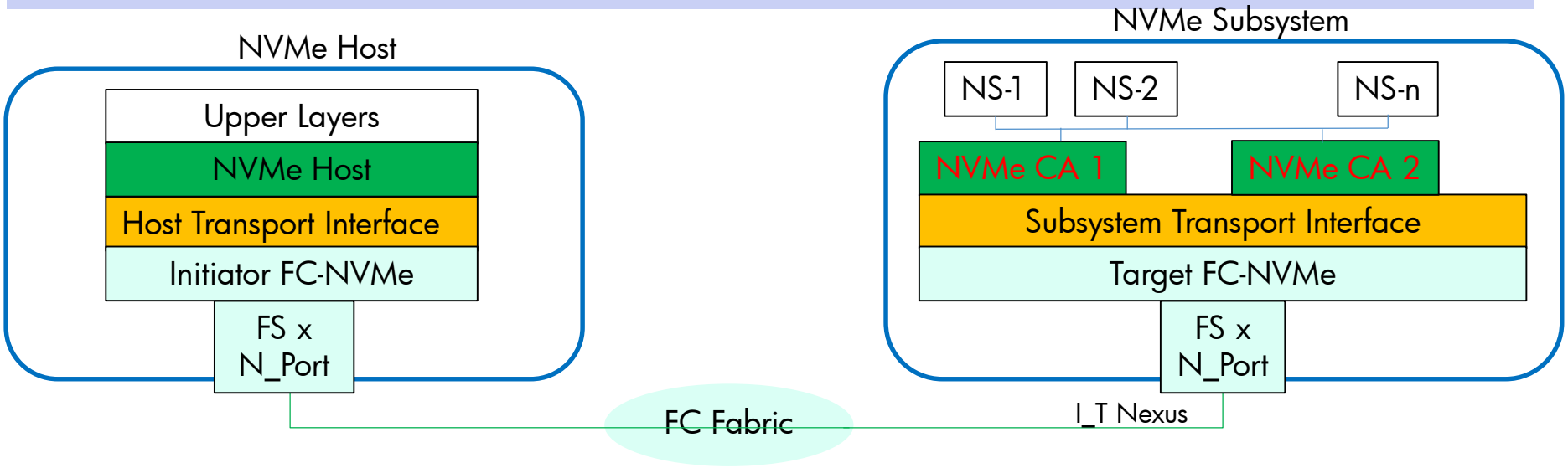
- ***NVMe subsystem Transport interface is identified by a NVM-Qualified-Name***
  - The NQN is mapped to the target FC WWN, and associated port WWNs and addresses
  - Optionally authenticates a host,
  - Responds to a "***connect***" capsule and returns a controller-ID to a given host with access to the host's name space
  - While all FC N_Ports are exposed to the NVMe subsystem Transport interface, capability to direct specific traffic to certain ports is accomplished using FC fabric zoning and discover capsule
  - ***Also need a means of communicating fabric events to the transport interface and NVMe layers***

An NVMe subsystem includes one or more controllers each offering multiple controller abstractions with one or more namespaces, one or more fabric ports, a non-volatile memory storage medium, and an interface between the controller(s) and non-volatile memory storage medium which is proprietary

NVMe – Host subsystem - Discovery

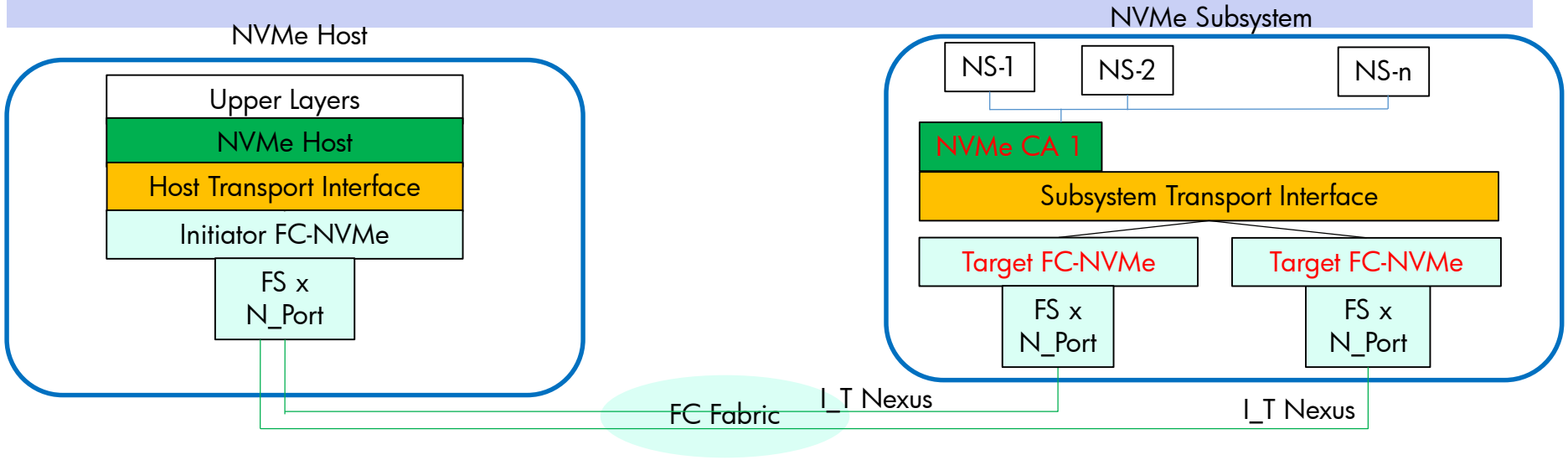
- Options for FC discovery are under discussion – along side changes made to discovery in NVMe WG
- For example one option may be for the FC-fabric name server to carry NVMe Qualified names to support NVMe Discovery
  - Or discovery can be provided by other means, perhaps beyond current FC fabric services or as a new service
- The fabric is zoned a priori to allow for NVMe host to subsystem port/process LOGIN
- Therefore FC-NVMe currently makes no use of addresses supplied by discovery as the plumbing has already happened at FC-FSx layers
- The FCIDs are also not known until this plumbing is up and running, so cannot populate the NVMe directory a priori
- Therefore it is recommended that the NVMe discovery should provide the associated subsystem qualified names and ***I_T_Nexus-ID(s)*** in ***discovery response records***
- The discovered subsystem qualified name is used to establish a session between the host and a subsystem-controller using the Connect-Capsule (using the HSID)
  - The discovered ITNID will be used as a local handle to steer the capsule along the selected I_T_Nexus
- Note that it is the FC fabric zoning policy that determines which subsystem ports are allowed for use by a given host port
- Discovery service should be dynamic enough to report any change to I_T_Nexus identifiers due to port up/down events

NVMe – Use Host Session-ID in NVMe IUs



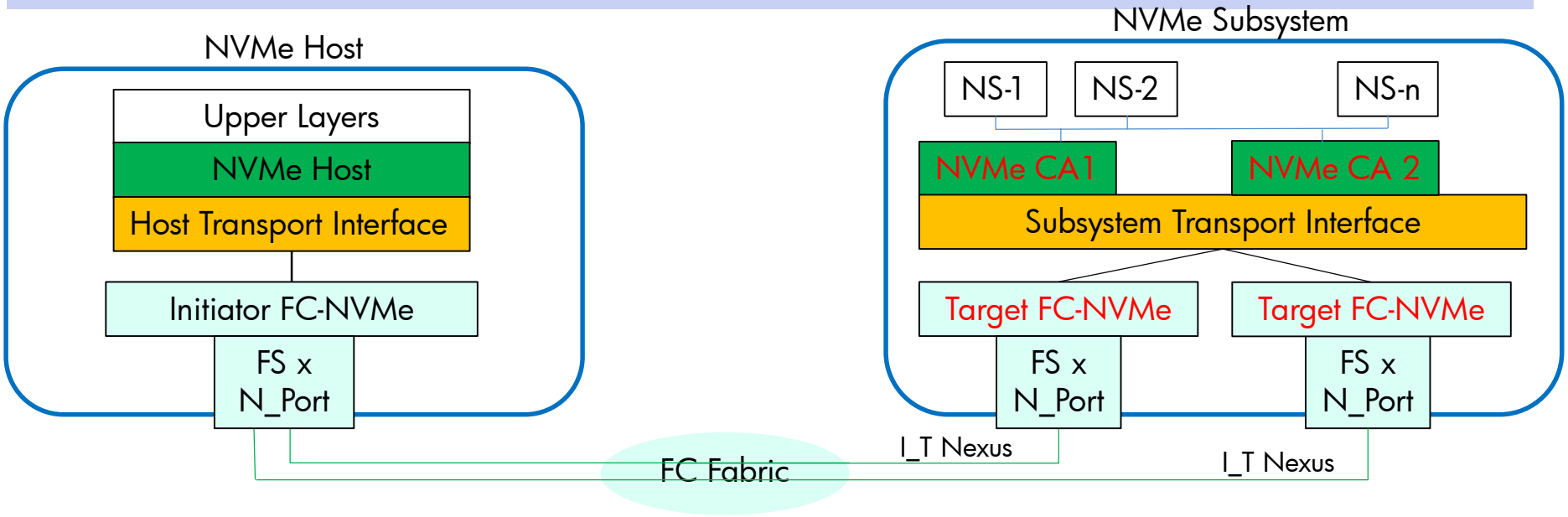
- The NVMe IU that carries the NVMe capsule currently has a queue-pair-ID field (QID)
- ***We recommend to carry the host session-ID in NVMe IU as well, here is how it is used***
- The unique identifier for the transport interface to communicate to a given queue-pair is the (Host session-ID, QID)
- The NVMe IU is sent as one or more frames each carrying a FQXID (unique per FCP command IU) – ***no concept of a connection on the wire in FC (as opposed to RDMA)***
- At the subsystem the target FC-NVMe de-capsulates the NVMe IU and passes the NVMe capsule to the Transport interface (using the ”HSID, ITNID” as the handle)
- Subsystem Fab-IF will in turn use the HSID, QID to map to a given controller/queue
- A response capsule carries the completion QID and uses the (HSID, ITNID) as handle to return via the same I_T_Nexus
- The host having reassembled the IU, uses the distinguished host session-ID to forward to the correct Completion Queue

NVMe – Dual Subsystem Ports



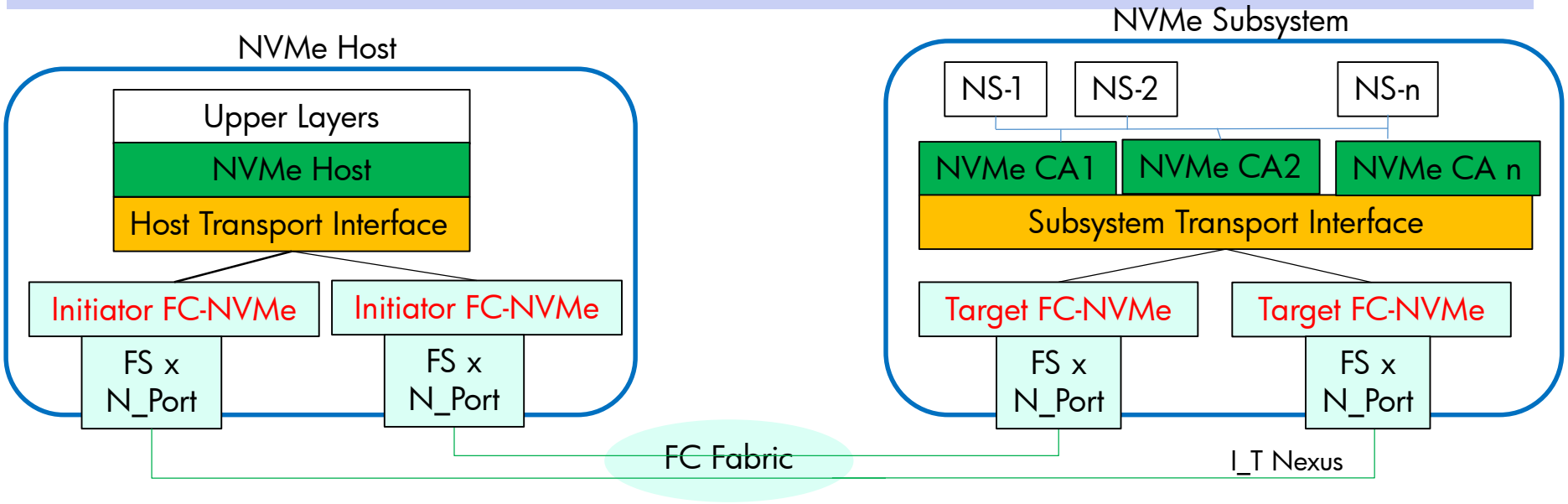
- NVMe sub-system with two ports both zoned for access by a host through port/process LOGINs; hence x2 I_T nexus are established by FC
- The provisioned subsystem/name-space is detected twice, once for each I_T nexus
  - ***And registered/presented by FC as two devices***
- Hence the host Transport interface would ***discover*** the subsystem with two I_T nexus (proposal to NVMe WG)
- ***Since FC-NVMe does not pick a path, all capsule service calls should include the distinguished HSID (HSID, ITNID)***
- Capsules for a given controller queue can be transported over all available I_T nexus:
  - ***If one subsystem port is down the host controller sessions (and admin/SCQ pairs) continue over the other I_T_Nexus***
  - ***For RDMA type transports to achieve similar capability would require sticky sessions***
- By allowing capsule & IU transport to occur over both I_T nexus FC-NVMe can offer a robust transport service

NVMe – Single Host -- Dual Subsystem Ports & Controllers



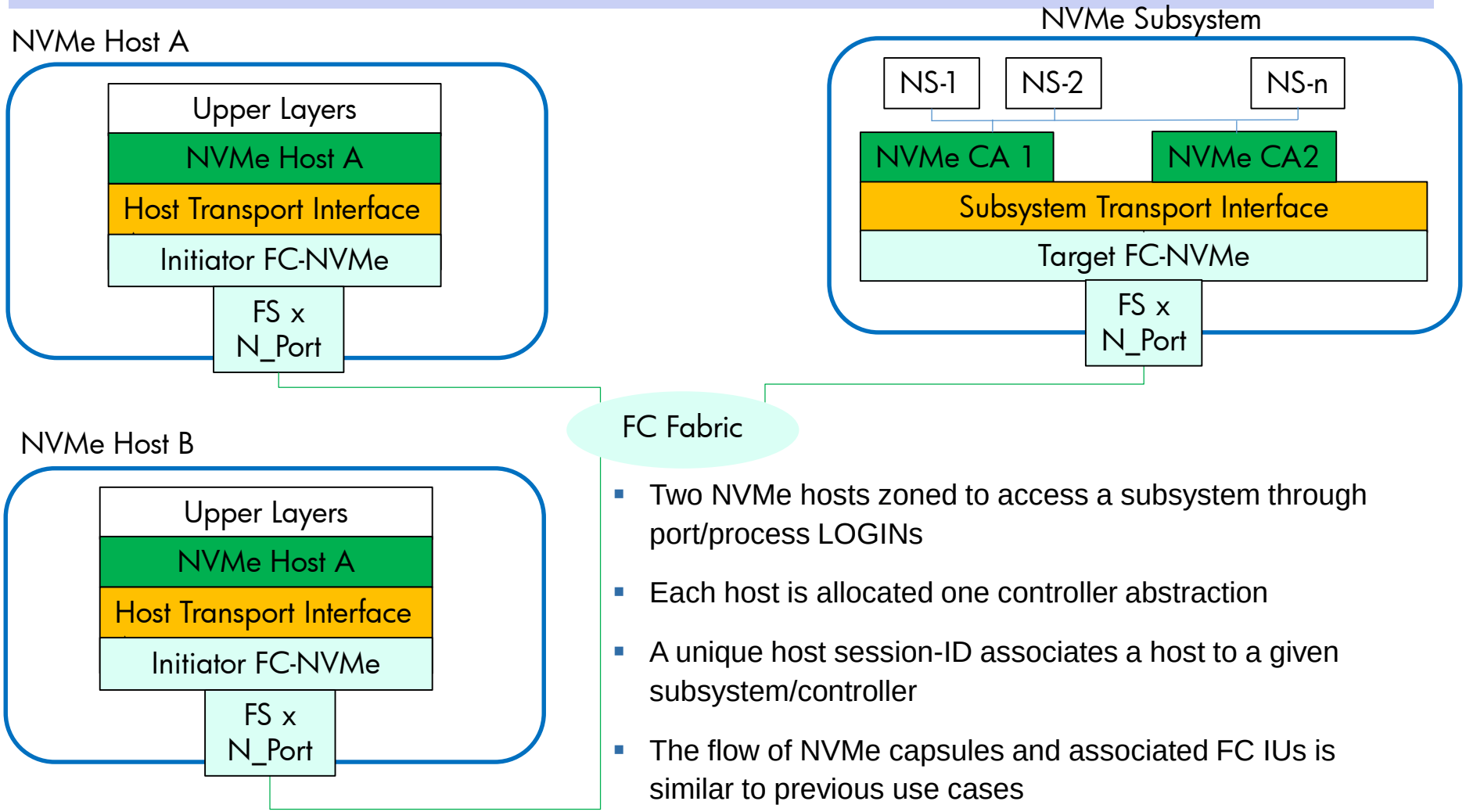
- In NVMe model/specification, a host has register access to multiple controllers (once discovered through PCIe) and subsystems
- In NVMe, the response to a connect-capsule from subsystem allocates a single controller-abstraction to a host
- Subsequent connects use the allocated controller-ID
- For a host to access a 2nd controller on the same subsystem a connect with Controller-ID=FFFFh needs to be used (as well as a new session)
- The host can communicate with both controller abstractions over both I_T Nexus using the (HSID, ITNID) as a handle to ensure responses are returned over the same path as corresponding command

NVMe – Dual Host to Dual Subsystem Ports



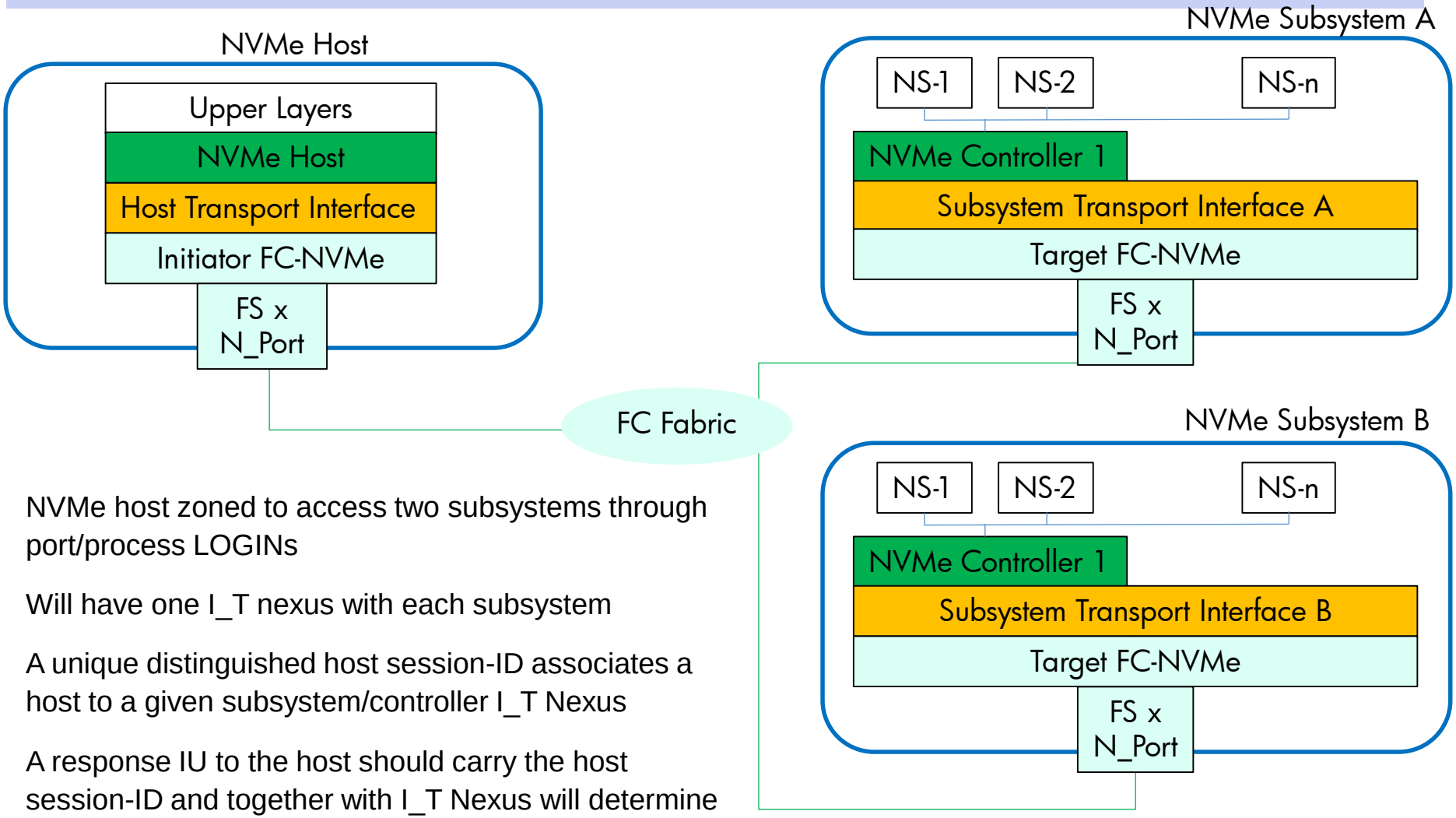
- NVMe sub-system with two ports both zoned for access by two host N_Ports through port/process LOGINs (x4 I_T Nexus e.g.)
- Each allocated NVMe controller/queue can receive exchanges/FCPIUs/Capsules from either port
- The flow of NVMe capsules and associated NVMe IUs is similar to previous slides
- ***The mechanisms supporting multipath while supported by NVMe and FC constructs are beyond the scope of this specification***
- Note that multipath is defined as having multiple physical path to a name space via the same controller Abstraction
  - Path via a secondary controller to the same name space is often used for HA

NVMe – Two Hosts Single Subsystem



- Two NVMe hosts zoned to access a subsystem through port/process LOGINs
- Each host is allocated one controller abstraction
- A unique host session-ID associates a host to a given subsystem/controller
- The flow of NVMe capsules and associated FC IUs is similar to previous use cases
- Controller would specify the distinguished HSID in various service calls to ensure delivery to the right host

NVMe – Single Host Two Subsystems



- NVMe host zoned to access two subsystems through port/process LOGINs
- Will have one I_T nexus with each subsystem
- A unique distinguished host session-ID associates a host to a given subsystem/controller I_T Nexus
- A response IU to the host should carry the host session-ID and together with I_T Nexus will determine the correct subsystem

Recommendations

- Defined a Functional Model with component descriptions, proposed for use in NVMe core fabric TP
- Add distinguished host session-ID to NVMe which is the tuple:
  - Host-Session-ID (2 bytes)
  - I_T_Nexus-ID (16 bytes)

  To be used as a local handle
- Proposed modifications to Discovery to return ITNID(s) for FC-NVMe
- Proposed modifications to Connect capsules to use the new formatted HSID to identify a session for a given host's communications with a unique controller
- Proposed modifications to NVMe IUs to carry the host session-ID, in addition to queue-ID
- Capsule in order delivery offered by FC-NVMe is on a per controller basis, not I_T nexus basis
- and clarified scope of reliable transport

Summary

- Showed the application of changes in the following use cases:

Single Host Port	Multiple Host Ports	Single Subsystem Port	Multiple Subsystem Ports	Single Controller	Multiple Controllers
√		√		√	
√			√	√	
√		√			√
√			√		√
	√		√		√
Two Hosts	Single Subsystem				
Single Host	Two Subsystems				

Thank You