

*Claudio DeSanti (Cisco), Landon Curt Noll (Cisco)
 Bob Nixon (Emulex), Fred Knight (NetApp), Erik Smith (EMC),
 Roger Hathorn (IBM), Lou Ricci (IBM), Craig Carlson (QLogic),
 John Hufferd (Hufferd Enterprises), Pat Thaler (Broadcom)*

1 Locally Unique N_Port_IDs

1.1 Overview

Figure 1 shows the VN_Port to VN_Port reference model. ENode MACs supporting VN_Port to VN_Port Virtual Links (i.e., VN2VN ENode MACs) shall have a VN_Port dedicated to the instantiation of VN_Port to VN_Port Virtual Links. This VN_Port is called VN2VN_Port in this document. FCoE frames originated by a VN2VN_Port are transported over the Lossless Ethernet network to the VN2VN_Ports with which the originating VN2VN_Port has established a Virtual Link.

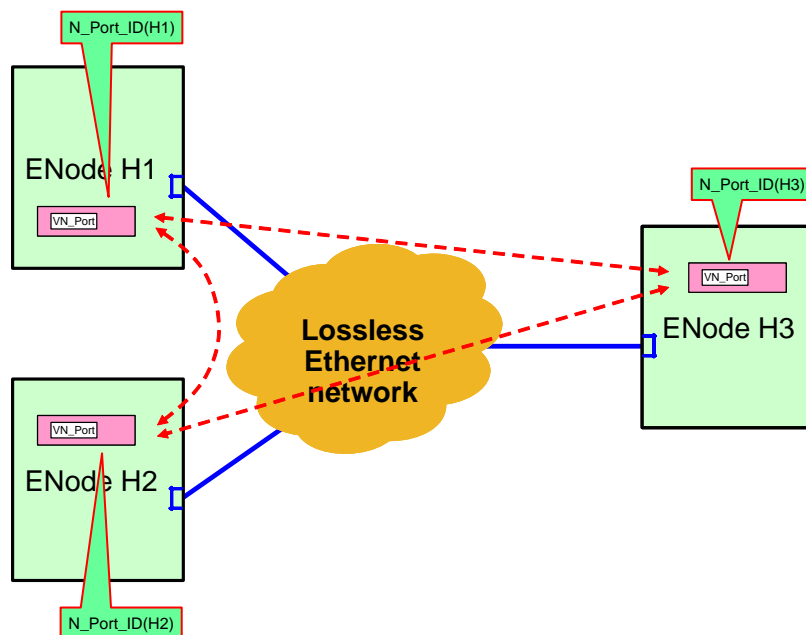


Figure 1 – VN_Port to VN_Port Reference Model

As shown in figure 1, VN2VN ENode MACs shall assign N_Port_IDs to themselves, because in the VN_Port to VN_Port reference model there is no FCF that performs N_Port_ID assignment.

N_Port_IDs used for VN_Port to VN_Port Virtual Links shall be unique over the Lossless Ethernet network to which VN2VN ENode MACs are connected and are suitable to be used only for communications over that Lossless Ethernet network. These N_Port_IDs are called Locally Unique N_Port_IDs.

The protocol defined in this document enables VN2VN ENode MACs to select Locally Unique N_Port_IDs (see 1.4.2), to discover the VN2VN_Ports reachable over the Lossless Ethernet network to which they are connected, including their FC-4 support (see 1.4.2), and to establish VN_Port to VN_Port Virtual Links with the discovered VN2VN_Ports (see 1.5).

The FPMA used as VN_Port MAC address for a VN2VN_Port is determined by concatenating its Locally Unique N_Port_ID to the constant VN2VN-FC-MAP. The constant VN2VN-FC-MAP has the value 0EFD00h. This enables easy recognition of the MAC addresses used by FCoE for VN_Port to VN_Port communications, because they all share the same VN2VN-FC-MAP prefix. The value VN2VN-FC-MAP shall not be used as a Fabric FC-MAP.

Figure 2 shows an example of a configuration in which two VN2VN ENode MACs instantiate a VN_Port to VN_Port Virtual Link in presence of an FCF.

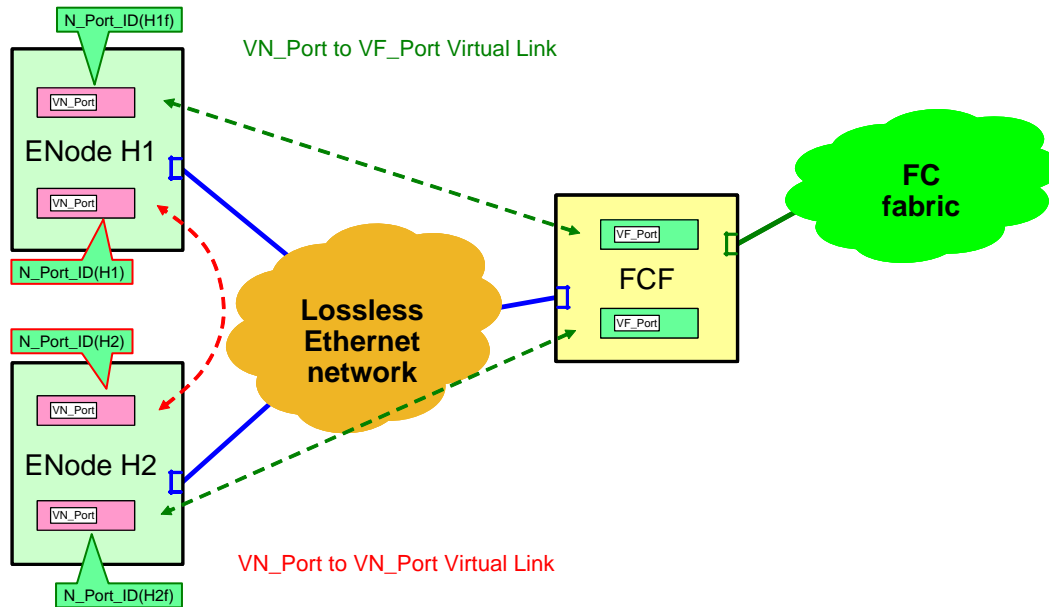


Figure 2 – Example of Mixed Configuration

Each VN2VN ENode MAC may instantiate a VN_Port to VF_Port Virtual Link with the FCF using the protocols defined in FC-BB-5. However, it has to be possible to instantiate VN_Port to VN_Port Virtual Links when the FCF is not connected to the Lossless Ethernet network and the VN_Port to VN_Port Virtual Links may remain operational when the FCF is disconnected from the Lossless Ethernet network. Therefore the VN_Ports involved in VN_Port to VN_Port Virtual Links shall be independent from the VN_Ports involved in VN_Port to VF_Port Virtual Links, because otherwise the VN_Port to VN_Port Virtual Links do not exist without the FCF.

Figure 2 shows that Locally Unique N_Port_IDs shall not conflict with and shall be independent from the N_Port_IDs assigned by a Fibre Channel Fabric. This ensures each VN_Port has a unique N_Port_ID. Locally Unique N_Port_IDs shall belong to the range 000001h .. 00FFFEh. This ensures that they do not conflict with N_Port_IDs assigned by a Fibre Channel Fabric, because a Fabric does not assign N_Port_IDs with Domain_ID zero.

A special case of VN_Port to VN_Port configuration is the point-to-point case shown in figure 3.

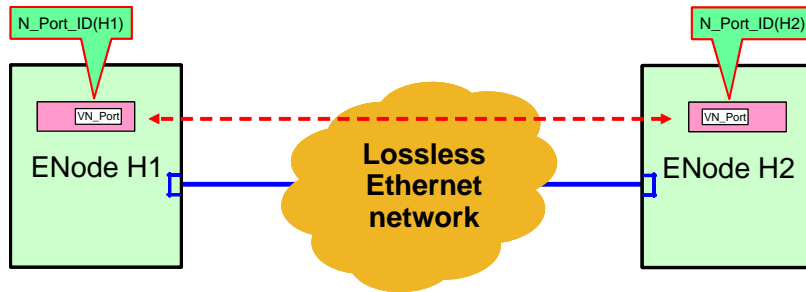


Figure 3 – VN_Port to VN_Port Point-to-Point Configuration

The point-to-point configuration is a special case because certain environments require to ensure that a maximum of two VN2VN ENode MACs are present over a Lossless Ethernet network and to operate in Fabric mode when one or more FCFs are detected (e.g., to instantiate only the VN_Port to VF_Port Virtual Links in the example configuration shown in figure 2). These requirements do not apply to the multipoint VN_Port to VN_Port configuration shown in figure 1, therefore VN2VN ENode MACs shall provide a way to enable the appropriate behavior (i.e., to operate in multipoint mode or in point-to-point mode). The point-to-point protocol is optimized for the case of two VN2VN ENode MACs connected through a single cable.

The protocol defined in this document enables VN2VN ENode MACs configured to operate in point-to-point mode to verify that only two ENode MACs are present over a Lossless Ethernet network, to select their Locally Unique N_Port_IDs (see 1.4.3) and to establish a VN_Port to VN_Port Virtual Link between the two VN2VN_Ports (see 1.5).

1.2 VN2VN ENode Functional Model

Figure 4 shows the functional model of a VN2VN ENode.

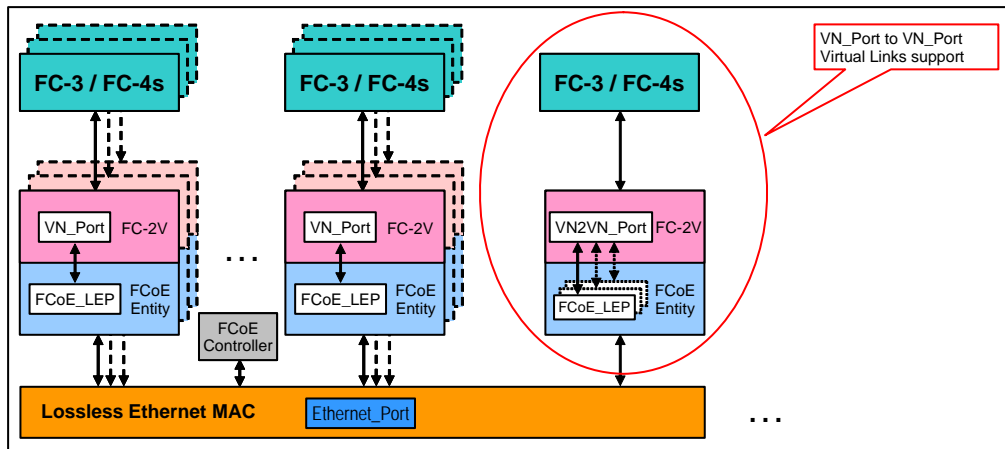


Figure 4 – VN2VN ENode Functional Model

A VN2VN ENode MAC has a VN_Port dedicated to the instantiation of VN_Port to VN_Port Virtual Links, called VN2VN_Port. The FCoE Controller of a VN2VN ENode MAC may perform FIP FLOGIs

with multiple remote VN2VN ENode MACs to instantiate multiple VN_Port to VN_Port Virtual Links. This results in multiple FCoE_LEPs associated with a VN2VN_Port. When operating in point-to-point mode, only one FCoE_LEP is associated with a VN2VN_Port.

The FCoE_LEP is the functional entity performing the encapsulation of FC frames into FCoE frames in transmission and the decapsulation of FCoE frames into FC frames in reception. An FCoE_LEP operates according to the MAC address of the local link end-point and the MAC address of the remote link end-point. When encapsulating FC frames into FCoE frames, the MAC address of the local link end-point shall be used as source address and the MAC address of the remote link end-point shall be used as destination address of the generated FCoE frame. When decapsulating FC frames from FCoE frames, the FCoE_LEP shall verify that the destination address of the received FCoE frame is equal to the MAC address of the local link end-point and shall verify that the source address of the received FCoE frame is equal to the MAC address of the remote link end-point. If either check fails the FCoE frame shall be discarded.

For a FCoE_LEP associated with a VN2VN_Port, the MAC address of the local link end-point is the FPMA associated with that VN2VN_Port and the remote link end-point address is the FPMA associated with the remote VN2VN_Port. Therefore the source MAC address of FCoE frames used for VN_Port to VN_Port Virtual Links is 'VN2VN-FC-MAP || S_ID' and the destination MAC address is 'VN2VN-FC-MAP || D_ID'. On receiving an FCoE frame, the FCoE_LEP associated with a VN2VN_Port shall verify that the least significant 24 bits of the source MAC address are equal to the S_ID of the encapsulated FC frame and that the least significant 24 bits of the destination MAC address are equal to the D_ID of the encapsulated FC frame. If any check fails the FCoE frame shall be discarded.

1.3 VN_Port to VN_Port Virtual Links

Figure 5 shows how the functional model defined in 1.2 models VN_Port to VN_Port Virtual Links for the multipoint case shown in figure 1.

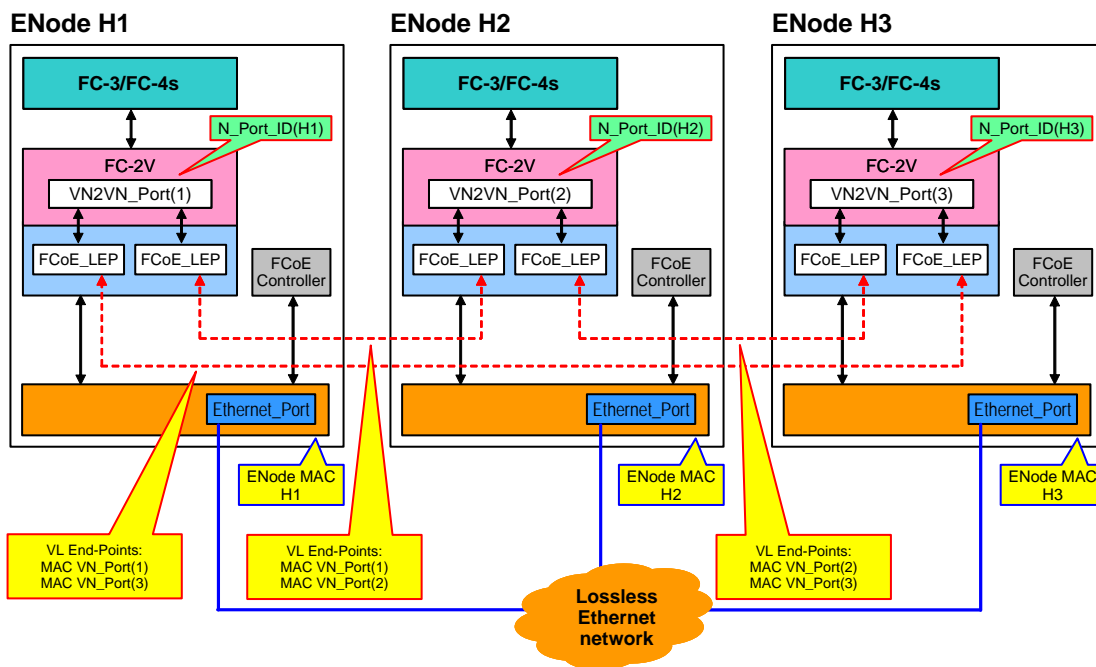


Figure 5 – VN_Port to VN_Port Virtual Links: Multipoint Example

Figure 6 shows how the functional model defined in 1.2 models VN_Port to VN_Port and VN_Port to VF_Port Virtual Links for the mixed configuration case shown in figure 2.

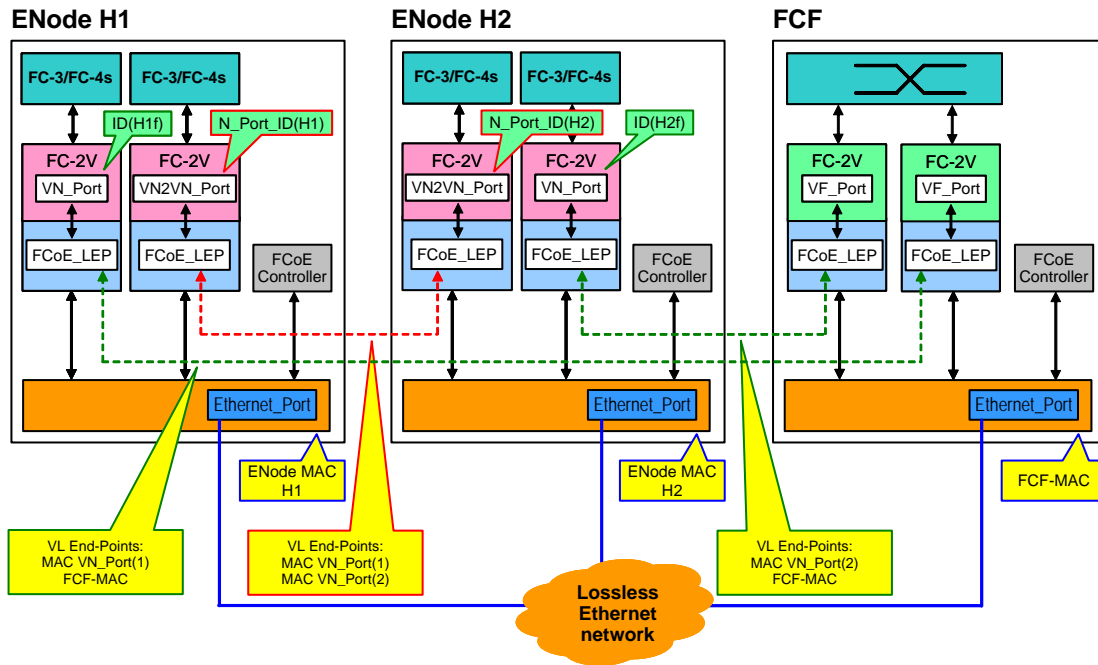


Figure 6 – VN_Port to VN_Port and VN_Port to VF_Port Virtual Links Example

Figure 7 shows how the functional model defined in 1.2 models VN_Port to VN_Port Virtual Links for the point-to-point case shown in figure 3.

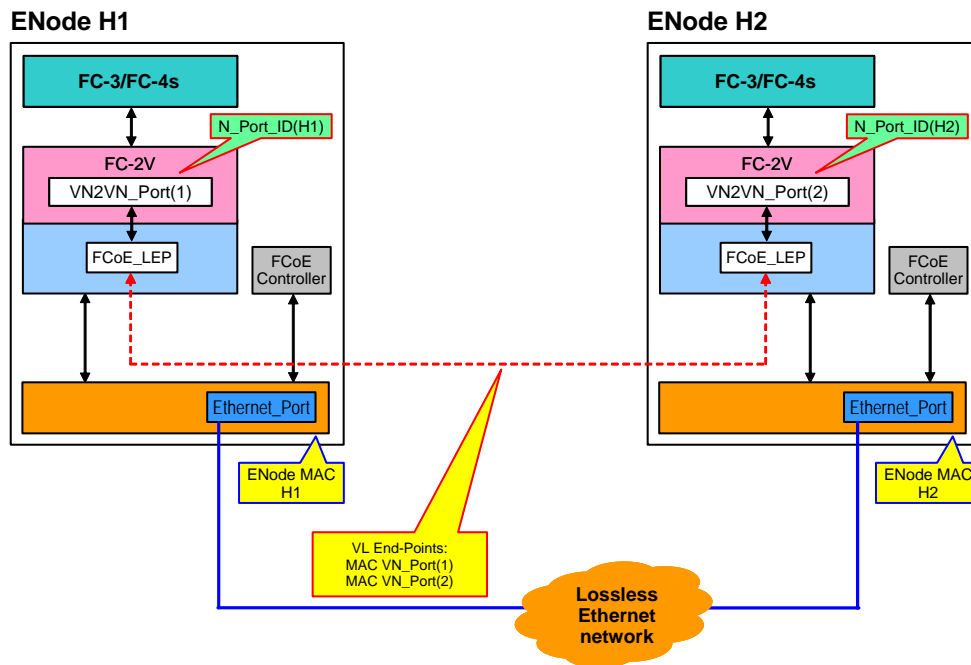


Figure 7 – VN_Port to VN_Port Virtual Link: Point-to-Point Example

VN_Port to VN_Port Virtual Links are instantiated on successful completion of point-to-point FIP FLOGI Exchanges (see 1.5). VN_Port to VN_Port Virtual Links are identified by the two VN_Port MAC addresses (FPMA) associated with the involved VN2VN_Ports.

1.4 Locally Unique N_Port_IDs

1.4.1 Overview

The Locally Unique N_Port_IDs protocol enables the establishment of VN_Port to VN_Port Virtual Links across a Lossless Ethernet network. The Locally Unique N_Port_ID protocol supports both multipoint (see 1.4.2) and point-to-point operations (see 1.4.3).

Two multicast MAC addresses are used by the protocol: All-VN2VN-ENode-MACs, used when a VN2VN ENode MAC operates in multipoint mode, and All-PT2PT-ENode-MACs, used when a VN2VN ENode MAC operates in point-to-point mode.

When becoming operational, a VN2VN ENode MAC shall enable reception of frames sent to both MAC addresses, All-VN2VN-ENode-MACs and All-PT2PT-ENode-MACs, and shall select a tentative Locally Unique N_Port_ID in the range 000001h .. 00FFFEh. A tentative Locally Unique N_Port_ID should be a recorded Locally Unique N_Port_ID if available (see 1.4.4), otherwise a tentative Locally Unique N_Port_ID should be generated using a pseudo-random number generator with a uniform distribution in the range 000001h .. 00FFFEh. The N_Port_Name of the VN2VN_Port for which a Locally Unique N_Port_ID is being selected should be used as one of the parameters to seed the pseudo-random number generator at its first invocation. In this way, different ENode MACs generate different numbers and a VN2VN_Port is usually associated with the same N_Port_ID each time it is enabled. The method to generate tentative Locally Unique N_Port_IDs shall be chosen so that different VN2VN ENode MACs are not likely to generate the same sequence of numbers (see annex A).

1.4.2 Multipoint Operation

1.4.2.1 Probing a Locally Unique N_Port_ID

After selecting a tentative Locally Unique N_Port_ID, a VN2VN ENode MAC operating in multipoint mode shall probe the Lossless Ethernet network to which it is connected to verify if the Locally Unique N_Port_ID is already in use. The VN2VN ENode MAC shall wait for a random time interval selected uniformly in the range zero to PROBE_WAIT milliseconds, and shall then transmit two multicast N_Port_ID Probe Requests (see 1.9.2) to All-VN2VN-ENode-MACs, spaced PROBE_WAIT milliseconds apart. If the selected tentative Locally Unique N_Port_ID is a recorded one, then the REC/P2P bit of the N_Port_ID Probe Requests shall be set to one, otherwise it shall be set to zero.

If during the period from the beginning of the probing process until ANNOUNCE_WAIT milliseconds after the last N_Port_ID Probe Request is sent the VN2VN ENode MAC receives any N_Port_ID Probe Reply (see 1.9.3), N_Port_ID Claim Notification (see 1.9.4) claiming the selected tentative Locally Unique N_Port_ID, or N_Port_ID Beacon (see 1.9.6) announcing the selected tentative Locally Unique N_Port_ID, then the VN2VN ENode MAC shall select a new random tentative Locally Unique N_Port_ID and repeat the process (i.e., wait for a random time interval and then transmit two N_Port_ID Probe Requests).

If during the same period the VN2VN ENode MAC receives any N_Port_ID Probe Request probing the selected tentative Locally Unique N_Port_ID, then the VN2VN ENode MAC shall check the N_Port_Name carried in the Vx_Port_Identification descriptor of the received N_Port_ID Probe Request and the value of the REC/P2P bit (see 1.7). If the REC/P2P bit is set to zero and the selected tentative Locally Unique N_Port_ID is a recorded one, then the VN2VN ENode MAC shall keep its selection and reply to the received N_Port_ID Probe Request with a N_Port_ID Probe Reply. If the

REC/P2P bit is set to one and the selected tentative Locally Unique N_Port_ID is not a recorded one, then the VN2VN ENode MAC shall select a new random tentative Locally Unique N_Port_ID and repeat the process (i.e., wait for a random time interval and then transmit two N_Port_ID Probe Requests) without replying to the received N_Port_ID Probe Request. If the REC/P2P bit is set to one and the selected tentative Locally Unique N_Port_ID is a recorded one, or the REC/P2P bit is set to zero and the selected tentative Locally Unique N_Port_ID is not a recorded one, then the processing is based on the value of the received N_Port_Name. If the received N_Port_Name is greater than the N_Port_Name of its VN2VN_Port then the VN2VN ENode MAC shall select a new random tentative Locally Unique N_Port_ID and repeat the process (i.e., wait for a random time interval and then transmit two N_Port_ID Probe Requests) without replying to the received N_Port_ID Probe Request, otherwise it shall keep its selection and reply to the received N_Port_ID Probe Request with a N_Port_ID Probe Reply.

N_Port_ID Probe Requests probing a different Locally Unique N_Port_ID, N_Port_ID Claim Notification claiming a different Locally Unique N_Port_ID, and N_Port_ID Beacon announcing a different Locally Unique N_Port_ID received during the same period shall be ignored.

A VN2VN ENode MAC should maintain a counter of the number of Locally Unique N_Port_ID conflicts it has experienced while trying to acquire a Locally Unique N_Port_ID. If the number of conflicts exceeds ten, then the VN2VN ENode MAC should report the situation in a vendor specific way as an indication of potential network failure and shall limit the rate at which it probes for new Locally Unique N_Port_IDs to no more than one new N_Port_ID per RATE_LIMIT_INTERVAL. This is to prevent multicast storms in pathological failure cases, such as a rogue VN2VN ENode MAC that answers all N_Port_ID Probe Requests, causing legitimate VN2VN ENode MACs to go into an infinite loop while attempting to select a usable Locally Unique N_Port_ID.

NOTE 1 – An ENode MAC connecting to a network with 1 300 ENodes has a 98% chance of selecting an unused Locally Unique N_Port_ID on the first try and a 99.96% chance of selecting an unused Locally Unique N_Port_ID within two tries. The probability that it will have to try more than ten times is about 10^{-17} .

At any time, upon receiving a N_Port_ID P2P Claim Notification (see 1.9.4) or a N_Port_ID P2P Beacon (see 1.9.6), a VN2VN ENode MAC operating in multipoint mode shall transmit a N_Port_ID Probe Request probing its N_Port_ID to All-VN2VN-ENode-MACs to indicate to the sending VN2VN ENode MAC operating in point-to-point mode that it is not in its expected network configuration.

1.4.2.2 Claiming a Locally Unique N_Port_ID

If by ANNOUNCE_WAIT milliseconds after the transmission of the last N_Port_ID Probe Request no conflicting N_Port_ID Probe Reply, N_Port_ID Probe Request, N_Port_ID Claim Notification, or N_Port_ID Beacon has been received, then the VN2VN ENode MAC shall claim the selected tentative Locally Unique N_Port_ID by transmitting a multicast N_Port_ID Claim Notification to All-VN2VN-ENode-MACs. A N_Port_ID Claim Notification notifies the other VN2VN ENode MACs on the Lossless Ethernet network of the selected Locally Unique N_Port_ID.

A VN2VN ENode MAC maintains the list of the reachable VN2VN_Ports in a VN2VN Neighbor Set. The VN2VN Neighbor Set is updated upon receiving N_Port_ID Claim Notifications and N_Port_ID Claim Responses (see 1.9.5) and checked upon receiving N_Port_ID Beacons.

Upon receiving a N_Port_ID Claim Notification claiming a different Locally Unique N_Port_ID, a VN2VN ENode MAC shall add the notifying VN2VN_Port to its VN2VN Neighbor Set and shall reply with a unicast N_Port_ID Claim Response. The N_Port_ID Claim Response shall be delayed by a random time uniformly distributed between 0 and 100 ms. The N_Port_ID Claim Notification provides the maximum FCoE size the VN2VN ENode MAC intends to use for VN_Port to VN_Port Virtual

Links, and the N_Port_ID Claim Response FIP PDU shall have a length equal to the minimum of the FCoE sizes supported by the two involved VN2VN ENode MACs.

Upon receiving a non conflicting N_Port_ID Claim Response a VN2VN ENode MAC shall add the responding VN2VN_Port to its VN2VN Neighbor Set. A VN2VN ENode MAC determines from its VN2VN Neighbor Set the VN2VN ENode MACs with which to establish VN_Port to VN_Port Virtual Links (i.e., the VN2VN Login Set), also on the basis of their FC-4 support. A VN2VN ENode may become ready to instantiate VN_Port to VN_Port Virtual Links upon receiving a N_Port_ID Claim Response. A VN2VN ENode wishing to discover all its VN2VN neighbors in order to select the VN2VN Login Set should wait for ANNOUNCE_WAIT milliseconds (i.e., the N_Port_ID Claim Responses timeout) upon transmitting a N_Port_ID Claim Notification to collect N_Port_ID Claim Responses.

When ready to instantiate VN_Port to VN_Port Virtual Links, a VN2VN ENode MAC shall transmit a multicast N_Port_ID Beacon to All-VN2VN-ENode-MACs and shall continue to transmit multicast N_Port_ID Beacons periodically every BEACON_PERIOD milliseconds. The BEACON_PERIOD period shall be randomized by adding a random delay uniformly distributed between 0 and 100 ms to avoid synchronized bursts of multicast traffic within the Ethernet network. Upon transmitting the first N_Port_ID Beacon the VN2VN ENode MAC may instantiate VN_Port to VN_Port Virtual Links (see 1.5).

If after having claimed a Locally Unique N_Port_ID a VN2VN ENode MAC receives a N_Port_ID Claim Notification claiming its Locally Unique N_Port_ID or a N_Port_ID Beacon announcing its Locally Unique N_Port_ID, then the VN2VN ENode MAC shall check the N_Port_Name carried in the Vx_Port_Identification descriptor of the received N_Port_ID Claim Notification or N_Port_ID Beacon.

NOTE 2 – This situation may happen only when joining two previously separated networks. In this case there is a chance of N_Port_ID conflict, resulting in one or more VN2VN_Ports needing to change their Locally Unique N_Port_IDs.

If the received N_Port_Name is greater than the N_Port_Name of its VN2VN_Port then the VN2VN ENode MAC shall implicitly deinstantiate all its VN_Port to VN_Port Virtual Links and select a new Locally Unique N_Port_ID (see 1.4.2.1). If the received N_Port_Name is lower than the N_Port_Name of its VN2VN_Port then the VN2VN ENode MAC shall transmit a N_Port_ID Claim Notification to All-VN2VN-ENode-MACs, to notify all other VN2VN ENode MACs on the network that it has that N_Port_ID. A VN2VN ENode MAC with a VN2VN_Port that has a Virtual Link with a VN2VN_Port that is changing its N_Port_ID shall implicitly deinstantiate that Virtual Link upon receiving a N_Port_ID Claim Notification associating that N_Port_ID to a different N_Port_Name.

Reception of a N_Port_ID Beacon from a VN2VN ENode MAC not listed in the VN2VN Neighbor Set is an indication of a network join. If a N_Port_ID Claim Notification has not been sent in the past ANNOUNCE_WAIT milliseconds, a VN2VN ENode MAC shall transmit a N_Port_ID Claim Notification to All-VN2VN-ENode-MACs upon receiving such a N_Port_ID Beacon. This N_Port_ID Claim Notification shall be delayed by a random time uniformly distributed between 0 and 100 ms.

If after having claimed a Locally Unique N_Port_ID a VN2VN ENode MAC receives a N_Port_ID Probe Request probing its Locally Unique N_Port_ID, then the VN2VN ENode MAC shall reply to the N_Port_ID Probe Request with a N_Port_ID Probe Reply.

N_Port_ID Probe Requests probing a different Locally Unique N_Port_ID shall be ignored.

At any time, upon receiving a N_Port_ID P2P Claim Notification (see 1.9.4) or a N_Port_ID P2P Beacon (see 1.9.6), a VN2VN ENode MAC operating in multipoint mode shall transmit a N_Port_ID Probe Request probing its N_Port_ID to All-VN2VN-ENode-MACs to indicate to the sending VN2VN ENode MAC operating in point-to-point mode that it is not in its expected network configuration.

1.4.3 Point-to-Point Operation

1.4.3.1 Claiming a Locally Unique N_Port_ID

After selecting a tentative Locally Unique N_Port_ID, a VN2VN ENode MAC operating in point-to-point mode shall claim it by sending a N_Port_ID P2P Claim Notification to All-PT2PT-ENode-MACs.

NOTE 3 – A VN2VN ENode MAC operating in point-to-point mode does not send N_Port_ID Probe Requests.

The N_Port_ID P2P Claim Notification (see 1.9.4) provides the maximum FCoE size the VN2VN ENode MAC intends to use for VN_Port to VN_Port Virtual Links. A responding N_Port_ID P2P Claim Response FIP PDU (see 1.9.5) shall have a length equal to the minimum of the FCoE sizes supported by the two involved VN2VN ENode MACs.

If no N_Port_ID P2P Claim Responses are received within ANNOUNCE_WAIT milliseconds, the VN2VN ENode MAC shall continue sending a N_Port_ID P2P Claim Notification every ANNOUNCE_WAIT milliseconds until a N_Port_ID Claim Response is received.

If more than one N_Port_ID P2P Claim Responses are received within ANNOUNCE_WAIT milliseconds, the VN2VN ENode MAC shall cease the point-to-point operations.

If one N_Port_ID P2P Claim Response is received within ANNOUNCE_WAIT milliseconds, the VN2VN ENode MAC shall add the responding VN2VN_Port to its Neighbor Set, shall transmit a multicast N_Port_ID P2P Beacon (see 1.9.6) to All-PT2PT-ENode-MACs and shall continue to transmit multicast N_Port_ID P2P Beacons periodically every BEACON_PERIOD milliseconds. Upon transmitting the first N_Port_ID P2P Beacon the VN2VN ENode MAC may instantiate a VN_Port to VN_Port Virtual Link with the VN2VN neighbor (see 1.5). A VN2VN ENode MAC may perform this processing as soon as a N_Port_ID Claim Response is received.

If a N_Port_ID P2P Claim Notification claiming its Locally Unique N_Port_ID is received, then the VN2VN ENode MAC shall check the N_Port_Name carried in the Vx_Port_Identification descriptor of the received N_Port_ID P2P Claim Notification. If the received N_Port_Name is greater than the N_Port_Name of its VN2VN_Port then the VN2VN ENode MAC shall select a new random Locally Unique N_Port_ID and shall reply with a unicast N_Port_ID P2P Claim Response indicating the new Locally Unique N_Port_ID. If the received N_Port_Name is lower than the N_Port_Name of its VN2VN_Port then the VN2VN ENode MAC shall keep its selected Locally Unique N_Port_ID and shall reply with a unicast N_Port_ID P2P Claim Response indicating the originally selected Locally Unique N_Port_ID.

If a N_Port_ID P2P Claim Notification claiming a different Locally Unique N_Port_ID is received, then the VN2VN ENode MAC shall reply with a unicast N_Port_ID P2P Claim Response.

If after transmitting the first N_Port_ID P2P Beacon the VN2VN ENode MAC receives a N_Port_ID P2P Claim Notification or a N_Port_ID P2P Claim Response or a N_Port_ID P2P Beacon from a VN2VN_Port different than the one present in its Neighbor Set, the VN2VN ENode MAC shall cease the point-to-point operations.

At any time, upon receiving a N_Port_ID Probe Request, a N_Port_ID Claim Notification, a N_Port_ID Beacon, or a FIP Advertisement, a VN2VN ENode MAC operating in point-to-point mode shall cease the point-to-point operations.

1.4.4 Persistence of Locally Unique N_Port_IDs

VN2VN ENode MACs that are equipped with persistent storage may record the Locally Unique N_Port_ID they have selected for themselves. The recorded Locally Unique N_Port_ID may be also administratively configured. On booting, VN2VN ENode MACs with a recorded Locally Unique N_Port_ID should use that Locally Unique N_Port_ID as their first tentative Locally Unique N_Port_ID. This increases the stability of N_Port_IDs (e.g., if some VN2VN ENode MACs are powered off, then when they are powered on they resume using the previous Locally Unique N_Port_IDs, instead of picking different N_Port_IDs and potentially having to resolve conflicts).

Depending on local configuration, a VN2VN ENode MAC may stop the processing of the Locally Unique N_Port_ID selection protocol and report the situation in a vendor specific way if it detects that its recorded Locally Unique N_Port_ID is already in use on the network.

1.4.5 Timers and Constants

Table 1 shows the timers and constants of the Locally Unique N_Port_IDs selection protocol.

Table 1 – Locally Unique N_Port_IDs Timers and Constants

Parameter	Value
PROBE_WAIT	100 ms
ANNOUNCE_WAIT	400 ms
RATE_LIMIT_INTERVAL	10 000 ms
BEACON_PERIOD	8 000 ms
All-VN2VN-ENode-MACs	01-10-18-01-00-04 ? <TBD>
All-PT2PT-ENode-MACs	01-10-18-01-00-05 ? <TBD>

1.5 VN_Port to VN_Port Virtual Link Instantiation Protocol

A VN2VN ENode MAC, operating in either multipoint or point-to-point mode, instantiates VN_Port to VN_Port Virtual Links on successful completion of a point-to-point FIP FLOGI, as defined in FC-LS-2. Both FIP FLOGI Request and LS_ACC shall have the Locally Unique N_Port_ID of the originating VN2VN_Port as S_ID, the Locally Unique N_Port_ID of the destination VN2VN_Port as D_ID for the point-to-point FLOGI protocol, and the originating VN_Port FPMA in the MAC Address descriptor. The MAC addresses of the FIP FLOGI Request and LS_ACC shall be the ENode MAC addresses of the involved VN2VN_Ports. As specified in FC-LS-2, the VN2VN_Port with the greater N_Port_Name proceeds to N_Port Login, with the PLOGI ELSs encapsulated in FCoE. Both FCoE PLOGI Request and LS_ACC shall have the Locally Unique N_Port_ID of the originating VN2VN_Port as S_ID and the Locally Unique N_Port_ID of the destination VN2VN_Port as D_ID. Upon completion of FCoE PLOGI the involved VN_Ports operate in point-to-point mode (see FC-LS-2).

A point-to-point FIP FLOGI Request coming from a VN2VN_Port not listed in the VN2VN Neighbor Set shall be rejected.

A VN_Port to VN_Port Virtual Link is explicitly deinstantiated by performing a FIP LOGO, that deinstantiates the FCoE_LEPs and performs a N_Port logout. The S_ID and D_ID on the encapsulated LOGO ELS shall be set to the Locally Unique N_Port_IDs of the involved VN2VN_Ports.

1.6 VN_Port to VN_Port Virtual Link Maintenance Protocol

To deal with local physical layer faults, a VN2VN ENode MAC shall de-instantiate all its VN_Port to VN_Port Virtual Links upon detecting that its physical layer is not operational.

To deal with non-local faults, the FCoE Controller of a VN2VN ENode MAC operating in multipoint mode shall continuously verify the state of the VN_Port to VN_Port Virtual Links by verifying received N_Port_ID Beacons per each VN2VN_Port in the VN2VN Login Set. The FCoE Controller of a VN2VN ENode MAC operating in point-to-point mode shall continuously verify the state of the VN_Port to VN_Port Virtual Link by verifying received N_Port_ID P2P Beacons per the VN2VN_Port in the VN2VN Login Set.

N_Port_ID Beacons and N_Port_ID P2P Beacons are expected to be received every BEACON_PERIOD. If N_Port_ID Beacons or N_Port_ID P2P Beacons from a VN2VN_Port are not received within 2.5 * BEACON_PERIOD, that VN2VN_Port shall be removed from the VN2VN Neighbor Set and from the VN2VN Login Set and VN_Port to VN_Port Virtual Links with that remote VN2VN_Port, if any, shall be implicitly de-instantiated.

1.7 VN2VN FIP Frames

The VN2VN FIP Protocol Code and FIP Subcode field values and operations are specified in table 2.

Table 2 – VN2VN FIP Protocol Code and FIP Subcode Field Values

FIP Protocol Code	FIP Subcode	FIP Protocol Messages	Reference
0005h	01h	N_Port_ID Probe Request	1.9.2
	02h	N_Port_ID Probe Reply	1.9.3
	03h	N_Port_ID Claim Notification	1.9.4
	04h	N_Port_ID Claim Response	1.9.5
	05h	N_Port_ID Beacon	1.9.6

An additional flag is defined, the REC/P2P flag (bit 3 of word 1 of the encapsulated FIP operation).

1.8 VN2VN FIP Descriptors

1.8.1 Overview

VN2VN operations require the additional descriptors shown in table 3.

Table 3 – FIP Descriptor Types

Range	Type	FIP Descriptor	Reference
Critical	15	FC-4 Attributes	1.8.2

1.8.2 FIP FC-4 Attributes Descriptor

The FIP FC-4 Attributes descriptor is used in FIP operations as shown in table 5.

The FIP FC-4 Attributes descriptor format shall be as specified in table 4.

Table 4 – FIP FC-4 Attributes Descriptor Format

Bit	3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	0	9	8	7	6	5	4	3	2	1	0																																	
Word	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0																																
0	Type = 0Fh								Length = 29h								Reserved																																															
1 .. 8	MSB																																FC-4 Types																LSB															
9 .. 40	MSB																																FC-4 Features																LSB															

FC-4 Types: the FC-4 Types object, as defined in FC-GS-6.

FC-4 Features: the FC-4 Features object, as defined in FC-GS-6.

1.9 VN2VN FIP Operations

1.9.1 Overview

Table 5 shows the FIP operations used by VN2VN ENodes.

Table 5 – FIP Operation Descriptors and Order

FIP Operation	FIP Protocol Code/Subcode	Originator	Expected Descriptors and Order
N_Port_ID Probe Request	0005h / 01h	VN2VN ENode	1) MAC address 2) Name_Identifier 3) Vx_Port Identification
N_Port_ID Probe Reply	0005h / 02h	VN2VN ENode	1) MAC address 2) Name_Identifier 3) Vx_Port Identification
N_Port_ID Claim Notification	0005h / 03h	VN2VN ENode	1) MAC address 2) Name_Identifier 3) Vx_Port Identification 4) FC-4 Attributes 5) Max FCoE Size
N_Port_ID Claim Response	0005h / 04h	VN2VN ENode	1) MAC address 2) Name_Identifier 3) Vx_Port Identification 4) FC-4 Attributes 5) Max FCoE Size
N_Port_ID Beacon	0005h / 05h	VN2VN ENode	1) MAC address 2) Name_Identifier 3) Vx_Port Identification

1.9.2 N_Port_ID Probe Request

As shown in table 5, an N_Port_ID Probe Request operation contains a MAC address descriptor, a Name_Identifier descriptor and a Vx_Port Identification descriptor. The REC/P2P flag (see 1.7) in a N_Port_ID Probe Request frame shall be set to one if the probed tentative Locally Unique N_Port_ID is a recorded one (see 1.4.4), otherwise shall be set to zero.

The MAC address field in the MAC address descriptor shall be set to the ENode MAC address of the originating VN2VN ENode MAC. The Name_Identifier field in the Name_Identifier descriptor shall be set to the Node_Name of the originating VN2VN_Port. In the Vx_Port Identification descriptor, the Port_Name field shall be set to the N_Port_Name of the VN2VN_Port whose tentative Locally Unique N_Port_ID is being probed, the Address Identifier field shall be set to the tentative Locally Unique N_Port_ID, and the MAC address field shall be set to the tentative VN_Port MAC address (FPMA).

An N_Port_ID Probe Request is multicast (i.e., addressed to the All-VN2VN-ENode-MACs multicast address) and uses the ENode MAC address as source address.

1.9.3 N_Port_ID Probe Reply

As shown in table 5, an N_Port_ID Probe Reply operation contains a MAC address descriptor, a Name_Identifier descriptor and a Vx_Port Identification descriptor. The REC/P2P flag (see 1.7) is not used by N_Port_ID Probe Reply frames and shall be set to zero.

The MAC address field in the MAC address descriptor shall be set to the ENode MAC address of the originating VN2VN ENode MAC. The Name_Identifier field in the Name_Identifier descriptor shall be set to the Node_Name of the originating VN2VN_Port. In the Vx_Port Identification descriptor, the Port_Name field shall be set to the N_Port_Name of the originating VN2VN_Port, the Address Identifier field shall be set to the Locally Unique N_Port_ID of the originating VN2VN_Port, and the MAC address field shall be set to the VN_Port MAC address (FPMA) of the originating VN2VN_Port.

An N_Port_ID Probe Reply is unicast (i.e., it replies to a multicast N_Port_ID Probe Request) and uses the ENode MAC address as source address.

1.9.4 N_Port_ID Claim Notification

As shown in table 5, an N_Port_ID Claim Notification operation contains a MAC address descriptor, a Name_Identifier descriptor, a Vx_Port Identification descriptor, an FC-4 Attributes descriptor, and a Max FCoE Size descriptor.

The MAC address field in the MAC address descriptor shall be set to the ENode MAC address of the originating VN2VN ENode MAC. The Name_Identifier field in the Name_Identifier descriptor shall be set to the Node_Name of the originating VN2VN_Port. In the Vx_Port Identification descriptor, the Port_Name field shall be set to the N_Port_Name of the originating VN2VN_Port, the Address Identifier field shall be set to the claimed Locally Unique N_Port_ID, and the MAC address field shall be set to the associated VN_Port MAC address (FPMA). The FC-4 Types and FC-4 Features fields of the FC-4 Attributes descriptor shall be set to the FC-4 Type and Features supported by the originating VN2VN_Port. The Max_FCoE_Size field in the Max FCoE Size descriptor shall be set to the maximum FCoE PDU size the ENode MAC intends to use for FCoE traffic.

An N_Port_ID Claim Notification is multicast and uses the ENode MAC address as source address. When the REC/P2P flag is set to zero an N_Port_ID Claim Notification is addressed to the All-VN2VN-ENode-MACs multicast address. When the REC/P2P flag is set to one an N_Port_ID Claim Notification is addressed to the All-PT2PT-ENode-MACs multicast address and is referred to as an N_Port_ID P2P Claim Notification.

1.9.5 N_Port_ID Claim Response

As shown in table 5, an N_Port_ID Claim Response operation contains a MAC address descriptor, a Name_Identifier descriptor, a Vx_Port Identification descriptor, an FC-4 Attributes descriptor, and a Max FCoE Size descriptor.

The MAC address field in the MAC address descriptor shall be set to the ENode MAC address of the originating VN2VN ENode MAC. The Name_Identifier field in the Name_Identifier descriptor shall be set to the Node_Name of the originating VN2VN_Port. In the Vx_Port Identification descriptor, the Port_Name field shall be set to the N_Port_Name of the originating VN2VN_Port, the Address Identifier field shall be set to the Locally Unique N_Port_ID of the originating VN2VN_Port, and the MAC address field shall be set to the VN_Port MAC address (FPMA) of the originating VN2VN_Port. The FC-4 Types and FC-4 Features fields of the FC-4 Attributes descriptor shall be set to the FC-4 Type and Features supported by the originating VN2VN_Port. The Max_FCoE_Size field in the Max FCoE Size descriptor shall be set to the maximum FCoE PDU size the ENode MAC intends to use for FCoE traffic. The FIP_Pad field shall be used to extend the FIP PDU to have a length that matches the minimum of the FCoE sizes supported by the two involved VN2VN ENode MACs.

An N_Port_ID Claim Response is unicast and uses the ENode MAC address as source address. When the REC/P2P flag is set to zero an N_Port_ID Claim Response replies to a multicast N_Port_ID Claim Notification. When the REC/P2P flag is set to one an N_Port_ID Claim Notification replies to a multicast N_Port_ID P2P Claim Notification and is referred to as an N_Port_ID P2P Claim Response.

1.9.6 N_Port_ID Beacon

As shown in table 5, a N_Port_ID Beacon operation contains a MAC address descriptor, a Name_Identifier descriptor and a Vx_Port Identification descriptor.

The MAC address field in the MAC address descriptor shall be set to the ENode MAC address of the originating VN2VN ENode MAC. The Name_Identifier field in the Name_Identifier descriptor shall be set to the Node_Name of the originating VN2VN_Port. In the Vx_Port Identification descriptor, the Port_Name field shall be set to the N_Port_Name of the originating VN2VN_Port, the Address Identifier field shall be set to the Locally Unique N_Port_ID of the originating VN2VN_Port, and the MAC address field shall be set to the VN_Port MAC address (FPMA) of the originating VN2VN_Port.

A N_Port_ID Beacon is multicast and uses the VN_Port MAC address as source address. When the REC/P2P flag is set to zero an N_Port_ID Beacon is addressed to the All-VN2VN-ENode-MACs multicast address. When the REC/P2P flag is set to one an N_Port_ID Beacon is addressed to the All-PT2PT-ENode-MACs multicast address and is referred to as an N_Port_ID P2P Beacon.

Annex A: Pseudo-Random Locally Unique N_Port_IDs (Informative)

This annex describes a possible method for generating pseudo-random Locally Unique N_Port_IDs. This method is computationally simple and keeps the number of N_Port_ID probes to a minimum.

NOTE 4 – This method falls under the guidance of the well-known Perl acronym, **TIMTOWTDIBSCINABTE** (pronounced Tim Toady Bicarbonate). This Perl acronym stands for: "There is more than one-way to do it, but sometimes consistency is not a bad thing either".

This generation method uses a 12-byte internal state that is initialized to the concatenation of the N_Port_Name and the low order 32 bits of the ENode MAC address.

NOTE 5 – From a purely mathematical perspective, it is possible to initialize the state with the concatenation of the 64-bit N_Port_Name and the low order 24 bits of the ENode MAC address. However many CPU architectures are optimized to perform Arithmetic and Logic computation on data that is a power of 2 bit length of at least 8 bits. To avoid the extra masking required to perform 24-bit increments, this method uses the 32 low order bits of the ENode MAC address.

The following C code describes the 12-byte internal state:

```
/*
 * The internal state is a concatenation of two values
 */
struct state_data {
    u_int64_t b64;          /* initialized to N_Port_Name */
    u_int32_t b32;          /* initialized to low order MAC bits */
};

#define STATE_LEN (sizeof(struct state_data))

union internal_state {
    struct state_data data; /* state as 64 & 32 bit values */
    u_int8_t byte[STATE_LEN]; /* the 12 bytes to FNV-1a hash */
};
```

The following C code initializes the 12-byte internal state:

```
union internal_state state;
u_int64_t n_port_name; /* ENode N_Port_Name */
u_int64_t enode_mac; /* ENode MAC as a 64 bit value */

state.data.b64 = n_port_name;
state.data.b32 = (u_int32_t) enode_mac;
```

This generation method first tweaks the internal state by incrementing the trailing 32-bit portion. Then a 32-bit FNV-1a hash is computed over the entire 12-byte internal state. That 32-bit hash value is XOR folded into a 16-bit value.

The XOR fold produces 16-bit values distributed over the range 0000h .. FFFFh inclusive. However, Locally Unique N_Port_IDs are in the range 0001h .. FFFEh, therefore the above code is repeated in case the XOR fold generates the values 0000h or FFFFh.

The following C code implements the procedure described in the previous paragraph:

```

union internal_state *state; /* pointer to state */
u_int32_t hash;             /* 32 bit FNV-1a hash value */
u_int16_t tentative_id;    /* tentative Locally Unique N_Port_ID */

do {
    /* tweak the state */
    ++state->data.b32;

    /* FNV-1a hash the state */
    hash = fnv1a_32bit_hash_state( state );

    /* XOR fold 32 bits into 16 bits */
    tentative_id = (u_int16_t) (hash ^ (hash>>16));

} while (tentative_id == 0 || tentative_id == 0xFFFF);

```

When a VN2VN ENode MAC has to generate a tentative Locally Unique N_Port_ID, it is useful avoiding retry recently generated tentative tentative Locally Unique N_Port_IDs. For this reason, this method maintains a circular buffer of the 16 previously generated tentative Locally Unique N_Port_IDs. The number 16 is chosen because it is a power of two. The following C code declares this table and its associated indices:

```

/*
 * parameters of the previously generated Locally Unique N_Port_IDs
 *
 * ID_MEMORY_BITS - power of 2 size of idmem, must be <= 16
 */
#define ID_MEMORY_BITS 4
#define ID_MEMORY (1<<ID_MEMORY_BITS)

/*
 * previously generated Locally Unique N_Port_IDs table
 *
 * 0 indicates an unused entry
 */
u_int16_t idmem[ID_MEMORY];

/*
 * working indices into the idmem table
 */
struct idmem_indices {
    /* check below to this slot for past Locally Unique N_Port_IDs */
    u_int32_t beyond;

    /* next slot to receive new tentative Locally Unique N_Port_ID */
    u_int16_t next : ID_MEMORY_BITS;
} idmem_indx;

```

The following C code initializes the above declared table and its associated indices:

```

memset(idmem, 0, sizeof(idmem));
idmem_indx.next = 0;
idmem_indx.beyond = 0;

```

The following C code implements this generation method as a C function that returns a tentative Locally Unique N_Port_ID in the range 0001h .. FFFEh, while avoiding returning the 16 most recently generated tentative Locally Unique N_Port_IDs:

```

/*
 * Generate a tentative Locally Unique N_Port_ID
 *
 * given:
 *     state - pointer to internal state
 *
 * returns:
 *     tentative Locally Unique N_Port_ID in the range [1,0xFFFE]
 *     that is not among the ID_MEMORY most recent return values
 */
u_int16_t
generate_tentative_LUID(union stuff *state)
{
    u_int32_t hash;          /* 32 bit FNV-1a hash of state */
    u_int16_t tentative_id; /* tentative Locally Unique N_Port_ID */
    int regen;              /* if 1 generate another Locally Unique N_Port_ID */
    u_int32_t i;

    /*
     * generate a tentative Locally Unique N_Port_ID not returned recently
     */
    do {

        /*
         * generate a non-reserved tentative Locally Unique N_Port_ID
         */
        do {
            /* tweak state */
            ++state->data.b32;

            /* generate 32-bit FNV-1a hash value */
            hash = fnv1a_32bit_hash_state( state );

            /* XOR fold 32 bits into 16 bits */
            tentative_id = (u_int16_t) (hash ^ (hash>>16));

        } while (tentative_id == 0 || tentative_id == 0xFFFF);

        /*
         * scan the table of recent tentative Locally Unique N_Port_IDs
         * reject if match found
         */
        for (regen=0, i=0; i < idmem_indx.beyond; ++i) {
            if (tentative_id == idmem[i]) {
                regen = 1;
                break;
            }
        }
    }
}

```

```

/*
 * remember a not recent tentative Locally Unique N_Port_ID
 */
if (regen == 0) {

    /*
     * remember the tentative Locally Unique N_Port_ID
     * advance next, OK to cycle back to 0
     */
    idmem[idmem_indx.next++] = tentative_id;

    /* if the table grows, remember new beyond slot */
    if (idmem_indx.beyond < ID_MEMORY) {
        ++idmem_indx.beyond;
    }
}

} while (regen != 0);

/* return a tentative Locally Unique N_Port_ID */
return tentative_id;
};

```

The return value of the `generate_tentative_LUID` function may be used as a tentative Locally Unique N_Port_ID as described in 1.4. Before the `generate_tentative_LUID` function is called for the first time, the state has to be initialized in the manner described above.

The core of this generation method involves computing a 32-bit hash of the 12-byte internal state. To minimize the number of probes, it is of the utmost importance that the hash returns substantially different values given substantially similar states. In particular, it is important for the hash function to disperse its output given a change of only a few bits of input. This method uses the 32-bit Fowler/Noll/Vo 1a hash (FNV-1a, see <http://www.isthe.com/chongo/tech/comp/fnv/index.html>), that has excellent dispersion properties. The FNV-1a is a widely used public domain algorithm that is easy to implement and requires little CPU resources to compute. The code of the FNV-1a code requires 2 CPU ops per byte of hashed data.

One of the key advantages of the FNV-1a hash is that it is very simple to implement. Start with an initial hash value of `FNV_offset_basis`. For each byte in the input, XOR the hash value with the next byte of data and then multiply the hash by the `FNV_prime`:

```

hash = FNV_offset_basis
for each byte_of_data to be hashed
    hash = hash XOR byte_of_data
    hash = hash * FNV_prime
return hash

```

The `FNV_prime` and `FNV_offset_basis` are dependent on the size of the hash, as shown in table A.1.

Table A.1 – FNV-1a Parameters

FNV Hash Size	FNV_prime	FNV_offset_basis
32	$2^{24} + 2^8 + 93h = 16777619$	2166136261
64	$2^{40} + 2^8 + B3h = 1099511628211$	14695981039346656037

The following C code performs a 32-bit FNV-1a hash of the 12-byte internal state:

```
#define FNV_32_PRIME (16777619U)          /* 32 bit FNV prime */
#define FNV_32_BASIS (2166136261U)      /* 32 bit FNV basis */

u_int32_t
fnvla_32bit_hash_state(union stuff *state)
{
    u_int32_t hash = FNV_32_BASIS;      /* FNV-1a hash state */
    size_t i;

    /* compute 32 bit FNV-1a hash of state */
    for (i=0; i < STATE_LEN; ++i) {
        hash = (hash ^ state->byte[i]) * FNV_32_PRIME;
    }

    /* return 32 bit FNV-1a hash of state */
    return hash;
}
```

Implementations with 64-bit CPUs may use 64-bit arithmetic:

```
union internal_state *state; /* pointer to state */
u_int64_t hash;             /* 64 bit FNV-1a hash value */
u_int16_t tentative_id;    /* tentative Locally Unique N_Port_ID */

do {
    /* tweak the state */
    ++state->data.b32;

    /* FNV-1a hash the state */
    hash = fnvla_64bit_hash_state( state );

    /* XOR fold 64 bits into 16 bits */
    trail_id = (u_int16_t)
        (hash ^ (hash>>16) ^ (hash>>32) ^ (hash>>48));

} while (tentative_id == 0 || tentative_id == 0xFFFF);
```

The following C code performs a 64-bit FNV-1a hash of the 12 bytes of the internal state:

```
#define FNV_64_PRIME (1099511628211ULL)  /* 64 bit FNV prime */
#define FNV_64_BASIS (14695981039346656037ULL) /* 64 bit FNV basis */

u_int64_t
fnvla_64bit_hash_state(union stuff *state)
{
    u_int64_t hash = FNV_64_BASIS; /* FNV-1a hash state */
    size_t i;

    /* compute 64 bit FNV-1a hash of state */
    for (i=0; i < STATE_LEN; ++i) {
        hash = (hash ^ state->byte[i]) * FNV_64_PRIME;
    }

    /* return 64 bit FNV-1a hash of state */
    return hash;
}
```

It is possible to quickly compute the FNV hash on CPUs without a multiply unit or where the multiply operation is too slow. All FNV_primes contain only 6 or 7 bits with the value of one. Therefore the multiply may be replaced with a set of shifts and additions.

The following C code performs a 32-bit FNV-1a hash of the 12-byte internal state without using the multiply operation:

```
u_int32_t
fnvla_32bit_hash_state_nomult(union stuff *state)
{
    u_int32_t hash = FNV_32_BASIS; /* FNV-1a hash state */
    size_t i;

    /* compute 32 bit FNV-1a hash of state */
    for (i=0; i < STATE_LEN; ++i) {
        hash ^= (u_int32_t) state->byte[i];
        hash += (hash<<1) + (hash<<4) + (hash<<7) +
                (hash<<8) + (hash<<24);
    }

    /* return 32 bit FNV-1a hash of state */
    return hash;
}
```

The following C code performs a 64-bit FNV-1a hash of the 12-byte internal state without using the multiply operation:

```
u_int64_t
fnvla_64bit_hash_state_nomult(union stuff *state)
{
    u_int64_t hash = FNV_64_BASIS; /* FNV-1a hash state */
    size_t i;

    /* compute 64 bit FNV-1a hash of state */
    for (i=0; i < STATE_LEN; ++i) {
        hash ^= (u_int64_t) state->byte[i];
        hash += (hash<<1) + (hash<<4) + (hash<<5) +
                (hash<<7) + (hash<<8) + (hash<<40);
    }

    /* return 64 bit FNV-1a hash of state */
    return hash;
}
```

Annex B: Increasing FC-BB_E Robustness Using Access Control Lists (Informative)

B.1 Overview

When security threats exist in a Locally Unique N_Port_ID configuration, it is important to protect the FCoE traffic with appropriate FCoE ACLs, documented in this annex.

B.2 Prevention of FCoE Traffic

The ACL described in this subclause is appropriate for a Ethernet bridge port intended to block all FCoE traffic:

```
SAPre = VN2VN-FC-MAP, deny;
Type = FCoE_TYPE, deny;
Type = FIP_TYPE, deny
```

B.3 FCoE Perimeter ACL

The set of ACLs described in this subclause is appropriate for a Ethernet bridge port intended to enable Locally Unique N_Port_IDs for a VN2VN ENode MACs connected to it. This set of ACLs protects also from accidental network joins, that otherwise may cause unwanted N_Port_ID changes. The default perimeter ACL for Locally Unique N_Port_IDs is:

```
SAPre = VN2VN-FC-MAP, deny;
Type = FCoE_TYPE, deny;
Type = FIP_TYPE, permit
```

The first entry blocks any frame that has a source address used for FCoE, including N_Port_ID Beacons. The second entry provide additional protection blocking any traffic with the FCoE Ethertype. The third entry allows ENode to ENode FIP communication. This ACL is sufficient to protect from accidental network joins.

The default perimeter ACL may be automatically updated to enable the Locally Unique N_Port_ID selection protocol to proceed. After receiving an N_Port_ID Claim Notification, the ACL becomes:

```
SA = src VN2VN_Port MAC, DA = All-VN2VN-ENode-MACs, Type = FIP_TYPE, permit;
SAPre = VN2VN-FC-MAP, deny;
Type = FCoE_TYPE, deny;
Type = FIP_TYPE, permit
```

The first entry allows transmission of N_Port_ID Beacons.

Upon detection of the point-to-point FIP FLOGI LS_ACC, the ACL becomes:

```
SA = src VN2VN_Port MAC, DA = dst VN2VN_Port MAC, Type = FCoE_TYPE, permit;
SA = src VN2VN_Port MAC, DA = All-VN2VN-ENode-MACs, Type = FIP_TYPE, permit;
SAPre = VN2VN-FC-MAP, deny;
Type = FCoE_TYPE, deny;
Type = FIP_TYPE, permit
```

The first entry allows VN_Port to VN_Port FCoE traffic. In order to not keep track of all possible destinations, an implementation may replace the first entry with the following one:

```
SA = src VN2VN_Port MAC, DA = DAPre = VN2VN-FC-MAP, Type = FCoE_TYPE, permit;
```

B.4 Graceful Handling of Network Joins

Duplication of Locally Unique N_Port_IDs across independent networks may have adverse effects when a network administrator desires to join two networks. The set of ACLs described in this sub-clause is appropriate for a Ethernet bridge port intended to be used to join two independent networks.

The starting point is the ACL that protects each of the two networks from undesired joins:

```
SApre = VN2VN-FC-MAP, deny;  
Type = FCoE_TYPE, deny
```

After a cable connects the two networks, this ACL should become:

```
SApre = VN2VN-FC-MAP, Type = FIP_TYPE, permit;  
SApre = VN2VN-FC-MAP, deny;  
Type = FCoE_TYPE, deny
```

The first entry allows the flow of periodic multicast N_Port_ID Beacons, that enable the detection and resolution of N_Port_ID conflicts, while the last entry ensures that FCoE traffic still does not flow across the two networks. With this ACL in place N_Port_ID conflicts are resolved and Ethernet forwarding tables reconfigurations are possible, however it is not possible for the FCoE traffic to go from one network to the other one, therefore no FCoE traffic is delivered to an unintended VN2VN_Port. This ACL should stay in place for at least $2.5 * \text{BEACON_PERIOD}$ milliseconds, the timeout for VN_Port to VN_Port Virtual Links, therefore if N_Port_ID Beacons are not received in this timeframe the Virtual Links are deinstantiated. During this time N_Port_ID conflicts are resolved.

After at least $2.5 * \text{BEACON_PERIOD}$ milliseconds the ACL should become:

```
SApre = VN2VN-FC-MAP, Type = FIP_TYPE, permit;  
SApre = VN2VN-FC-MAP, Type = FCoE_TYPE, permit;  
SApre = VN2VN-FC-MAP, deny
```

At this point the two networks are joined and FCoE traffic is able to flow.

NOTE 6 – Only the VN2VN_Ports that happen to have the same N_Port_ID are affected by a network join, any other VN2VN_Port is not affected.