

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: August 29, 2013

L. Kreeger
Cisco
T. Narten
IBM
D. Black
EMC
February 25, 2013

Network Virtualization Hypervisor-to-NVE Overlay Control Protocol
Requirements
draft-kreeger-nvo3-hypervisor-nve-cp-01

Abstract

The document "Problem Statement: Overlays for Network Virtualization" discusses the needs for network virtualization using overlay networks in highly virtualized data centers. The problem statement outlines a need for control protocols to facilitate running these overlay networks. This document outlines the high level requirements related to the interaction between hypervisors and the Network Virtualization Edge device when the two entities are not co-located on the same physical device.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 29, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents

Internet-Draft NV03 Hypervisor-NVE Control Protocol Reqs February 2013

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
2.	Terminology	4
3.	Entity Relationships	6
3.1.	VNIC Containment Relationship	6
3.1.1.	Layer 2 Virtual Network Service	7
3.1.2.	Layer 3 Virtual Network Service	8
4.	Hypervisor-to-NVE Control Plane Protocol Functionality	9
4.1.	VN Connect/Disconnect	11
4.2.	VNIC Address Association	12
4.3.	VNIC Address Disassociation	12
4.4.	VNIC Shutdown/Startup/Migration	13
4.5.	VN Profile	14
5.	Security Considerations	14
6.	Acknowledgements	14
7.	Informative References	14
	Authors' Addresses	15

Internet-Draft NV03 Hypervisor-NVE Control Protocol Reqs February 2013

1. Introduction

Note: the contents of this document were originally in [\[I-D.kreeger-nvo3-overlay-cp\]](#). The content has been pulled into its own document because the problem area covered is distinct and different from what most folk think of as a "control protocol" for NV03. Other related documents on this same general topic include [\[I-D.kompella-nvo3-server2nve\]](#), [\[I-D.gu-nvo3-overlay-cp-arch\]](#), and [\[I-D.gu-nvo3-tes-nve-mechanism\]](#).

"Problem Statement: Overlays for Network Virtualization" [\[I-D.ietf-nvo3-overlay-problem-statement\]](#) discusses the needs for network virtualization using overlay networks in highly virtualized data centers and provides a general motivation for building such networks. "Framework for DC Network Virtualization" [\[I-D.ietf-nvo3-framework\]](#) provides a framework for discussing overlay networks generally and the various components that must work together in building such systems. The reader is assumed to be familiar with both documents.

Section 4.5 of [\[I-D.ietf-nvo3-overlay-problem-statement\]](#) describes three separate work areas that fall under the general category of a control protocol for NV03. This document focuses entirely on the control protocol related to the hypervisor-to-NVE interaction, labeled as the "third work item" in [\[I-D.ietf-nvo3-overlay-problem-statement\]](#). Requirements for the interaction between an NVE and the "oracle" are described in [\[I-D.kreeger-nvo3-overlay-cp\]](#).

The NV03 WG needs to decide on a better term for "oracle". This document will use Information Mapping Authority (IMA) until a decision is made.

This document uses the term "hypervisor" throughout when describing the scenario where NVE functionality is implemented on a separate device from the "hypervisor" that contains a VM connected to a VN.

In this context, the term "hypervisor" is meant to cover any device type where the NVE functionality is offloaded in this fashion, e.g., a Network Service Appliance.

This document often uses the term "VM" and "Tenant System" (TS) interchangeably, 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 may be another type of TS. When this document uses the term VM, it will in most cases apply to other types of TSs.

[2.](#) Terminology

This document uses the same terminology as found in the NV03 Framework document, [[I-D.ietf-nvo3-framework](#)]. Some of the terms defined in the Framework document have been repeated in this section for the convenience of the reader, along with additional terminology that is used by this document.

IMA: Information Mapping Authority.

[[I-D.ietf-nvo3-overlay-problem-statement](#)] uses the term "oracle" to describe this. It is a back-end system that is responsible for distributing and maintaining the mapping information for the entire overlay system. Note that the WG never reached consensus on what to call this architectural entity within the overlay system, so this term is subject to change.

Tenant System: A physical or virtual system that can play the role of a host, or a forwarding element such as a router, switch, firewall, etc. It belongs to a single tenant and connects to one or more VNs of that tenant.

End Device: A physical system to which networking service is provided. Examples include hosts (e.g. server or server blade), storage systems (e.g., file servers, iSCSI storage systems), and network devices (e.g., firewall, load-balancer, IPSec gateway). An end device may include internal networking functionality that interconnects the device's components (e.g. virtual switches that interconnect VMs running on the same server). NVE functionality may be implemented as part of that internal networking.

Network Service Appliance: A stand-alone physical device or a virtual device that provides a network service, such as a firewall, load balancer, etc. Such appliances may embed Network Virtualization Edge (NVE) functionality within them in order to more efficiently operate as part of a virtualized network.

VN: Virtual Network. This is a virtual L2 or L3 domain that belongs to a tenant.

VDC: Virtual Data Center. A container for virtualized compute, storage and network services. Managed by a single tenant, a VDC can contain multiple VNs and multiple Tenant Systems that are connected to one or more of these VNs.

VN Alias: A string name for a VN as used by administrators and customers to name a specific VN. A VN Alias is a human-usable string that can be listed in contracts, customer forms, email, configuration files, etc. and that can be communicated easily

vocally (e.g., over the phone). A VN Name is independent of the underlying technology used to implement a VN and will generally not be carried in protocol fields of control protocols used in virtual networks. Rather, a VN Alias will be mapped into a VN Name where precision is required.

VN Name: A globally unique identifier for a VN suitable for use within network protocols. A VN Name will usually be paired with a VN Alias, with the VN Alias used by humans as a shorthand way to name and identify a specific VN. A VN Name should have a compact representation to minimize protocol overhead where a VN Name is carried in a protocol field. Using a Universally Unique Identifier (UUID) as discussed in [RFC 4122](#), may work well because it is both compact and a fixed size and can be generated locally with a very high likelihood of global uniqueness.

VN ID: A unique and compact identifier for a VN within the scope of a specific NV03 administrative domain. It will generally be more efficient to carry VN IDs as fields in control protocols than VN Aliases. There is a one-to-one mapping between a VN Name and a VN ID within an NV03 Administrative Domain. Depending on the technology used to implement an overlay network, the VN ID could

be used as the Context Identifier in the data plane, or would need to be mapped to a locally-significant Context Identifier.

VN Profile: Meta data associated with a VN that is 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 could use the VN ID or VN Name to obtain the VN Profile.

VNIC: A Virtual NIC that connects a Tenant System to a Virtual Network Instance (VNI). Virtual NICs have virtual MAC addresses that may not be globally unique, but must be unique within a VN for proper network operation.

VNIC Name: A globally unique identifier for a VNIC suitable for use within network protocols. Note that because VNIC MAC addresses may not be globally unique, they cannot be used as the VNIC Name. A VNIC Name should have a compact representation to minimize protocol overhead where a VNIC Name is carried in a protocol field. Using a Universally Unique Identifier (UUID) as discussed in [RFC 4122](#), may work well because it is both compact and a fixed size and can be generated locally with a very high likelihood of global uniqueness.

[3.](#) Entity Relationships

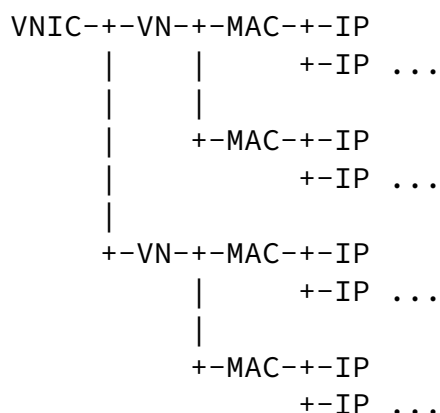
This section describes the relationships between the entities involved in the Hypervisor-to-NVE control protocol.

[3.1.](#) VNIC Containment Relationship

The root of the containment tree is a VNIC. Even though a VM may have multiple VNICs, from the point of view of an NVE, each VNIC can be treated independently. There is no need to identify the VM itself within the Hypervisor-to-NVE protocol.

Each VNIC can connect to multiple VNs. Within each VNIC-VN pair, multiple MAC addresses may be reachable. Within each VNIC-VN-MAC triplet, there may be multiple IP addresses. This containment

hierarchy is depicted below.



VNIC Containment Relationship

Figure 1

Any of these entities can be added or removed dynamically at any time.

The relationship implies that if one entity in the hierarchy is deleted then all the entities it contains are also deleted. For example, if a given VNIC disassociates from one VN, all the MAC and IP addresses are also disassociated. There is no need to signal the deletion of every entity within a VNIC when the VNIC is brought down or deleted (or the VM it is attached to is powered off or migrates away from the hypervisor).

If a VNIC provides connectivity to a range of IP addresses (e.g. the VM is a load balancer with many Virtual IP addresses), it will be more efficient to signal a range or address mask in place of

individual IP addresses.

In the majority of cases, a VM will be acting as a simple host that will have the following containment tree:

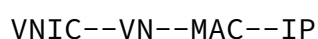


Figure 2

Since this is the most common case, the Hypervisor-to-NVE protocol should be optimized to handle this case.

Tenant Systems (TS) that are providing network services (such as firewall, load balancer, VPN gateway) are likely to have a more complex containment hierarchy. For example, a TS acting as a load balancer is quite likely to terminate multiple IP addresses, one for each application, or farm of servers that it is providing the front end for.

Hypervisors often have a limit on the number of VNICs that a VM can have (e.g. in the range of 8 to 10 VNICs). If a VM has the need to connect to more networks than the number of VNICs the hypervisor supports, the solution is often to configure the VNIC (and the associated virtual port on the virtual switch the VNIC connects to) as an 802.1Q trunk. In the case of a virtual switch that supports only VLANs, the VLAN tags used by all the VNICs connected to the switch (as well as the bridged network the hypervisor is physically connected to) share a common VLAN ID.

In a multi-tenant scenario using overlay Virtual Networks instead of VLANs, VNICs can still use 802.1Q tagging to isolate traffic from different VNs as it crosses the virtual link between the VNIC and the virtual switch; However, The tags would have only local significance across that virtual link, with the virtual switch mapping each tag value to a different VN. This implies that two different virtual links may use different 802.1Q tag values but with each mapped to the same VN by the virtual switch. Similarly, two VNICs could use the same VLAN tag value but the virtual switch can map each vPort/Tag pair to a different VN.

Each VNIC must attach to at least one VN and have at minimum one MAC address. An IP address can be optional depending on whether the VN is providing L2 or L3 service.

[3.1.1.](#) Layer 2 Virtual Network Service

When the Virtual Network is providing only Layer 2 forwarding, the NVEs only require knowledge of the Tenant System's MAC addresses,

while layer 3 termination and routing happens only in the Tenant

Systems.

For example, if a VM is acting as a router to connect together two layer 2 VNs, the overlay system will forward frames to this router VM based on the VNIC's MAC address, but inside the frames may be packets destined to many different IP addresses. There is no need for the NVEs to know the IP address of the router VM itself, nor the IP addresses of other TS that have packets routing through the VM. However, it may be useful for the NVE to know the IP address of the router itself for either troubleshooting, or for providing other network optimizations such as local termination of ARP (even though ARP optimizations are not strictly layer 2). It is recommended (but optional) for an End Device to provide an IP address for a VNIC even if the NVE is providing an L2 service.

When the overlay VN is forwarding at layer 2, it is possible for Tenant Systems to perform bridging between two VNs belonging to that tenant (provided the tenant MAC addresses do not overlap between the two VNs that are being bridged). Reasons for VMs to do this are the same as in the physical world, such as the insertion of a transparent firewall device. For example, a VM running firewall software can be inserted in between two groups of Tenant Systems on the same subnet by putting each group on a different VN and having the firewall VM bridge between them.

When a VM is acting as a transparent bridge, it will appear to the overlay system that the VM is terminating multiple MAC addresses - one for each TS that exists on the other VN the VM is bridging to. In order for the overlay system to properly forward traffic to the bridging VM, it must know the MAC addresses of all the tenant systems the VM is bridging towards. This is one case where a VNIC can appear to terminate more than one MAC address for the same VN/VN.

[3.1.2.](#) Layer 3 Virtual Network Service

When the Virtual Network is providing Layer 3 forwarding, the NVEs must have knowledge of the Tenant System IP addresses. In the case where there is a Tenant System providing L3 forwarding for the tenant (e.g. an L3 VPN gateway), The TS VNIC may only terminate frames with a single MAC address, but will be forwarding IP packets on the behalf of other Tenant Systems. This scenario requires more exploration to determine how the TS forwarding interacts with the VN forwarding; However, in one scenario, the TS VNIC may be seen as containing many IP addresses.

Note that a MAC address is required even for a pure L3 VN service because VNICs filter out frames with destination MAC addresses that

do not match the VNIC's address; Therefore, the NVE providing an L3 service must first encapsulate an IP packet in an Ethernet frame with the VNIC's destination MAC before it is sent to the End Device containing the VNIC.

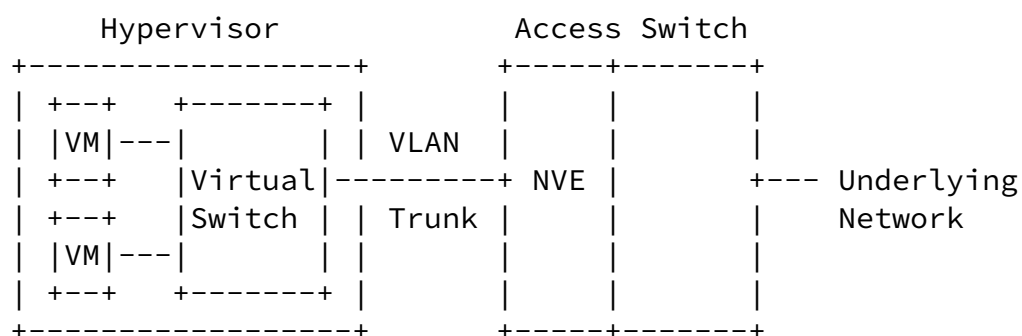
4. Hypervisor-to-NVE Control Plane Protocol Functionality

The problem statement [[I-D.ietf-nvo3-overlay-problem-statement](#)], discusses the needs for a control plane protocol (or protocols) to populate each NVE with the state needed to perform its functions.

In one common scenario, an NVE provides overlay encapsulation/decapsulation packet forwarding services to Tenant Systems (TSs) that are co-resident with 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.

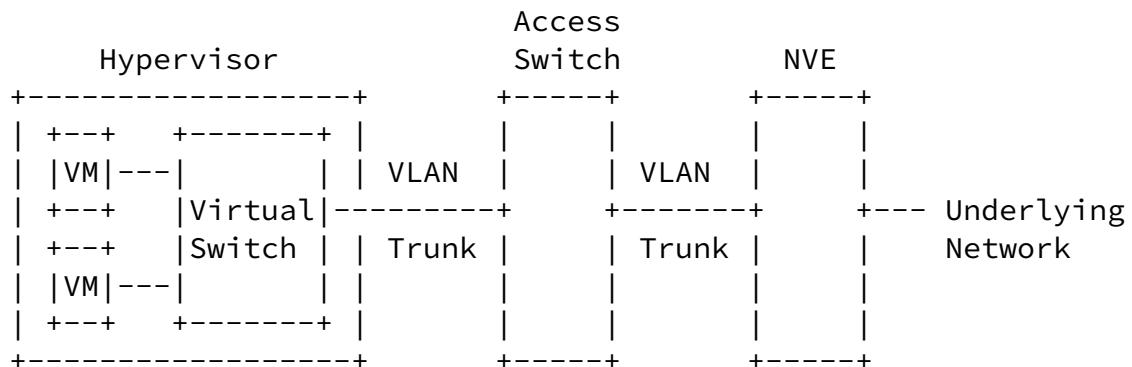
Alternatively, a Tenant System may use an externally connected NVE. An external NVE can provide an offload of the encapsulation / decapsulation function, network policy enforcement, as well as the VN Overlay protocol overheads. This offloading may provide performance improvements and/or resource savings to the End Device (e.g. hypervisor) making use of the external NVE.

The following figures give example scenarios where the Tenant System and NVE are on different devices separated by an access network.



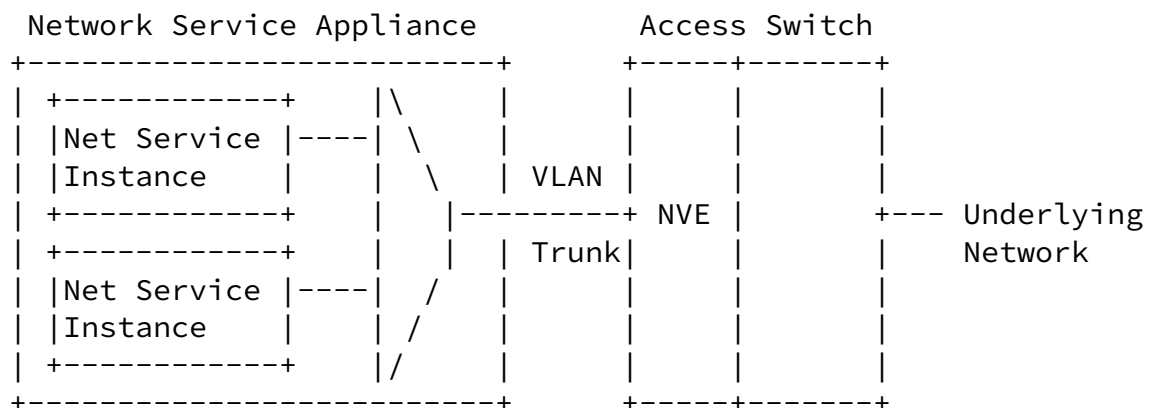
Hypervisor with an External NVE.

Figure 3

Internet-Draft NV03 Hypervisor-NVE Control Protocol Reqs February 2013


Hypervisor with an External NVE across an Ethernet Access Switch.

Figure 4



Physical Network Service Appliance with an External NVE.

Figure 5

In the examples above, the physical VLAN Trunk from the Hypervisor or Network Services Appliance towards the external NVE only needs to carry locally significant VLAN tag values. How "local" the significance is depends on whether the Hypervisor has a direct physical connection to the 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 NVE receives packets, it uses the VLAN tag to identify the VN of packets coming from a given Tenant System's VNIC, strips the tag, and

adds the appropriate overlay encapsulation for that VN.

On the hypervisor-facing side of the NVE, a control plane protocol is necessary to provide an NVE with the information it needs to provide connectivity across the Virtual Network for a given VNIC.

Specifically, the Hypervisor (or Network Service Appliance) utilizing an external NVE needs to "attach to" and "detach from" a VN, as well as communicate the addresses within that VN that are reachable within it. Thus, they will need a protocol that runs across the access network between the two devices that identifies the Tenant System (TS) VNIC addresses and VN Name (or ID) for which the NVE is providing service. In addition, such a protocol will identify a locally significant tag (e.g., an 802.1Q VLAN tag) that can be used to identify the data frames that flow between the TS VNIC and the VN.

[4.1.](#) VN Connect/Disconnect

In the previous figures, NVEs reside on an external networking device (e.g. an access switch). When an NVE is external, a protocol is needed between the End Device (e.g. Hypervisor) making use of the external NVE and the external NVE in order to make the 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.

The NVE must be notified when an End Device requires connection to a particular VN and when it no longer requires connection. In addition, the external NVE must provide a local tag value for each

connected VN to the End Device to use for exchange of packets between the End Device and the NVE (e.g. a locally significant 802.1Q tag value).

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. When a VN within a VDC is instantiated within a particular administrative domain, it can be allocated a VN Context which only the NVE needs to use. Once an 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-oracle (called NVE-to-IMA in this document) protocol which are part of work items 1 and 2 in [\[I-D.ietf-nvo3-overlay-problem-statement\]](#).

An End Device that is making use of an offloaded NVE only needs to communicate the VN Name or ID to the NVE, and get back a locally significant tag value.

[4.2.](#) VNIC Address Association

Typically, a VNIC is assigned a single MAC address and all frames transmitted and received on that VNIC use that single MAC address. As discussed in the section above on the containment hierarch, it is also possible for a Tenant System to exchange frames using multiple MAC addresses (ones that are not assigned to the VNIC) or packets with multiple IP addresses.

Particularly in the case of a TS that is forwarding frames or packets from other TSs, the 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 to the Information Mapping Authority (IMA).

The NVE has two ways in which it can discover the tenant addresses for which frames must 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 End Device can explicitly signal the addresses to the NVE. The End Device could have discovered the addresses for a given VNIC by gleaning them itself from data packets sent by the VNIC, or by some other internal means within the End Device itself.

To perform the second approach above, the "hypervisor-to-NVE" protocol requires a means to allow End Devices to communicate new tenant addresses associations for a given VNIC within a given VN.

4.3. VNIC Address Disassociation

When a VNIC within an End Device terminates function (due to events such as VNIC shutdown, Tenant System (TS) shutdown, or VM migration to another hypervisor), all addresses associated with that VNIC must be disassociated with the End Device on the connected NVE.

If the VNIC only has a single address associated with it, then this can be a single address disassociate message to the NVE. However, if the VNIC had hundreds of addresses associated with it, then the protocol with the NVE would be better optimized to simply disassociate the VNIC with the NVE, and the NVE can automatically disassociate all addresses that were associated with the VNIC.

Having TS addresses associated with a VNIC can also provide scalability benefits when the VM migrates between hypervisors that are connected to the same NVE. When a VM migrates to another hypervisor connected to the same NVE, if the NVE is aware of the migration, there is no need for all the addresses to be purged from NVE (and IMA) only to be immediately re-established again when the VM migration completes.

If the device containing the NVE is supporting many hypervisors, it may be quite likely that the VM migration will result in the VNICs still being associated with the same NVE, but simply on a different port. From the point of view of the IMA, nothing has changed and it would be inefficient to signal these changes to the IMA for no benefit. The NVE only needs to associate the addresses with a different port/tag pair.

It is possible for the NVE to handle a VM migration by using a timer to retain the VNIC addresses for a short time to see if the

disassociated VNIC re-associates on another NVE port, but this could be better handled if the NVE knew the difference between a VNIC/VM shutdown and a VM migration. This leads to the next section.

[4.4.](#) VNIC Shutdown/Startup/Migration

As discussed above, the NVE can make optimizations if it knows which addresses are associated with which VNICs within an End Device and also is notified of state changes of that VNIC, specifically the difference between VNIC shutdown/startup and VNIC migration arrival/departure.

Upon VNIC shutdown, the NVE can immediately signal to the IMA that the bindings of the VNIC's addresses to the NVE's IP address can be removed.

Upon VNIC arrival, the NVE could either start a timer to hold the VNIC address bindings waiting to see if the VNIC arrives on a different port, or if there is a pre-arrival handshake with the NVE, then it will already know that the VNIC is going to be reassociated with the same NVE.

Upon VNIC arrival, the NVE knows that any addresses previously bound to the VNIC are still present and has no need to signal any change in address mappings to the IMA.

Note that if the IMA is also aware of VNIC address bindings, it can similarly participate efficiently in a VM migration that occurs across two different NVEs.

[4.5.](#) VN Profile

Once an NVE (embedded or external) receives a VN connect indication with a specified VN Name or ID, the NVE must determine the VN Context value to encapsulate packets with as well as other information that may be needed (e.g., QoS settings). The NVE serving that hypervisor 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). A protocol for an NVE to get this mapping may be a useful function, but would be the subject of work items 1 and 2 in

[\[I-D.ietf-nvo3-overlay-problem-statement\]](#).

[5.](#) Security Considerations

Editor's Note: This is an initial start on the security considerations section; it will need to be expanded, and suggestions for material to add are welcome.

NVEs must ensure that only properly authorized Tenant Systems are allowed to join and become a part of any specific Virtual Network. In addition, NVEs will need appropriate mechanisms to ensure that any hypervisor wishing to use the services of an NVE are properly authorized to do so. One design point is whether the hypervisor should supply the NVE with necessary information (e.g., VM addresses, VN information, or other parameters) that the 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 NVE using that information to obtain the information needed to validate the hypervisor-provided parameters or obtain related parameters in a secure manner.

[6.](#) Acknowledgements

Thanks to the following people for reviewing and providing feedback: Vipin Jain and Shyam Kapadia.

[7.](#) Informative References

[I-D.gu-nvo3-overlay-cp-arch]

Yingjie, G. and W. Hao, "Analysis of external assistance to NVE and consideration of architecture", [draft-gu-nvo3-overlay-cp-arch-00](#) (work in progress), July 2012.

[I-D.gu-nvo3-tes-nve-mechanism]

Yingjie, G. and L. Yizhou, "The mechanism and signalling between TES and NVE", [draft-gu-nvo3-tes-nve-mechanism-01](#) (work in progress), October 2012.

[I-D.ietf-nvo3-framework]

Lasserre, M., Balus, F., Morin, T., Bitar, N., and Y. Rekhter, "Framework for DC Network Virtualization", [draft-ietf-nvo3-framework-02](#) (work in progress), February 2013.

[I-D.ietf-nvo3-overlay-problem-statement]

Narten, T., Gray, E., Black, D., Dutt, D., Fang, L., Kreeger, L., Napierala, M., and M. Sridharan, "Problem Statement: Overlays for Network Virtualization", [draft-ietf-nvo3-overlay-problem-statement-02](#) (work in progress), February 2013.

[I-D.kompella-nvo3-server2nve]

Kompella, K., Rekhter, Y., and T. Morin, "Signaling Virtual Machine Activity to the Network Virtualization Edge", [draft-kompella-nvo3-server2nve-01](#) (work in progress), October 2012.

[I-D.kreeger-nvo3-overlay-cp]

Kreeger, L., Dutt, D., Narten, T., and M. Sridharan, "Network Virtualization Overlay Control Protocol Requirements", [draft-kreeger-nvo3-overlay-cp-02](#) (work in progress), October 2012.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", [RFC 4301](#), December 2005.

Authors' Addresses

Lawrence Kreeger
Cisco

Email: kreeger@cisco.com

Thomas Narten
IBM

Email: narten@us.ibm.com

Internet-Draft NV03 Hypervisor-NVE Control Protocol Reqs February 2013

David Black
EMC

Email: david.black@emc.com

