NVO3 Working Group   Y. Li
INTERNET-DRAFT   D. Eastlake
Intended Status: Informational   Huawei Technologies
                                                               L. Kreeger
                                                              Arrcus, Inc
                                                                T. Narten
                                                                      IBM
                                                                 D. Black
                                                                 Dell EMC
Expires: September 14, 2018   March 13, 2018

Split Network Virtualization Edge (Split-NVE) Control Plane Requirements

draft-ietf-nvo3-hpvr2nve-cp-req-17

Abstract

In a Split Network Virtualization Edge (Split-NVE) architecture, the functions of the NVE (Network Virtualization Edge) are split across a server and an external network equipment which is called an external NVE. The server-resident control plane functionality resides in control software, which may be part of hypervisor or container management software; for simplicity, this document refers to the hypervisor as the location of this software.

Control plane protocol(s) between a hypervisor and its associated external NVE(s) are used by the hypervisor to distribute its virtual machine networking state to the external NVE(s) for further handling. This document illustrates the functionality required by this type of control plane signaling protocol and outlines the high level requirements. Virtual machine states as well as state transitioning are summarized to help clarify the protocol requirements.

Status of this Memo

Copyright and License Notice

Table of Contents

1. Introduction

In the Split-NVE architecture shown in Figure 1, the functionality of the NVE (Network Virtualization Edge) is split across an end device supporting virtualization and an external network device which is called an external NVE. The portion of the NVE functionality located on the end device is called the tNVE (terminal-side NVE) and the portion located on the external NVE is called the nNVE (network-side NVE) in this document. Overlay encapsulation/decapsulation functions are normally off-loaded to the nNVE on the external NVE.

The tNVE is normally implemented as a part of hypervisor or container and/or virtual switch in an virtualized end device. This document uses the term "hypervisor" throughout when describing the Split-NVE scenario where part of the NVE functionality is off-loaded to a separate device from the "hypervisor" that contains a VM (Virtual Machine) connected to a VN (Virutal Network). In this context, the term "hypervisor" is meant to cover any device type where part of the NVE functionality is off-loaded in this fashion, e.g.,a Network Service Appliance or Linux Container.

The NVO3 problem statement [RFC7364], discusses the needs for a control plane protocol (or protocols) to populate each NVE with the state needed to perform the required functions. In one scenario, an NVE provides overlay encapsulation/decapsulation packet forwarding services to Tenant Systems (TSs) that are co-resident within the NVE on the same End Device (e.g. when the NVE is embedded within a hypervisor or a Network Service Appliance). In such cases, there is no need for a standardized protocol between the hypervisor and NVE, as the interaction is implemented via software on a single device. While in the Split-NVE architecture scenarios, as shown in figure 2 to figure 4, control plane protocol(s) between a hypervisor and its associated external NVE(s) are required for the hypervisor to distribute the virtual machines networking states to the NVE(s) for further handling. The protocol is an NVE-internal protocol and runs between tNVE and nNVE logical entities. This protocol is mentioned in the NVO3 problem statement [RFC7364] and appears as the third work item.

Virtual machine states and state transitioning are summarized in this document showing events where the NVE needs to take specific actions. Such events might correspond to actions the control plane signaling protocol(s) need to take between tNVE and nNVE in the Split-NVE scenario. The high level requirements to be fulfilled are stated.

```
                       +------------ Split-NVE ---------+
                       |                                |
                       |                                |
     +-----------------|-----+                          |
     | +---------------|----+|                          |
     | | +--+         \|/   ||                          |
     | | |V |TSI  +-------+ ||                   +------|-------------+
     | | |M |-----+       | ||                   |     \|/            |
     | | +--+     |       | ||                   |+--------+          |
     | | +--+     | tNVE  | ||-------------------||        |          |
     | | |V |TSI  |       | ||                   || nNVE   |          |
     | | |M |-----|       | ||                   ||        |          |
     | | +--+     +-------+ ||                   |+--------+          |
     | |                    ||                   +--------------------+
     | +-----Hypervisor-----+|
     +-----------------------+
            End Device                               External NVE
```

Figure 1 Split-NVE structure

This document uses VMs as an example of Tenant Systems (TSs) in order to describe the requirements, even though a VM is just one type of Tenant System that may connect to a VN. For example, a service instance within a Network Service Appliance is another type of TS, as are systems running on an OS-level virtualization technologies like containers. The fact that VMs have lifecycles (e.g., can be created and destroyed, can be moved, and can be started or stopped) results in a general set of protocol requirements, most of which are applicable to other forms of TSs although not all of the requirements are applicable to all forms of TSs.

Section 2 describes VM states and state transitioning in the VM's lifecycle. Section 3 introduces Hypervisor-to-NVE control plane protocol functionality derived from VM operations and network events. Section 4 outlines the requirements of the control plane protocol to achieve the required functionality.

1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document uses the same terminology as found in [RFC7365]. This section defines additional terminology used by this document.

Split-NVE: a type of NVE (Network Virtualization Edge) where the functionalities are split across an end device supporting virtualization and an external network device.

tNVE: the portion of Split-NVE functionalities located on the end device supporting virtualization. It interacts with a tenant system through an internal interface in the end device.

nNVE: the portion of Split-NVE functionalities located on the network device that is directly or indirectly connected to the end device holding the corresponding tNVE. nNVE normally performs encapsulation to and decapsulation from the overlay network.

External NVE: the physical network device holding the nNVE

Hypervisor: the logical collection of software, firmware and/or hardware that allows the creation and running of server or service appliance virtualization. tNVE is located under a Hypervisor. Hypervisor is loosely used in this document to refer to the end device supporting the virtualization. For simplicity, we also use Hypervisor to represent both hypervisor and container.

Container: Please refer to Hypervisor. For simplicity this document use the term hypervisor to represent both hypervisor and container.

VN Profile: Meta data associated with a VN (Virtual Network) that is applied to any attachment point to the VN. That is, VAP (Virtual Access Point) properties that are applied to all VAPs associated with a given VN and used by an NVE when ingressing/egressing packets to/from a specific VN. Meta data could include such information as ACLs, QoS settings, etc. The VN Profile contains parameters that apply to the VN as a whole. Control protocols between the NVE and NVA (Network Virtualization Authority) could use the VN ID or VN Name to obtain the VN Profile.

VSI: Virtual Station Interface. [IEEE 802.1Q]

VDP: VSI Discovery and Configuration Protocol [IEEE 802.1Q]

1.2 Target Scenarios

In the Split-NVE architecture, an external NVE can provide an offload of the encapsulation / decapsulation functions and network policy enforcement as well as the VN Overlay protocol overhead. This

offloading may improve performance and/or save resources in the End Device (e.g. hypervisor) using the external NVE.

The following figures give example scenarios of a Split-NVE architecture.

```
     Hypervisor             Access Switch
+------------------+       +-----+-------+
| +--+   +-------+ |       |     |       |
| |VM|---|       | | VLAN  |     |       |
| +--+   | tNVE  |---------+ nNVE|       +--- Underlying
| +--+   |       | | Trunk |     |       |    Network
| |VM|---|       | |       |     |       |
| +--+   +-------+ |       |     |       |
+------------------+       +-----+-------+
      Figure 2 Hypervisor with an External NVE
```

```
     Hypervisor      L2 Switch
+---------------+     +-----+     +----+---+
| +--+   +----+ |     |     |     |    |   |
| |VM|---|    | |VLAN |     |VLAN |    |   |
| +--+   |tNVE|-------+     +-----+nNVE|   +--- Underlying
| +--+   |    | |Trunk|     |Trunk|    |   | Network
| |VM|---|    | |     |     |     |    |   |
| +--+   +----+ |     |     |     |    |   |
+---------------+     +-----+     +----+---+
 Figure 3 Hypervisor with an External NVE
          connected through an Ethernet Access Switch
```

```
 Network Service Appliance         Access Switch
+--------------------------+      +-----+-------+
| +------------+    | \    |      |     |       |
| |Net Service |----|  \   |      |     |       |
| |Instance    |    |   \  | VLAN |     |       |
| +------------+    |tNVE| |------+nNVE |       +--- Underlying
| +------------+    |    | | Trunk|     |       |    Network
| |Net Service |----|   /  |      |     |       |
| |Instance    |    |  /   |      |     |       |
| +------------+    | /    |      |     |       |
+--------------------------+      +-----+-------+
Figure 4 Physical Network Service Appliance with an External NVE
```

Tenant Systems connect to external NVEs via a Tenant System Interface (TSI). The TSI logically connects to the external NVE via a Virtual Access Point (VAP) [RFC8014]. The external NVE may provide Layer 2 or Layer 3 forwarding. In the Split-NVE architecture, the external NVE may be able to reach multiple MAC and IP addresses via a TSI. An IP address can be in either IPv4 or IPv6 format. For example, Tenant Systems that are providing network services (such as transparent firewall, load balancer, or VPN gateway) are likely to have a complex address hierarchy. This implies that if a given TSI disassociates from one VN, all the MAC and/or IP addresses are also disassociated. There is no need to signal the deletion of every MAC or IP when the TSI is brought down or deleted. In the majority of cases, a VM will be acting as a simple host that will have a single TSI and single MAC and IP visible to the external NVE.

Figures 2 through 4 show the use of VLANs to separate traffic for multiple VNs between the tNVE and nNVE; VLANs are not strictly necessary if only one VN is involved, but multiple VNs are expected in most cases. Hence this draft assumes the presence of VLANs.

2. VM Lifecycle

Figure 2 of [RFC7666] shows the state transition of a VM. Some of the VM states are of interest to the external NVE. This section illustrates the relevant phases and events in the VM lifecycle. Note that the following subsections do not give an exhaustive traversal of VM lifecycle state. They are intended as the illustrative examples which are relevant to Split-NVE architecture, not as prescriptive text; the goal is to capture sufficient detail to set a context for the signaling protocol functionality and requirements described in the following sections.

2.1 VM Creation Event

The VM creation event causes the VM state transition from Preparing to Shutdown and then to Running [RFC7666]. The end device allocates and initializes local virtual resources like storage in the VM Preparing state. In the Shutdown state, the VM has everything ready except that CPU execution is not scheduled by the hypervisor and VM's memory is not resident in the hypervisor. The transition from the Shutdown state to the Running state normally requires human action or a system triggered event. Running state indicates the VM is in the normal execution state. As part of transitioning the VM to the Running state, the hypervisor must also provision network connectivity for the VM's TSI(s) so that Ethernet frames can be sent and received correctly. Initially, when Running, no ongoing

migration, suspension or shutdown is in process.

In the VM creation phase, the VM's TSI has to be associated with the external NVE. Association here indicates that hypervisor and the external NVE have signaled each other and reached some agreement. Relevant networking parameters or information have been provisioned properly. The External NVE should be informed of the VM's TSI MAC address and/or IP address. In addition to external network connectivity, the hypervisor may provide local network connectivity between the VM's TSI and other VM's TSI that are co-resident on the same hypervisor. When the intra- or inter-hypervisor connectivity is extended to the external NVE, a locally significant tag, e.g. VLAN ID, should be used between the hypervisor and the external NVE to differentiate each VN's traffic. Both the hypervisor and external NVE sides must agree on that tag value for traffic identification, isolation, and forwarding.

The external NVE may need to do some preparation before it signals successful association with the TSI. Such preparation may include locally saving the states and binding information of the tenant system interface and its VN, communicating with the NVA for network provisioning, etc.

Tenant System interface association should be performed before the VM enters the Running state, preferably in the Shutdown state. If association with an external NVE fails, the VM should not go into the Running state.

2.2 VM Live Migration Event

Live migration is sometimes referred to as "hot" migration in that, from an external viewpoint, the VM appears to continue to run while being migrated to another server (e.g., TCP connections generally survive this class of migration). In contrast, "cold" migration consists of shutting down VM execution on one server and restarting it on another. For simplicity, the following abstract summary of live migration assumes shared storage, so that the VM's storage is accessible to the source and destination servers. Assume VM live migrates from hypervisor 1 to hypervisor 2. Such a migration event involves state transitions on both source hypervisor 1 and destination hypervisor 2. The VM state on source hypervisor 1 transits from Running to Migrating and then to Shutdown [RFC7666]. The VM state on destination hypervisor 2 transits from Shutdown to Migrating and then Running.

The external NVE connected to destination hypervisor 2 has to associate the migrating VM's TSI with it by discovering the TSI's MAC

and/or IP addresses, its VN, locally significant VLAN ID if any, and provisioning other network related parameters of the TSI. The external NVE may be informed about the VM's peer VMs, storage devices and other network appliances with which the VM needs to communicate or is communicating. The migrated VM on destination hypervisor 2 should not go to Running state until all the network provisioning and binding has been done.

The states of VM on the source and destination hypervisors both are Migrating during transfer of migration execution. The migrating VM should not be in Running state at the same time on the source hypervisor and destination hypervisor during migration. The VM on the source hypervisor does not transition into Shutdown state until the VM successfully enters the Running state on the destination hypervisor. It is possible that the VM on the source hypervisor stays in Migrating state for a while after the VM on the destination hypervisor enters Running state.

2.3 VM Termination Event

A VM termination event is also referred to as "powering off" a VM. A VM termination event leads to its state becoming Shutdown. There are two possible causes of VM termination [RFC7666]. One is the normal "power off" of a running VM; the other is that the VM has been migrated to another hypervisor and the VM image on the source hypervisor has to stop executing and be shutdown.

In VM termination, the external NVE connecting to that VM needs to deprovision the VM, i.e. delete the network parameters associated with that VM. In other words, the external NVE has to de-associate the VM's TSI.

2.4 VM Pause, Suspension and Resumption Events

A VM pause event leads to the VM transiting from Running state to Paused state. The Paused state indicates that the VM is resident in memory but no longer scheduled to execute by the hypervisor [RFC7666]. The VM can be easily re-activated from Paused state to Running state.

A VM suspension event leads to the VM transiting from Running state to Suspended state. A VM resumption event leads to the VM transiting state from Suspended state to Running state. Suspended state means the memory and CPU execution state of the virtual machine are saved to persistent store. During this state, the virtual machine is not scheduled to execute by the hypervisor [RFC7666].

In the Split-NVE architecture, the external NVE should not

disassociate the paused or suspended VM as the VM can return to Running state at any time.

3. Hypervisor-to-NVE Control Plane Protocol Functionality

The following subsections show illustrative examples of the state transitions of an external NVE which are relevant to Hypervisor-to-NVE Signaling protocol functionality. It should be noted this is not prescriptive text for the full state machine.

3.1 VN Connect and Disconnect

In the Split-NVE scenario, a protocol is needed between the End Device (e.g. Hypervisor) and the external NVE it is using in order to make the external NVE aware of the changing VN membership requirements of the Tenant Systems within the End Device.

A key driver for using a protocol rather than using static configuration of the external NVE is because the VN connectivity requirements can change frequently as VMs are brought up, moved, and brought down on various hypervisors throughout the data center or external cloud.

```
+---------------+   Receive VN_connect;     +-------------------+
|VN_Disconnected|   return Local_Tag value  |VN_Connected       |
+---------------+   for VN if successful;   +-------------------+
|VN_ID;         |-------------------------->|VN_ID;             |
|VN_State=      |                           |VN_State=connected;|
|disconnected;  |                           |Num_TSI_Associated;|
|               |<--Receive VN_disconnect---|Local_Tag;         |
+---------------+                           |VN_Context;        |
                                            +-------------------+
```

Figure 5. State Transition Example of a VAP Instance
on an External NVE

Figure 5 shows the state transition for a VAP on the external NVE. An NVE that supports the hypervisor to NVE control plane protocol should support one instance of the state machine for each active VN. The state transition on the external NVE is normally triggered by the hypervisor-facing side events and behaviors. Some of the interleaved interaction between NVE and NVA will be illustrated to better explain the whole procedure; while others of them may not be shown.

The external NVE must be notified when an End Device requires connection to a particular VN and when it no longer requires connection. Connection clean up for the failed devices should be employed which is out of the scope of the protocol specified in this document.

In addition, the external NVE should provide a local tag value for each connected VN to the End Device to use for exchanging packets between the End Device and the external NVE (e.g. a locally significant [IEEE 802.1Q] tag value). How "local" the significance is depends on whether the Hypervisor has a direct physical connection to the external NVE (in which case the significance is local to the physical link), or whether there is an Ethernet switch (e.g. a blade switch) connecting the Hypervisor to the NVE (in which case the significance is local to the intervening switch and all the links connected to it).

These VLAN tags are used to differentiate between different VNs as packets cross the shared access network to the external NVE. When the external NVE receives packets, it uses the VLAN tag to identify their VN coming from a given TSI, strips the tag, adds the appropriate overlay encapsulation for that VN, and sends it towards the corresponding remote NVE across the underlying IP network.

The Identification of the VN in this protocol could either be through a VN Name or a VN ID. A globally unique VN Name facilitates portability of a Tenant's Virtual Data Center. Once an external NVE receives a VN connect indication, the NVE needs a way to get a VN Context allocated (or receive the already allocated VN Context) for a given VN Name or ID (as well as any other information needed to transmit encapsulated packets). How this is done is the subject of the NVE-to-NVA protocol which are part of work items 1 and 2 in [RFC7364]. The external NVE needs to synchronize the mapping information of the local tag and VN Name or VN ID with NVA.

The VN_connect message can be explicit or implicit. Explicit means the hypervisor sends a request message explicitly for the connection to a VN. Implicit means the external NVE receives other messages, e.g. very first TSI associate message (see the next subsection) for a given VN, that implicitly indicate its interest in connecting to a VN.

A VN_disconnect message indicates that the NVE can release all the resources for that disconnected VN and transit to VN_disconnected state. The local tag assigned for that VN can possibly be reclaimed for use by another VN.

3.2 TSI Associate and Activate

Typically, a TSI is assigned a single MAC address and all frames transmitted and received on that TSI use that single MAC address. As mentioned earlier, it is also possible for a Tenant System to exchange frames using multiple MAC addresses or packets with multiple IP addresses.

Particularly in the case of a TS that is forwarding frames or packets from other TSs, the external NVE will need to communicate the mapping between the NVE's IP address on the underlying network and ALL the addresses the TS is forwarding on behalf of the corresponding VN to the NVA.

The NVE has two ways it can discover the tenant addresses for which frames are to be forwarded to a given End Device (and ultimately to the TS within that End Device).

1. It can glean the addresses by inspecting the source addresses in packets it receives from the End Device.

2. The hypervisor can explicitly signal the address associations of a TSI to the external NVE. An address association includes all the MAC and/or IP addresses possibly used as source addresses in a packet sent from the hypervisor to external NVE. The external NVE may further use this information to filter the future traffic from the hypervisor.

To use the second approach above, the "hypervisor-to-NVE" protocol must support End Devices communicating new tenant addresses associations for a given TSI within a given VN.

Figure 6 shows the example of a state transition for a TSI connecting to a VAP on the external NVE. An NVE that supports the hypervisor to NVE control plane protocol may support one instance of the state machine for each TSI connecting to a given VN.

```
                 disassociate   +--------+     disassociate
               +--------------->|  Init  |<--------------------+
               |                +--------+                     |
               |                |        |                     |
               |                |        |                     |
               |                +--------+                     |
               |                  |    |                       |
               |      associate   |    |  activate             |
               |      +-----------+    +-----------+           |
               |      |                            |           |
               |      |                            |           |
               |     \|/                          \|/          |
       +--------------------+                  +---------------------+
       |     Associated     |                  |       Activated     |
       +--------------------+                  +---------------------+
       |TSI_ID;             |                  |TSI_ID;              |
       |Port;               |-----activate---->|Port;                |
       |VN_ID;              |                  |VN_ID;               |
       |State=associated;   |                  |State=activated ;    |-+
     +-|Num_Of_Addr;        |<---deactivate ---|Num_Of_Addr;         | |
     | |List_Of_Addr;       |                  |List_Of_Addr;        | |
     | +--------------------+                  +---------------------+ |
     |                    /|\                     /|\                  |
     |                     |                       |                   |
     +---------------------+                       +-------------------+
      add/remove/updt addr;                        add/remove/updt addr;
      or update port;                              or update port;
```

Figure 6 State Transition Example of a TSI Instance
on an External NVE

The Associated state of a TSI instance on an external NVE indicates all the addresses for that TSI have already associated with the VAP of the external NVE on a given port e.g. on port p for a given VN but no real traffic to and from the TSI is expected and allowed to pass through. An NVE has reserved all the necessary resources for that TSI. An external NVE may report the mappings of its underlay IP address and the associated TSI addresses to NVA and relevant network nodes may save such information to their mapping tables but not their forwarding tables. An NVE may create ACL or filter rules based on the associated TSI addresses on that attached port p but not enable them yet. The local tag for the VN corresponding to the TSI instance should be provisioned on port p to receive packets.

The VM migration event (discussed section 2) may cause the hypervisor to send an associate message to the NVE connected to the destination hypervisor of the migration. A VM creation event may also cause to

the same practice.

The Activated state of a TSI instance on an external NVE indicates that all the addresses for that TSI are functioning correctly on a given port e.g. port p and traffic can be received from and sent to that TSI via the NVE. The mappings of the NVE's underlay IP address and the associated TSI addresses should be put into the forwarding table rather than the mapping table on relevant network nodes. ACL or filter rules based on the associated TSI addresses on the attached port p in the NVE are enabled. The local tag for the VN corresponding to the TSI instance must be provisioned on port p to receive packets.

The Activate message makes the state transit from Init or Associated to Activated. VM creation, VM migration, and VM resumption events discussed in Section 4 may trigger sending the Activate message from the hypervisor to the external NVE.

TSI information may get updated in either the Associated or Activated state. The following are considered updates to the TSI information: add or remove the associated addresses, update the current associated addresses (for example updating IP for a given MAC), and update the NVE port information based on where the NVE receives messages. Such updates do not change the state of TSI. When any address associated with a given TSI changes, the NVE should inform the NVA to update the mapping information for NVE's underlying address and the associated TSI addresses. The NVE should also change its local ACL or filter settings accordingly for the relevant addresses. Port information updates will cause the provisioning of the local tag for the VN corresponding to the TSI instance on new port and removal from the old port.

3.3 TSI Disassociate and Deactivate

Disassociate and deactivate behaviors are conceptually the reverse of associate and activate.

From Activated state to Associated state, the external NVE needs to make sure the resources are still reserved but the addresses associated to the TSI are not functioning. No traffic to or from the TSI is expected or allowed to pass through. For example, the NVE needs to tell the NVA to remove the relevant addresses mapping information from forwarding and routing tables. ACL and filtering rules regarding the relevant addresses should be disabled.

From Associated or Activated state to the Init state, the NVE releases all the resources relevant to TSI instances. The NVE should also inform the NVA to remove the relevant entries from mapping table. ACL or filtering rules regarding the relevant addresses should

be removed. Local tag provisioning on the connecting port on NVE should be cleared.

A VM suspension event (discussed in section 2) may cause the relevant TSI instance(s) on the NVE to transit from Activated to Associated state.

A VM pause event normally does not affect the state of the relevant TSI instance(s) on the NVE as the VM is expected to run again soon.

A VM shutdown event will normally cause the relevant TSI instance(s) on the NVE to transition to Init state from Activated state. All resources should be released.

A VM migration will cause the TSI instance on the source NVE to leave Activated state. When a VM migrates to another hypervisor connecting to the same NVE, i.e. source and destination NVE are the same, NVE should use TSI_ID and incoming port to differentiate two TSI instances.

Although the triggering messages for the state transition shown in Figure 6 does not indicate the difference between a VM creation/shutdown event and a VM migration arrival/departure event, the external NVE can make optimizations if it is given such information. For example, if the NVE knows the incoming activate message is caused by migration rather than VM creation, some mechanisms may be employed or triggered to make sure the dynamic configurations or provisionings on the destination NVE are the same as those on the source NVE for the migrated VM. For example an IGMP query [RFC2236] can be triggered by the destination external NVE to the migrated VM so that VM is forced to send an IGMP report to the multicast router. Then a multicast router can correctly route the multicast traffic to the new external NVE for those multicast groups the VM joined before the migration.

4. Hypervisor-to-NVE Control Plane Protocol Requirements

Req-1: The protocol MUST support a bridged network connecting End Devices to the External NVE.

Req-2: The protocol MUST support multiple End Devices sharing the same External NVE via the same physical port across a bridged network.

Req-3: The protocol MAY support an End Device using multiple external NVEs simultaneously, but only one external NVE for each VN.

Req-4: The protocol MAY support an End Device using multiple external NVEs simultaneously for the same VN.

Req-5: The protocol MUST allow the End Device to initiate a request to its associated External NVE to be connected/disconnected to a given VN.

Req-6: The protocol MUST allow an External NVE initiating a request to its connected End Devices to be disconnected from a given VN.

Req-7: When a TS attaches to a VN, the protocol MUST allow for an End Device and its external NVE to negotiate one or more locally-significant tag(s) for carrying traffic associated with a specific VN (e.g., [IEEE 802.1Q] tags).

Req-8: The protocol MUST allow an End Device initiating a request to associate/disassociate and/or activate/deactive some or all address(es) of a TSI instance to a VN on an NVE port.

Req-9: The protocol MUST allow the External NVE initiating a request to disassociate and/or deactivate some or all address(es) of a TSI instance to a VN on an NVE port.

Req-10: The protocol MUST allow an End Device initiating a request to add, remove or update address(es) associated with a TSI instance on the external NVE. Addresses can be expressed in different formats, for example, MAC, IP or pair of IP and MAC.

Req-11: The protocol MUST allow the External NVE and the connected End Device to authenticate each other.

Req-12: The protocol MUST be able to run over L2 links between the End Device and its External NVE.

Req-13: The protocol SHOULD support the End Device indicating if an associate or activate request from it is the result of a VM hot migration event.

5. VDP Applicability and Enhancement Needs

Virtual Station Interface (VSI) Discovery and Configuration Protocol (VDP) [IEEE 802.1Q] can be the control plane protocol running between the hypervisor and the external NVE. Appendix A illustrates VDP for the reader's information.

VDP facilitates the automatic discovery and configuration of Edge

Virtual Bridging (EVB) stations and Edge Virtual Bridging (EVB) bridges. An EVB station is normally an end station running multiple VMs. It is conceptually equivalent to a hypervisor in this document. An EVB bridge is conceptually equivalent to the external NVE.

VDP is able to pre-associate/associate/de-associate a VSI on an EVB station with a port on the EVB bridge. A VSI is approximately the concept of a virtual port by which a VM connects to the hypervisor in this document's context. The EVB station and the EVB bridge can reach agreement on VLAN ID(s) assigned to a VSI via VDP message exchange. Other configuration parameters can be exchanged via VDP as well. VDP is carried over the Edge Control Protocol(ECP) [IEEE 802.1Q] which provides a reliable transportation over a layer 2 network.

VDP protocol needs some extensions to fulfill the requirements listed in this document. Table 1 shows the needed extensions and/or clarifications in the NVO3 context.

Req	Supported by VDP?	remarks	
Req-1 Req-2 Req-3 Req-4	Partially	Needs extension. Must be able to send to a specific unicast MAC and should be able to send to a non-reserved well known multicast address other than the nearest customer bridge address.	
Req-5	Yes	VN is indicated by GroupID	
Req-6	Yes	Bridge sends De-Associate	
Req-7	Yes	VID==NULL in request and bridge returns the assigned value in response or specify GroupID in request and get VID assigned in returning response. Multiple VLANs per group are allowed.	
Req-8	Partially	requirements	VDP equivalence
		associate/disassociate	pre-asso/de-associate
		activate/deactivate	associate/de-associate
		Needs extension to allow associate->pre-assoc	

Req-9	Yes	VDP bridge initiates de-associate
Req-10	Partially	Needs extension for IPv4/IPv6 address. Add a new "filter info format" type.
Req-11	No	Out-of-band mechanism is preferred, e.g. MACSec or 802.1X. Implicit authentication based on control of physical connectivity exists in VDP when the External NVE connects to the End Device directly and is reachable with the nearest customer bridge address.
Req-12	Yes	L2 protocol naturally
Req-13	Partially	M bit for migrated VM on destination hypervisor and S bit for that on source hypervisor. It is indistinguishable when M/S is 0 between no guidance and events not caused by migration where NVE may act differently. Needs new New bits for migration indication in new "filter info format" type.

Table 1 Compare VDP with the requirements

Simply adding the ability to carry layer 3 addresses, VDP can serve the Hypervisor-to-NVE control plane functions pretty well. Other extensions are the improvement of the protocol capabilities for better fit in an NVO3 network.

6. Security Considerations

External NVEs must ensure that only properly authorized Tenant Systems are allowed to join and become a part of any particular Virtual Network. In some cases, tNVE may want to connect to the nNVE for provisioning purposes. This may require that the tNVE authenticate the nNVE in addition to the nNVE authenticating the tNVE. If a secure channel is required between tNVE and nNVE to carry encrypted split-NVE control plane protocol, then existing mechanisms such as MACsec [IEEE 802.1AE] can be used. In some deployments, authentication may be implicit based on control of physical connectivity, e.g., if the nNVE is located in the bridge that is directly connected to the server that contains the tNVE. Use of "nearest customer bridge address" in VDP [IEEE 802.1Q] is an example where this sort of implicit authentication is possible, although explicit authentication also applies in that case.

As the control plane protocol results in configuration changes for

both the tNVE and nNVE, tNVE and nNVE implementations should log all state changes, including those described in Section 3. Implementations should also log significant protocol events, such as establishment or loss of control plane protocol connectivity between the tNVE and nNVE and authentication results.

In addition, external NVEs will need appropriate mechanisms to ensure that any hypervisor wishing to use the services of an NVE is properly authorized to do so. One design point is whether the hypervisor should supply the external NVE with necessary information (e.g., VM addresses, VN information, or other parameters) that the external NVE uses directly, or whether the hypervisor should only supply a VN ID and an identifier for the associated VM (e.g., its MAC address), with the external NVE using that information to obtain the information needed to validate the hypervisor-provided parameters or obtain related parameters in a secure manner. The former approach can be used in a trusted environment so that the external NVE can directly use all the information retrieved from the hypervisor for local configuration. It saves the effort on the external NVE side from information retrieval and/or validation. The latter approach gives more reliable information as the external NVE needs to retrieve them from some management system database. Especially some network related parameters like VLAN IDs can be passed back to hypervisor to be used as a more authoritative provisioning. However in certain cases, it is difficult or inefficient for an external NVE to have access or query on some information to those management systems. Then the external NVE has to obtain those information from hypervisor.

7. IANA Considerations

No IANA action is required.

8. Acknowledgements

This document was initiated based on the merger of the drafts draft-kreeger-nvo3-hypervisor-nve-cp, draft-gu-nvo3-tes-nve-mechanism, and draft-kompella-nvo3-server2nve. Thanks to all the co-authors and contributing members of those drafts.

The authors would like to specially thank Lucy Yong and Jon Hudson for their generous help in improving this document.

8. References

8.1 Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate

Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC7365] Lasserre, M., Balus, F., Morin, T., Bitar, N., and Y. Rekhter, "Framework for DC Network Virtualization", October 2014.

[RFC7666] Asai H., MacFaden M., Schoenwaelder J., Shima K., Tsou T., "Management Information Base for Virtual Machines Controlled by a Hypervisor", October 2015.

[RFC8014] Black, D., Hudson, J., Kreeger, L., Lasserre, M., Narten, T., "An Architecture for Data-Center Network Virtualization over Layer 3 (NVO3)", December 2016.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words ", BCP 14, RFC 8174, May 2017.

[IEEE 802.1Q] IEEE, "Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks", IEEE Std 802.1Q-2014, November 2014.

8.2 Informative References

[RFC2236] Fenner, W., "Internet Group Management Protocol, Version 2", RFC 2236, November 1997.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, July 2005.

[RFC7364] Narten, T., Gray, E., Black, D., Fang, L., Kreeger, L., and M. Napierala, "Problem Statement: Overlays for Network Virtualization", October 2014.

[IEEE 802.1AE] IEEE, "MAC Security (MACsec)", IEEE Std 802.1AE-2006, August 2006.

Appendix A. IEEE 802.1Q VDP Illustration (For information only)

The VDP (VSI Discovery and Discovery and Configuration Protocol, clause 41 of [IEEE 802.1Q]) can be considered as a controlling protocol running between the hypervisor and the external bridge. VDP association TLV structure are formatted as shown in Figure A.1.

```
+--------+--------+------+-----+--------+------+------+------+------+
|TLV type|TLV info|Status|VSI  |VSI Type|VSI ID|VSI ID|Filter|Filter|
|        |string  |      |Type |Version |Format|      |Info  |Info  |
|        |length  |      |ID   |        |      |      |format|      |
+--------+--------+------+-----+--------+------+------+------+------+
|                 |      |<----VSI type&instance----->|<--Filter--->|
|                 |      |<-------------VSI attributes------------->|
|<--TLV header--->|<-----------TLV information string ------------->|
```

Figure A.1: VDP association TLV

There are basically four TLV types.

1. Pre-associate: Pre-associate is used to pre-associate a VSI instance with a bridge port. The bridge validates the request and returns a failure Status in case of errors. Successful pre-associate does not imply that the indicated VSI Type or provisioning will be applied to any traffic flowing through the VSI. The pre-associate enables faster response to an associate, by allowing the bridge to obtain the VSI Type prior to an association.

2. Pre-associate with resource reservation: Pre-associate with Resource Reservation involves the same steps as Pre-associate, but on success it also reserves resources in the bridge to prepare for a subsequent Associate request.

3. Associate: Associate creates and activates an association between a VSI instance and a bridge port. An bridge allocates any required bridge resources for the referenced VSI. The bridge activates the configuration for the VSI Type ID. This association is then applied to the traffic flow to/from the VSI instance.

4. De-associate: The de-associate is used to remove an association between a VSI instance and a bridge port. Pre-associated and associated VSIs can be de-associated. De-associate releases any resources that were reserved as a result of prior associate or pre-Associate operations for that VSI instance.

De-associate can be initiated by either side and the other types can only be initiated by the server side.

Some important flag values in VDP Status field:

1. M-bit (Bit 5): Indicates that the user of the VSI (e.g., the VM) is migrating (M-bit = 1) or provides no guidance on the migration of the user of the VSI (M-bit = 0). The M-bit is used as an indicator relative to the VSI that the user is migrating to.

2. S-bit (Bit 6): Indicates that the VSI user (e.g., the VM) is suspended (S-bit = 1) or provides no guidance as to whether the user of the VSI is suspended (S-bit = 0). A keep-alive Associate request with S-bit = 1 can be sent when the VSI user is suspended. The S-bit is used as an indicator relative to the VSI that the user is migrating from.

The filter information format currently defines 4 types. Each of the filter information is shown in details as follows.

1. VID Filter Info format

```
+---------+------+-------+--------+
| #of     | PS   | PCP   | VID    |
|entries  |(1bit)|(3bits)|(12bits)|
|(2octets)|      |       |        |
+---------+------+-------+--------+
          |<--Repeated per entry->|
```

Figure A.2 VID Filter Info format

2. MAC/VID Filter Info format

```
+---------+--------------+------+-------+--------+
| #of     |  MAC address | PS   | PCP   | VID    |
|entries  |  (6 octets)  |(1bit)|(3bits)|(12bits)|
|(2octets)|              |      |       |        |
+---------+--------------+------+-------+--------+
          |<--------Repeated per entry---------->|
```

Figure A.3 MAC/VID filter format

3. GroupID/VID Filter Info format

```
+---------+--------------+------+-------+--------+
| #of     |  GroupID     | PS   | PCP   | VID    |
|entries  |  (4 octets)  |(1bit)|(3bits)|(12bits)|
|(2octets)|              |      |       |        |
+---------+--------------+------+-------+--------+
          |<--------Repeated per entry---------->|
```

Figure A.4 GroupID/VID filter format

4. GroupID/MAC/VID Filter Info format

```
+---------+----------+------------+------+-----+--------+
| #of     | GroupID  | MAC address | PS   | PCP | VID    |
|entries  |(4 octets)| (6 octets)  |(1bit)|(3b )|(12bits)|
|(2octets)|          |             |      |     |        |
+---------+----------+------------+------+-----+--------+
          |<-------------Repeated per entry------------->|
```

Figure A.5 GroupID/MAC/VID filter format

The null VID can be used in the VDP Request sent from the station to the external bridge. Use of the null VID indicates that the set of VID values associated with the VSI is expected to be supplied by the bridge. The set of VID values is returned to the station via the VDP Response. The returned VID value can be a locally significant value. When GroupID is used, it is equivalent to the VN ID in NVO3. GroupID will be provided by the station to the bridge. The bridge maps GroupID to a locally significant VLAN ID.

The VSI ID in VDP association TLV that identify a VM can be one of the following format: IPV4 address, IPV6 address, MAC address, UUID [RFC4122], or locally defined.

Authors' Addresses

Yizhou Li
Huawei Technologies
101 Software Avenue,
Nanjing 210012
China

Phone: +86-25-56625409
EMail: liyizhou@huawei.com

Donald Eastlake
Huawei R&D USA
155 Beaver Street
Milford, MA 01757 USA

Phone: +1-508-333-2270
EMail: d3e3e3@gmail.com

Lawrence Kreeger
Arrcus, Inc

Email: lkreeger@gmail.com

Thomas Narten
IBM

Email: narten@us.ibm.com

David Black
Dell EMC
176 South Street,
Hopkinton, MA 01748 USA

Email: david.black@dell.com