

[Home](#) [Archives](#) [Profile](#) [Subscribe](#)[Main](#)

10/16/2012

VN2VN - The Good, the Bad and the Ugly

About 19 months ago I wrote a post that described two new FCoE protocols ([PT2PT](#) and [VN2VN](#)) being worked on by the T11 working group "FC-BB-6". In that post I also promised to provide two additional posts describing how these protocol work and this post is intended to fulfill the VN2VN portion of that promise. BTW, although this post was originally going to be the third in the series, I felt compelled to post about it now due to all of the attention that VN2VN has been receiving recently.

It's probably worth mentioning that I'm listed as one of the co-authors of the [proposal](#) that was eventually incorporated into FC-BB-6 as VN2VN. In this case, co-author means that Claudio DeSanti had an idea that was better than mine so I agreed to sign onto his proposal and provide constructive feedback before it was presented to the rest of the working group. During this review, I noticed a couple of minor issues and one big problem that could cause Data Corruption. Although the minor issues have been resolved, the potential Data Corruption issue remains and will probably require an extension to FIP snooping in order to resolve it. Before I get into all of that, let me describe how VN2VN works.

The VN2VN Protocol

Please note, as usual, I am not going to define every bit and potential error condition possible with VN2VN, I am merely going to show how the virtual links should normally be initialized when a host or, as in this example, a storage port is attaching to a VN2VN FCoE network.

The following diagram shows the network and attached VN2VN capable devices that will be used for the sake of this example.

[Erik Smith](#)[Follow](#)

[ARCHIVES](#)

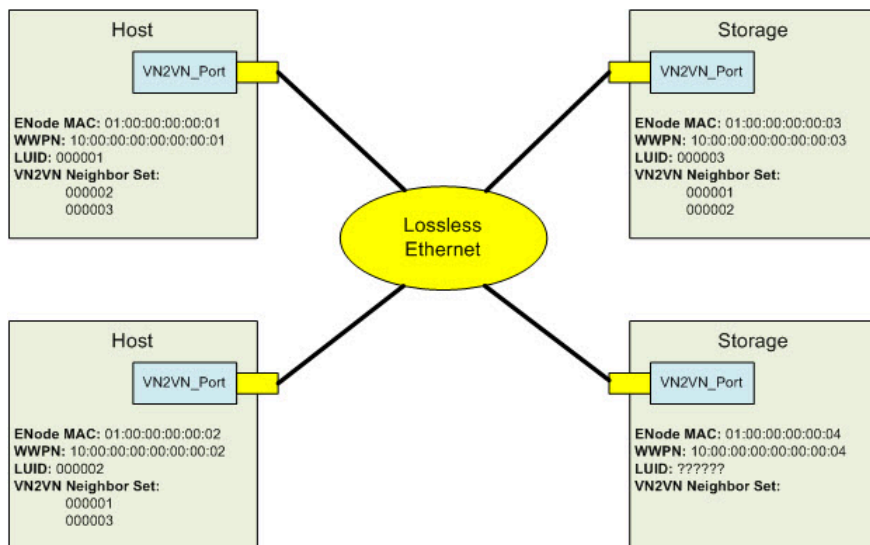
SEARCH

[CATEGORIES](#)

[Subscribe to this blog's feed](#)

DISCLAIMER

This is not an official EMC blog.
The opinions expressed here are my personal opinions.
Content published here is not read or approved in advance by EMC and does not necessarily reflect the views and opinions of EMC nor does it constitute any official communication of EMC.



The topology shown above consists of two hosts, two storage ports and a “VN2VN aware” FIP Snooping Ethernet Bridge. Each of the end devices contains a VN2VN_Port and this simply means that the end device supports the VN2VN protocol. In addition, each end device also contains:

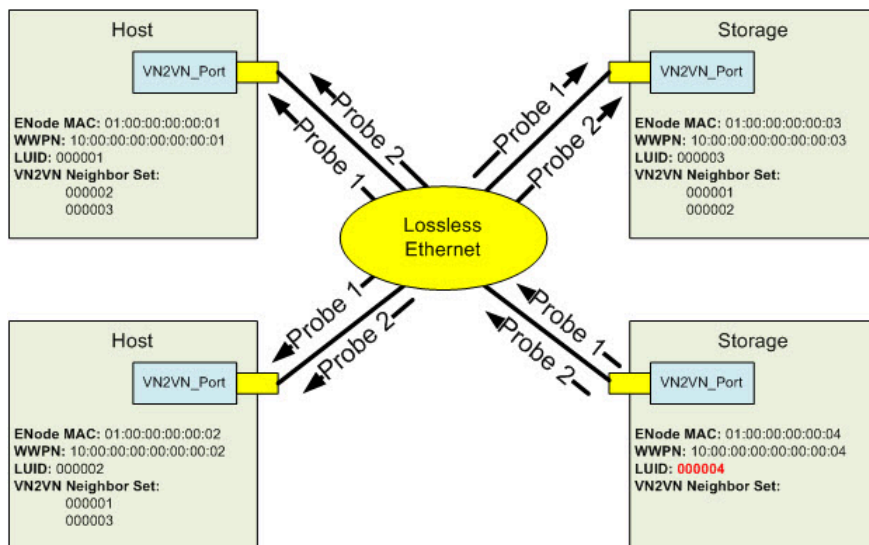
- An ENode MAC Address - This is typically the burnt in MAC Address of the physical port.
- A WWPN (World Wide Port Name) - This is the FC WWPN and could be based off of the burnt in MAC Address of the physical port. It could also be manually entered by some end user configuration utility.
- A LUID (Locally Unique N_Port ID) - This is the 24 bit FC Address that when concatenated to an FC-MAP value, will be used for FCoE frame delivery. As you’ll see, the LUIDs are randomly selected by the end devices and must be unique on the same network segment.
- A VN2VN Neighbor Set - Each end device needs to keep a list of all other VN2VN capable end devices that are on this Network.

Also note in the diagram that the Storage port in the lower right corner does not have a LUID or VN2VN Neighbor set. I’ll walk through the process of how these are selected and discovered next.

Selecting a Locally Unique N_Port ID (LUID)

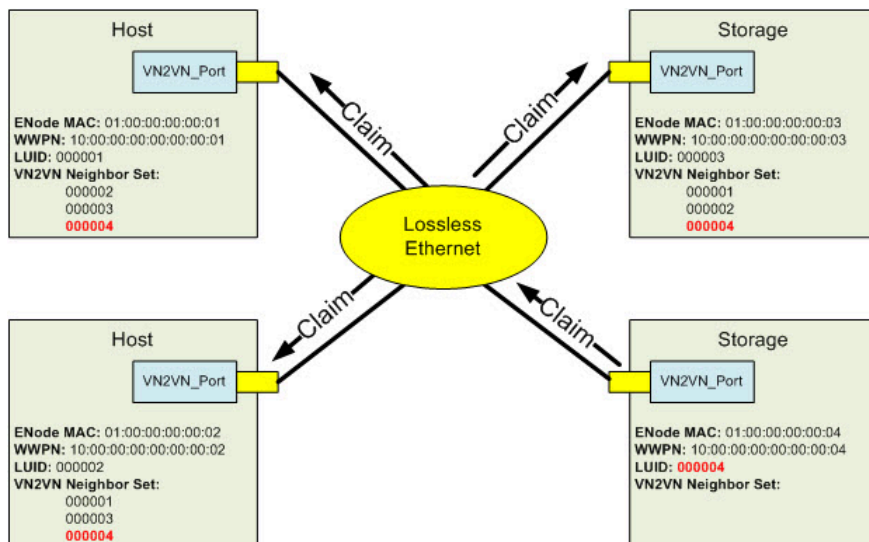
If you’re laughing and playing around with phrases such as “Hey man, got any LUIDs” or “you’re LUID behavior is troubling me”, someone beat you to those punch lines about 2 years ago! And no, that’s not the way the acronym is pronounced, it should be (Lew-eed).

After a VN2VN capable end device is powered on, it will select a candidate LUID. The process for doing this as well as sample code ([the FNV hash](#)) is included in FC-BB-6 Annex G. The point is, the end device has to select a “random” value between 000001h and 00FFFEh and then probe the network segment to determine if the LUID is already in use or not. The initializing end device performs this probe by transmitting two multicast N_Port_ID Probe Requests. BTW, two probes are transmitted to handle the case where one is dropped.



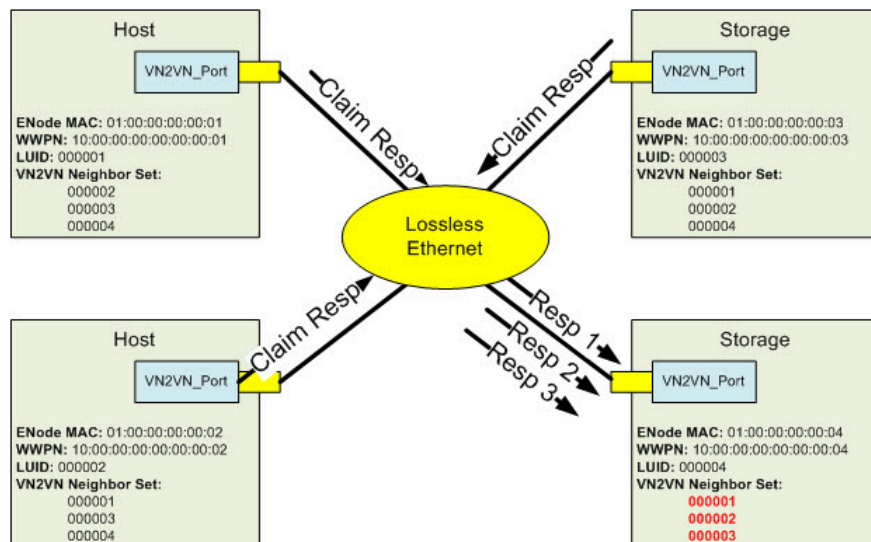
The Probe contents include the candidate LUID and the end devices WWPN. When other VN2VN capable devices on the network segment receive the Probe, they will notify the end device that originated the Probe if they are already using that LUID. If the end device receives a notification that the candidate LUID is already in use, it will select another value and transmit another two probes with a new candidate LUID. If no responses are received after ~400ms, the end device that transmitted the Probe will Claim the LUID by transmitting a mulicast N_Port_ID Claim Notification.

N_Port_ID Claim Notification



The information contained in the Claim Notification includes the maximum size FCoE frame supported by the end device that transmitted the Claim as well as an FC-4 descriptor. When the other end devices on the same network segment receive the Claim Notification, they will record the new LUID in their Neighbor set as well as the contents of the FC-4 descriptor. The FC-4 descriptor is 160 bytes long and this in addition to the LUID, N_Port_Name and Max_FCoE_Size that will be recorded will require each end device to store at least 192 bytes of information for every end device that is in the network. Keep this in mind; I'll get back to it in a little bit. The response from the other end devices on the network segment to the Claim Notification is a unicast N_Port_ID Claim Response.

N_Port_ID Claim Response



The contents of the unicast N_Port_ID Claim Response includes a Name_Identifier descriptor, a Vx_Port Identification descriptor, an FC-4 Attributes descriptor and then the frame is padded to be the maximum length FCoE Frame supported by all other devices on the network segment. When the device, Storage in this case, receives the Claim Responses, it will add information from each response to its VN2VN Neighbor Set. Again, keep this in mind, since each Response is padded to the Max_FCoE_Size, for each end device in the network segment, at least 2k of bandwidth will be consumed each time an end device initializes.

After the Claim Responses have been received, the end devices may instantiate Virtual Links with each other by using FIP FLOGI and following the rules for point-to-point operation defined in FC-LS-2. Also, when ready to instantiate virtual links, the initializing end device, storage in the case, shall also start to transmit N_Port_ID Beacons which will be used by the other end devices in the network for keep-alive purposes. There are two important things to point out here:

1. As of today, every Host must login to every Target Port on the same network segment in order to perform SCSI-FCP discovery and determine if the Host has been granted access to any LUNs on that Target Port or not.
2. Every VN2VN device participating in the network will effectively need to keep a local copy of a "Name Server" (A.K.A., the VN2VN Neighbor Set).

Now that I've given you a rough idea how the protocol functions, I'll get into the Good, the Bad and the Ugly points of VN2VN.

VN2VN - The Good...

1. It's very simple to administer! Since there is no FCF, there is no FC zoning. As a result, every Host will Login to every storage port and only the LUN Mask on the storage will need to be updated to allow each Host access. Oh, yeah, don't even THINK about using VN2VN if your array doesn't support LUN Masking. You'd be asking for trouble.
2. It's cheap! Since there is no FCF requirement with VN2VN, you can use it with any DCB capable switch that supports FIP Snooping and is

“VN2VN aware”. In the past I’ve described [what FIP Snooping is](#) and [why it’s needed](#). The “VN2VN aware” requirement is new and the reason it’s needed is described in “the Ugly” section below.

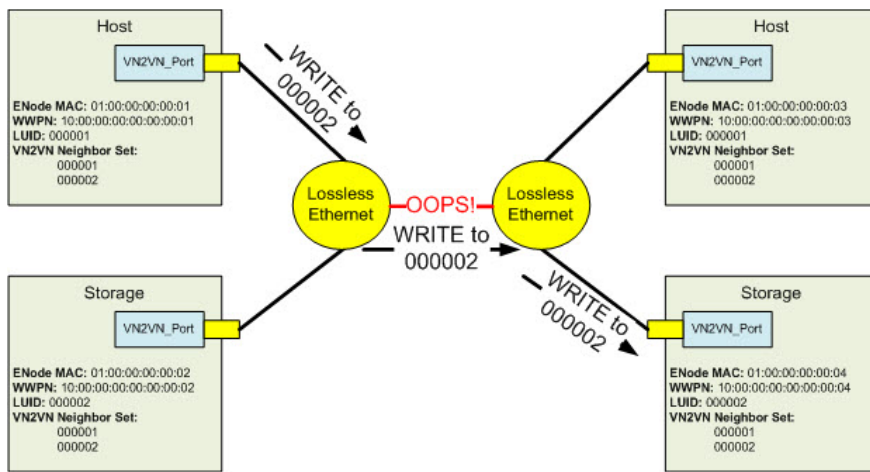
3. It should perform very well! Because frames do not have to be handled by an FCF, you can bypass a potential bottleneck in the network.

VN2VN - The Bad...

1. There is no zoning! Every Host will login to every Target and this will limit the scalability of the entire VN2VN network to be less than or equal to the number of logins that can be handled by any one array port. For example, if you have an array port that can only handle 64 logins, you will currently be unable to have more than 64 end devices in the entire network even if other array ports can handle 2k logins. Also, based on some testing that I’ve done in the lab for the FC-SCM working group, the amount of time that it will take each Host to complete SCSI-FCP discovery on each Target could exceed 500ms. This could be a problem if you imagine a VN2VN network with 120 target ports in it. In this case, if each Host were to require 500ms to perform SCSI-FCP discovery on each Target, the Hosts would need at least 60 seconds to complete discovery during boot or even worse, after a cable pull and push. Of course you could limit the scope of discovery by using multiple VLANs, but this sort of kills the ease of use story for VN2VN. In fairness, I have heard of an effort to utilize a [TDZ](#) like function to limit the scope of discovery, but the details have yet to be worked out.
2. There is no FCF! Because there is no FCF, I’m concerned that the network will not be FC aware. The problem with this is that FC switches commonly provide protection against end devices that fault and do things like continuously FLOGI, PLOGI/query the Name Server and perform SCSI-FCP discovery. In these cases, how will a VN2VN network react? Imagine an end device repeatedly initializing and then every other end device in the network sends 2k unicast N_Port_ID Claim Responses.. I haven’t observed this failure scenario in the lab but based on past experience, it seems like it would be very bad.
3. There is no Name Server! Every end device needs to keep a local copy of a “Name Server” (A.K.A., the VN2VN Neighbor Set). This means if there are N devices in the VN2VN network, each end device will need to allocate at least $(N-1)*200$ bytes. This might not seem like much but some of the implementers I’ve spoken with seem to think it could be problematic in larger configurations.
4. Every end device needs to Beacon and track the state (via Beacon responses) of all other end devices in the network. This just screams scalability problem to me.

VN2VN - The Ugly...

1. The problem I’ve already referred to a couple of times has to do with the potential for Data Corruption to occur when two VN2VN networks are joined. Consider the following topology.



As I described in the earlier [FIP Snooping post](#), when a link is connected between two switches and duplicate MAC Addresses exist, it is possible for frames to be routed to the wrong destination. In this case, you need to realize that the LUID accounts for the least significant 3 bytes of the MAC Address used by FCoE. The 3 most significant bytes are set to a constant value of 0EFC00h by default. As a result, if you have duplicate LUIDs, you'll probably have duplicate MAC Addresses. This condition could exist for several seconds until detected by the end devices (via Beacon, etc) and during this time, if the host on the left sends a WRITE it could be misrouted to the Storage on the right. If the CDB of the WRITE describes a valid portion of data visible to the original host with the LUID of 000001h (top right of diagram), then valid data will be written to the incorrect Storage port and Data Corruption will result.

The solution for the above problem almost certainly requires a new set of Dynamic ACL rules to be defined and vetted by the FC-BB-6 working group, but this work hasn't been done yet.

Conclusion

With all of these issues, you may be wondering why the VN2VN protocol was put into the emerging FC-BB-6 standard. The answer is, it was designed to solve a very specific problem; connecting a small number of FCoE Ports together without the need for an FCF. In this specific use case, I'm comfortable with using VN2VN in a test environment at present and would be probably be comfortable supporting a small (< 255 end devices) production environment once I could verify that the "Ugly" problem has been resolved and was able to determine how well the protocol actually scales.

Although some effort is being put into making VN2VN work in larger configurations, once all of these efforts have been completed, I am seriously wondering how much simpler it will be to configure than a traditional FCF based FCoE configuration.

With all of this in mind, I think if you need a direct connection to a block storage array right now, you should seriously consider using iSCSI instead of VN2VN.

Thanks for reading!

Erik

Posted at 09:29 AM | [Permalink](#)

Reblog (0)

Comments

 You can follow this conversation by subscribing to the [comment feed](#) for this post.



Do I get in trouble for saying Erik is as always making some good points ? Anyway there is a lot in here and I am ascertain I will make more than one comment/response. Here I want to comment on a few high level topics.

1) I of course agree and indeed further extend what Erik says in terms of choice. I am a firm believer that whilst there will be a long tail of physical FC that over time Ethernet will dominate in the DC and that users have choices with FCoE (BB5 and BB6) iSCSI NFS pNFS and SMB. Indeed virtualization and orchestration makes it easier than ever for the DC to change protocol or use multiple protocols for different purposes. We see this mobility and multiplicity as increasingly common in our deployments.

2) I also completely agree that as with most things in this space implementation will take time, starting on low end devices and niche use cases and moving up line over time, and starting on switches and servers and moving to storage over time. This is good and of course allows people to gain confidence and for wrinkles to be ironed out.

3) convergence has many unintended consequences. One is that we have to think about things differently, we may need solve problems in different ways as a result of convergence. This does not mean the old way was wrong nor does it mean the new way is complex or that we should not change. Another is that we may need to learn or relearn things we know or think we know about our own domain let alone learn more about other domains. I remember that with FCIP iFCP iSCSI and wan acceleration we found that many Ethernet and ip experts did not understand things they needed to understand about their own protocols.

No matter FCoE with BB5, FCoE with FDF with BB6, FCoE with VN2VN with BB6, iSCSI, NFS, pNFS, or SMB, network convergence is a journey where we need to learn. Also there may be both problems in implementation of existing protocols and maybe in new protocols that will need to be fixed over time.

However, Ethernet works, if it did not we would get many cases of corruption in the Datacenter. Ethernet is used for lan backup, iSCSI, NAS, clusters, connectivity between the layers and components in a complex multiserver database deployment. As such I struggle to believe that some of the problems are as bad as they seem and believe that even if they exist the use of good network products and robust best practices will ensure that life in Ethernet world is as safe and as scalable as in the fc world. I'm sure erik would agree that fc is not perfect and works in part through good best practice.

I recently had to correct myself, I sometimes say if you have infinite bandwidth and zero latency in the network you don't need cos/qos. Over the last few months we actually found this was not true and

that you still need a very good network to avoid the new problems we are finding with modern DataCentres. Erik's concerns are real, but some may be misplaced, others may be readily solvable with the rut deployment model, others may as he notes need protocol or product enhancements.

Posted by: [Simon gordon](#) | [10/16/2012 at 12:49 PM](#)



Hey Simon, no complaints from me! Thanks for the mention...

Just curious, which concerns are real and which may be misplaced?

Posted by: [Erik](#) | [10/16/2012 at 01:09 PM](#)



Hi Erik, Simon,

All good points and, when looking at the intention of VN2VN, the development seems to be mainly focused on cost-control and easy-of-deployment. Although I'm all for this noble goal I do think we're being entangled in this ever so encompassing triangle of cost-control vs. availability vs. performance. You can't have them all three and leaning towards cost-control will always be at the expense of another.

Especially when looking at very simple architectures like the ones you've depicted above, you must agree that "playing" with customer data like this needs some very significant boost in RAS development to prevent such issues to occur, ever.

I still see some significant issues with FCoE in general and not only from a technical standpoint. Then again, Rome wasn't build in a day so we're likely be in business for a long time. :-)

Regards,
Erwin

Posted by: [Erwin van Londen](#) | [11/19/2012 at 12:19 AM](#)



Simon,

One more comment. You mention that Ethernet is ubiquitous in the datacentre and it "just works" however you also know that in order to bring reliability in such a lossy protocol it needs a stability factor which has been bolted on in the form of TCP/IP. It's not the Ethernet side that provides all the examples you mentioned (iSCSI, NAS, NFS Clusters etc.) It is TCP/IP that has made all this possible. Around 95% of the worlds wire does not run Ethernet (Framerelay and other "telco" protocols are very much alive around the world connecting continents, countries and cities. No Ethernet here.) but they all do run TCP/IP.

If the development of FC had two additional efforts whereby multicast and broadcast was enhanced to allow for greater scalability plus the fact that all FC vendors had brought down the cost of FC to the level of Ethernet then FC would have had all the option to run a total consolidation of all protocols in the datacentre. As you know the FC4 mapping is the most flexible and easiest to

adopt upper layer protocol like it already has done for SCSI, IP, HIPPI, IPI, SBCCS, ATM etc etc...

Anyway, just my \$0.02

Kind regards,
Erwin

Posted by: [Erwin van Londen](#) | [11/19/2012 at 12:37 AM](#)



Hi Eric

Very good post and discussion.

Based on the discussion and last post from Erwin, I started to think that behaviors in FCoE case and FC case are not fundamentally different.

1. Both forward bad packets when in cut-through mode.
2. Both handle error condition either at store-forward switch on the path or at the end node (receiver) - bad packet will be discarded (assuming CRC catches the error).

Do you agree? Or am I missing something?

(There is difference about handling of unresolved broadcast in case of Ethernet - but I am assuming same behavior for #1 and #2 will take place on all the unresolved broadcast paths).

- Manoj

Posted by: [Manoj](#) | [01/16/2013 at 07:13 PM](#)

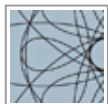


Hi Manoj, I agree with the similarities you explicitly point out, but there are a couple of important differences between the FC and the FCoE case.

With FC there is no concept of a unicast flood. As a result, corrupted FC frames will not be forwarded to every single N_Port that exists on some default VSAN. Also with FC, there are zoning mechanisms in place that could prevent forwarding data to unintended recipients.

With FCoE, in the unlikely event that the wrong bit gets flipped, you could end up with SCSI Data being unicast flooded to every Ethernet end station that is sitting on the default VLAN. The problem is that this data could be visible to anyone via something like TCPdump or wireshark.

Posted by: [Erik](#) | [01/17/2013 at 08:28 AM](#)



Erik:

Outstanding post, as usual. Informative, lucid and rational. No technical bigotry or emotional gobbledygook. Just the facts. :-)

Quick question (maybe not so quick, sorry):

In an architecture that includes appliances that only support BB5, is it possible to connect an FCoE initiator (server CNA) and an FCoE target (storage array to an FCF in NPV/Access gateway mode, that is NOT connected to a FC switch for FLOGI and PLOGI services, and actually have this work? In other words, picture a Dell M8428-k FCoE blade switch (FCF) in Access Gateway mode that is NOT connected to an FC switch. Or perhaps a UCS Fabric Interconnect with an FCoE target plugged directly into it while in NPV mode and NOT connected to an FC switch.

My thought is that, with an FCF in NPV/Access Gateway mode that is also NOT connected to an FC appliance that can provide FIP FLOGI and PLOGI services, I am not sure how VN_Port to VN_Port communication can take place. The FIP FLOGI and PLOGI semantics do NOT go away. In other words, upon initialization, a VN_Port must discover the FCF (maybe the FCoE VLAN, too), log into the fabric, receive an FC-ID and FPMA, and then register with the name server and perform peer discovery as part of the PLOGI process. Without an FCF in full fabric mode or an FCF in NPV/Access gateway mode that is not connected to an FC switch, how the devil can you get this to work?!?!?

Seems logical that you would need to have the FCF provide full fabric services OR have it connected to an FC switch. HOWEVER, I have been told by some UCS experts, for example, that one CAN connect an FCoE array directly to a UCS Fabric Interconnect in End-host/NPV mode and actually be able to provision storage. Again, the FI is supposedly NOT connected to an upstream FC switch.

I have also been told that the Dell (Brocade, really) M8428-k in AG mode, for example, would also support VN_Port to VN_Port communication, even if it, too, is not connected to an FC switch!

What the devil is going on?? :-)

Victor Lama

Posted by: Victor Lama | [03/13/2013 at 11:49 PM](#)



Hi Victor, comments inline and marked with ES -:

Erik: Outstanding post, as usual. Informative, lucid and rational. No technical bigotry or emotional gobbledygook. Just the facts. :-)

ES - Thanks! This is exactly the type of information that I am trying to provide...

Quick question (maybe not so quick, sorry): In an architecture that includes appliances that only support BB5, is it possible to connect an FCoE initiator (server CNA) and an FCoE target (storage array to an FCF in NPV/Access gateway mode, that is NOT connected to a FC switch for FLOGI and PLOGI services, and actually have this work? In other words, picture a Dell M8428-k FCoE blade switch (FCF) in Access Gateway mode that is NOT connected to an FC switch. Or perhaps a UCS Fabric Interconnect with an FCoE target plugged directly into it while in NPV mode and NOT connected to an FC switch. My thought is that, with an FCF in NPV/Access Gateway

mode that is also NOT connected to an FC appliance that can provide FIP FLOGI and PLOGI services,

ES - The answer is no. With FC-BB-5 ENodes, an FCF must be present and it either needs to handle the FIP FLOGI itself (such as when the FCF is running in FC-SW mode) or it needs to utilize the services of another device to service the FIP FLOGIs (such as when the FCF is running in NPV mode).

I am not sure how VN_Port to VN_Port communication can take place. The FIP FLOGI and PLOGI semantics do NOT go away. In other words, upon initialization, a VN_Port must discover the FCF (maybe the FCoE VLAN, too), log into the fabric, receive an FC-ID and FPMA, and then register with the name server and perform peer discovery as part of the PLOGI process. Without an FCF in full fabric mode or an FCF in NPV/Access gateway mode that is not connected to an FC switch, how the devil can you get this to work?!?!

ES - The short answer is this won't work until VN2VN is supported by both the host and the storage.

Seems logical that you would need to have the FCF provide full fabric services OR have it connected to an FC switch. HOWEVER, I have been told by some UCS experts, for example, that one CAN connect an FCoE array directly to a UCS Fabric Interconnect in End-host/NPV mode and actually be able to provision storage. Again, the FI is supposedly NOT connected to an upstream FC switch. I have also been told that the Dell (Brocade, really) M8428-k in AG mode, for example, would also support VN_Port to VN_Port communication, even if it, too, is not connected to an FC switch! What the devil is going on?? :-) Victor Lama

ES - In regards to the M8428-k, if it's running in AG mode, then you are absolutely correct. You need to have a core switch to provide the FC services. In regards to the UCS, there are two different End-host modes, one for Ethernet and one for FC. I'm pretty sure it's possible to run Ethernet in end-host mode while running FC in FC-SW mode. Perhaps this could explain the confusion?

Posted by: [Erik](#) | [03/29/2013 at 08:37 AM](#)

The comments to this entry are closed.

[Brass Tacks](#) [Powered by TypePad](#)