

Network Working Group
Internet-Draft
Intended status: Informational
Expires: October 31, 2013

K. Kompella
Y. Rekhter
Juniper Networks
T. Morin
France Telecom - Orange Labs
D. Black
EMC Corporation
April 29, 2013

Signaling Virtual Machine Activity to the Network Virtualization Edge
draft-kompella-nvo3-server2nve-02

Abstract

This document proposes a simplified approach for provisioning network parameters related to Virtual Machine creation, migration and termination on servers. The idea is to provision the server, then have the server signal the requisite parameters to the relevant network device(s). Such an approach reduces the workload on the provisioning system and simplifies the data model that the provisioning system needs to maintain. It is also more resilient to topology changes in server-network connectivity, for example, reconnecting a server to a different network port or switch.

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 October 31, 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](http://trustee.ietf.org/license-info) and the IETF Trust's Legal Provisions Relating to IETF Documents (<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	2
1.1.	VM Creation	3
1.2.	VM Live Migration	4
1.3.	VM Termination	5
2.	Acronyms Used	6
3.	Virtual Networks	7
3.1.	Current Mode of Operation	8
3.2.	Future Mode of Operation	8
4.	Provisioning DCVPNs	9
5.	Signaling	9
5.1.	Preliminaries	9
5.2.	VM Operations	10
5.2.1.	Network Parameters	10
5.2.2.	Creating a VM	12
5.2.3.	Terminating a VM	14
5.2.4.	Migrating a VM	15
5.3.	Signaling Protocols	16
6.	Interfacing with DCVPN Control Planes	16
7.	Security Considerations	16
8.	IANA Considerations	17
9.	Acknowledgments	17
10.	Informative References	17
	Authors' Addresses	18

[1.](#) Introduction

To create a Virtual Machine (VM) on a server in a data center, one must specify parameters for the compute, storage, network and appliance aspects of the VM. At a minimum, this requires provisioning the server that will host the VM, and the Network Virtualization Edge (NVE) that will implement the virtual network for

the VM in addition to the VM's storage. Similar considerations apply to live migration and terminating VMs. This document proposes mechanisms whereby a server can be provisioned with all of the parameters for the VM, and the server in turn signals the networking aspects to the NVE. The NVE may be located on the server or in an

external network switch that may be directly connected to the server or accessed via an L2 (Ethernet) LAN or VLAN. The following sections capture the abstract sequence of steps for VM creation, live migration and deletion.

While much of the material in this draft may apply to virtual entities other than virtual machines that exist on physical entities other than servers, this draft is written in terms of virtual machines and servers for clarity.

[1.1](#). VM Creation

This section describes an abstract sequence of steps involved in creating a VM and making it operational (the latter is also known as "powering on" the VM). The following steps are intended as an illustrative example, not as prescriptive text; the goal is to capture sufficient detail to set a context for the signaling described in [Section 5](#).

Creating a VM requires:

1. gathering the compute, network, storage, and appliance parameters required for the VM;
2. deciding which server, network, storage and network appliance devices best match the VM requirements in the current state of the data center;
3. provisioning the server with the VM parameters;
4. provisioning the network element(s) to which the server is connected with the network-related parameters of the VM;
5. informing the network element(s) to which the server is connected about the VM's peer VMs, storage devices and other network appliances with which the VM needs to communicate;

6. informing the network element(s) to which a VM's peer VMs are connected about the new VM and its addresses;
7. provisioning storage with the storage-related parameters; and
8. provisioning necessary network appliances (firewalls, load balancers and "middle boxes").

Steps 1 and 2 are primarily information gathering. For Steps 3 to 8, the provisioning system talks actively to servers, network switches, storage and appliances, and must know the details of the physical

server, network, storage and appliance connectivity topologies. Step 4 is typically done using just provisioning, whereas Steps 5 and 6 may be a combination of provisioning and other techniques that may defer discovery of the relevant information. Steps 4 to 6 accomplish the task of provisioning the network for a VM, the result of which is a Data Center Virtual Private Network (DCVPN) overlaid on the physical network.

While shown as a numbered sequence above, some of these steps may be concurrent (e.g., server, storage and network provisioning for the new VM may be done concurrently), and the two "informing" steps for the network (5 and 6) may be partially or fully lazily evaluated based on network traffic that the VM sends or receives after it becomes operational.

This document focuses on the case where the network elements in Step 4 are not co-resident with the server, and describes how the provisioning in Step 4 can be replaced by signaling between server and network, using information from Step 3.

[1.2.](#) VM Live Migration

This subsection describes an abstract sequence of steps involved in live migration of a VM. 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, suspend/resume (or "cold") migration consists of suspending VM execution on one server and resuming it on another. The following

live migration steps are intended as an illustrative example, not as prescriptive text; the goal is to capture sufficient detail to provide context for the signaling described in [Section 5](#).

For simplicity, this set of abstract steps assumes shared storage, so that the VM's storage is accessible to the source and destination servers. Live migration of a VM requires:

1. deciding which server should be the destination of the migration based on the VM's requirements, data center state and reason for the migration;
2. provisioning the destination server with the VM parameters and creating a VM to receive the live migration;
3. provisioning the network element(s) to which the destination server is connected with the network-related parameters of the VM;

4. transferring the VM's memory image between the source and destination servers;
5. actually moving the VM: pausing the VM's execution on the source server, transferring the VM's execution state and any remaining memory state to the destination server and continuing the VM's execution on the destination server;
6. informing the network element(s) to which the destination server is connected about the VM's peer VMs, storage devices and other network appliances with which the VM needs to communicate;
7. informing the network element(s) to which a VM's peer VMs are connected about the VM's new location;
8. activating the VM's network parameters at the destination server;
9. deprovisioning the VM from the network element(s) to which the source server is connected; and
10. deleting the VM from the source server.

Step 1 is primarily information gathering. For Steps 2, 3, 9 and 10, the provisioning system talks actively to servers, network switches and appliances, and must know the details of the physical server, network and appliance connectivity topologies. Steps 4 and 5 are usually handled directly by the servers involved. Steps 6 to 9 may be handled by the servers (e.g., one or more "gratuitous" ARPs or RARPs from the destination server may accomplish all four steps) or other techniques. For steps 6 and 7, the other techniques may involve discovery of the relevant information after the VM has been migrated.

While shown as a numbered sequence above, some of these steps may be concurrent (e.g., moving the VM and associated network changes), and the two "informing" steps (6 and 7) may be partially or fully lazily evaluated based on network traffic that the VM sends and/or receives after it is migrated to the destination server.

This document focuses on the case where the network elements are not co-resident with the server, and shows how the provisioning in Step 3 and the deprovisioning in Step 9 can be replaced by signaling between server and network, using information from Step 3.

[1.3.](#) VM Termination

This subsection describes an abstract sequence of steps involved in termination of a VM, also referred to as "powering off" a VM. The following termination steps are intended as an illustrative example, not as prescriptive text; the goal is to capture sufficient detail to set a context for the signaling described in [Section 5](#).

Termination of a VM requires:

1. ensuring that the VM is no longer executing;
2. deprovisioning the VM from the network element(s) to which the server is connected; and
3. deleting the VM from the server (the VM's image may remain on storage for reuse).

Steps 1 and 3 are handled by the server, based on instructions from the provisioning system. For Step 2, the provisioning system talks actively to servers, network switches, storage and appliances, and must know the details of the physical server, network, storage and appliance connectivity topologies.

While shown as a numbered sequence above, some of these steps may be concurrent (e.g., network deprovisioning and VM deletion).

This document focuses on the case where the network elements in Step 2 are not co-resident with the server, and shows how the deprovisioning in Step 3 can be replaced by signaling between server and network.

[2.](#) Acronyms Used

The following acronyms are used:

DCVPN: Data Center Virtual Private Network -- a virtual connectivity topology overlaid on physical devices to provide virtual devices with the connectivity they need and isolation from other DCVPNs. This corresponds to the concept of a Virtual Network Instance (VNI) in [[I-D.ietf-nvo3-framework](#)].

NVE: Network Virtualization Edge -- the entities that realize private communication among VMs in a DCVPN

l-NVE: local NVE: wrt a VM, NVE elements to which it is directly connected

r-NVE: remote NVE: wrt a VM, NVE elements to which the VM's peer VMs are connected

NVGRE: Network Virtualization using Generic Routing Encapsulation

VDP: VSI Discovery and Configuration Protocol

VID: 12-bit VLAN tag or identifier used locally between a server and its l-NVE

VLAN: Virtual Local Area Network

VM: Virtual Machine (same as Virtual Station)

Peer VM: wrt a VM, other VMs in the VM's DCVPN

VNID: DCVPN Identifier

VSI: Virtual Station Interface

VXLAN: Virtual eXtensible Local Area Network

[3.](#) Virtual Networks

The goal of provisioning a network for VMs is to create an "isolation domain" wherein a group of VMs can talk freely to each other, but communication to and from VMs outside that group is restricted (either prohibited, or mediated via a router, firewall or other network gateway). Such an isolation domain, sometimes called a Closed User Group, here will be called a Data Center Virtual Private Network (DCVPN). The network elements on the outer border or edge of the overlay portion of a Virtual Network are called Network Virtualization Edges (NVEs).

A DCVPN is assigned a global "name" that identifies it in the management plane; this name is unique in the scope of the data center, but may be unique across several cooperating data centers. A DCVPN is also assigned an identifier unique in the scope of the data center, the Virtual Network Group ID (VNID). The VNID is a control plane entity. A data plane tag is also needed to distinguish different DCVPNs' traffic; more on this later.

For a given VM, the NVE can be classified into two parts: the network elements to which the VM's server is directly connected (the local NVE or l-NVE), and those to which peer VMs are connected (the remote NVE or r-NVE). In some cases, the l-NVE is co-resident with the server hosting the VM; in other cases, the l-NVE is separate (distributed l-NVE). The latter case is the one of primary interest in this document.

A created VM is added to a DCVPN through Steps 4 to 6 in section

[Section 1.1](#) which can be recast as follows. In Step 4, the l-NVE(s) are informed about the VM's VNID, network addresses and policies, and the l-NVE and server agree on how to distinguish traffic for different DCVPNs from and to the server. In Step 5 the relevant r-NVE elements and the addresses of their VMs are discovered, and in Step 6, the r-NVE(s) are informed of the presence of the new VM and obtain or discover its addresses; for both steps 5 and 6, the discovery may be lazily evaluated so that it occurs after the VM begins sending and receiving DCVPN traffic.

Once a DCVPN is created, the next steps for network provisioning are to create and apply policies such as for QoS or access control. These occur in three flavors: policies for all VMs in the group, policies for individual VMs, and policies for communication across DCVPN boundaries.

[3.1.](#) Current Mode of Operation

DCVPNs are often realized as Ethernet VLAN segments. A VLAN segment satisfies the communication properties of a DCVPN. A VLAN also has data plane mechanisms for discovering network elements (Layer 2 switches, aka bridges) and VM addresses. When a DCVPN is realized as a VLAN, Step 4 in [section 1.1](#) requires provisioning both the server and l-NVE with the VLAN tag that identifies the DCVPN. Step 6 requires provisioning all involved network elements with the same VLAN tag. Address learning is done by flooding, and the announcement of a new VM or the new location of a migrated VM is often via a "gratuitous" ARP or RARP.

While VLANs are familiar and well-understood, they have scaling challenges because they are Layer 2 infrastructure. The number of independent VLANs in a Layer 2 domain is limited by the 12-bit size of the VLAN tag. In addition, data plane techniques (flooding and broadcast) are another source of scaling concerns as the overall size of the network grows.

[3.2.](#) Future Mode of Operation

There are multiple scalable realizations of DCVPNs that address the isolation requirements of DCVPNs as well as the need for a scalable substrate for DCVPNs and the need for scalable mechanisms for NVE and VM address discovery. While describing these approaches beyond the scope of this document, a secondary goal of this document is to show how the signaling that replaces Step 4 in [section 1.1](#) can seamlessly interact with realizations of DCVPNs.

VLAN tags (VIDs) will be used as the data plane tag to distinguish traffic for different DCVPNs' between a server and its l-NVE. Note that, as used here, VIDs only have local significance between server and NVE, and should not be confused with data-center-wide usage of VLANs. If VLAN tags are used for traffic between NVEs, that tag usage depends on the encapsulation mechanism among the NVEs and is orthogonal to VLAN tag usage between servers and l-NVEs.

[4.](#) Provisioning DCVPNs

For VM creation as described in section [Section 1.1](#), Step 3 provisions the server; Steps 4 and 5 provision the l-NVE elements; Step 6 provisions the r-NVE elements.

In some cases, the l-NVE is located within the server (e.g., a software-implemented switch within a hypervisor); in this case, Steps 3 and 4 are "single-touch" in that the provisioning system need only talk to the server, as both compute and network parameters are applied by the server. However, in other cases, the l-NVE is separate from the server, requiring that the provisioning system talk to both the server and l-NVE. This scenario, which we call "distributed local NVE", is the one considered in this document. This draft's goal is to describe how "single-touch" provisioning can be achieved in the distributed l-NVE case.

The overall approach is to provision the server, and have the server signal the requisite parameters to the l-NVE. This approach reduces the workload on the provisioning system, allowing it to scale both in the number of elements it can manage, as well as the rate at which it can process changes. It also simplifies the data model of the network that is used by the provisioning system, because a complete, up-to-date map of server to network connectivity is not required. This approach is also more resilient to server-network connectivity/topology changes that have not yet been transmitted to the provisioning system. For example, if a server is reconnected to a different port or a different l-NVE to recover from a malfunctioning port, the server can contact the new l-NVE over the new port without the provisioning system needing to immediately be aware of the change.

While this draft focuses on provisioning networking parameters via signaling, extensions may address the provisioning of storage and network appliance parameters in a similar fashion.

[5.](#) Signaling

This draft considers three common VM operations in a virtualized data center: creating a VM; migrating a VM from one physical server to another; and terminating a VM. Creating a VM requires "associating" it with its DCVPN and "activating" that association; decommissioning a VM requires "dissociating" the VM from its DCVPN. Moving a VM consists of associating it with its DCVPN in its new location, activating that association, and dissociating the VM from its old location.

5.2. VM Operations

5.2.1. Network Parameters

For each VM association or dissociation operation, a subset of the following information is needed from server to l-NVE:

operation: one of associate or dissociate.

authentication: proof that this operation was authorized by the provisioning system

VNID: identifier of DCVPN to which VM belongs

VID: tag to use between server and l-NVE to distinguish DCVPN traffic; the value zero in an associate operation is a request that the l-NVE to assign an unused VID. This approach provides extensibility by allowing the VID to be a VLAN-id, although other local means of multiplexing traffic between the server and the NVE could be used instead of VIDs.

encapsulation type: type of encapsulation used by the DCVPN for traffic exchanged between NVEs (see below).

network addresses: network addresses for VM on the server (e.g., MACs)

policy: VM-specific and/or network-address-specific network policies, such as access control lists and/or QoS policies

hold time: time (in milliseconds) to keep a VM's addresses after it migrates away from this l-NVE. This is usually set to zero when a VM is terminated.

per-address-VID-allocation: boolean flag which can optionally be set to "yes", resulting in the VID allocated to the this address being distinct from the VID allocated to other addresses (for the same VM or other VMs) connected to the same DCVPN on a same NVE port; this behavior will result in traffic always transiting through the

NVE, even to/from other addresses for the same DCVPN on the same server.

The "activate" operation is a dataplane operations that references a previously established association via the address and VID; all other parameters are obtained at the NVE by mapping the source address, VID and port involved to obtain information established by a prior associate operation.

Realizations of DCVPNs include, E-VPNs ([[I-D.ietf-l2vpn-evpn](#)]), IP VPNs ([[RFC4364](#)]), NVGRE ([[I-D.sridharan-virtualization-nvgre](#)]), VPLS ([[RFC4761](#)], [[RFC4762](#)]), and VXLAN ([[I-D.mahalingam-dutt-dcops-vxlan](#)]). The encapsulation type determines whether forwarding at the NVE for the DCVPN is based on Layer 2 or Layer 3 service.

Typically, for the associate messages, all of the above information except hold time would be needed. Similarly, for the dissociate message, all of the above information except VID and encapsulation type would typically be needed.

These operations are stateful in that their results remain in place until superseded by another operation. For example, on receiving an associate message, an NVE is expected to create and maintain the DCVPN information for the addresses until the NVE receives a dissociate message to remove that information. A separate liveness protocol may be run between server and NVE to let each side know that the other is still operational; if the liveness protocol fails, each side may remove state installed in response to messages from the other.

The descriptions below generally assume that the NVEs participate in

a mechanism for control plane distribution of VM addresses, as opposed to doing this in the data plane. If this is not the case, NVE elements can lazily evaluate (via data plane discovery) the parts of the procedures below that involve address distribution.

As VIDs are local to server-NVE communication, in fact to a specific port connecting these two elements, a mapping table containing 4-tuples of the following form will prove useful to the NVE:

<VID, port, VNID, VM network address>

The valid VID values are from 1 to 4094, inclusive. A value of 0 is used to mean "unassigned". When a VID can be shared by more than one VM, it is necessary to reference-count entries in this table; the list of addresses in an entry serves this purpose. Entries in this table have multiple uses:

- o Finding the VNID for a VID and port for association, activation and traffic forwarding;
- o Determining whether a VID exists (has already been assigned) for a VNID and port.
- o Determining which <VID, port> pairs to use for forwarding traffic that requires flooding on the DCVPN.

For simplicity and clarity, this draft assumes that the network interfaces in VMs (vNICs) do not use VLAN tags.

[5.2.2.](#) Creating a VM

When a VM is instantiated on a server (powered on, e.g., after creation), each of the VM's interfaces is assigned a VNID, one or more network addresses and an encapsulation type for the DCVPN. The VM addresses may be any of IPv4, IPv6 and MAC addresses. There may also be network policies specific to the VM or its interfaces. To

connect the VM to its DCVPN, the server signals these parameters to the l-NVE via an "associate" operation followed by an "activate" operation to put the parameters into use. (Note that the l-NVE may consist of more than one device.)

On receiving an associate message on port P from server S, an NVE device does the following for each network address in that message:

A.1: Validate the authentication (if present). If not, inform the provisioning system, log the error, and stop processing the associate message. This validation may include authorization checks.

A.2: Check the per-address-VID-allocation flag in the associate message:

- * if this flag is not set:

- + Check if the VID in the associate message is zero (i.e., the associate message requests VID allocation); if so, look up the VID for <VNID, port, network address> ; if there is no current VID for that tuple, allocate a new VID

- + If the VID in the associate message is non-zero, look up the VID for <VNID, port>. If that lookup results in the same VID as the one in the associate message, associate that VID with <VNID, network address>. If the lookup indicates that there is no current VID for that tuple, associate the VID in the associate message with <VNID, port, network address>. Otherwise, the VID in the associate message does not match the VID that is currently in use for <VNID, port>, so respond to S with an error, and stop processing the associate message.
- * if this flag is set, check if the VID in the associate message is zero :
 - + if so, this is an allocation request, so allocate a new VID, distinct from other VIDs allocated on this port;
 - + if the VID is non-zero, check that the provided VID is

distinct from other VIDs allocated on this port; if so, associate the VID with <VNID, port, network address>. If not, the provided VID is already in used and hence cannot be dedicated to this network address, so respond to S with an error, and stop processing the associate message.

A.3: Add the <VID, port, VNID, network address> entry to the NVE's mapping table. This table entry includes information about the DCVPN encapsulation type for the VNID.

A.4: Communicate with the control plane to advertise the network address, and (if the VNID is new to the NVE) also to get other network addresses in the DCVPN. Populate the NVE's mapping table with all of these network addresses (some control planes may not provide all or even any of the other addresses in the DCVPN at this point).

A.5: Finally, respond to S with the VID for <VNID, port, network address>, and indicate that the operation was successful.

After a successful associate, the network has been provisioned (at least in the local NVE) for traffic, but forwarding has not been enabled. On receiving an activate message on port P from server S, an NVE device does the following (activate is a one-way message that does not have a response):

B.1: Validate the authentication (if present). If not, inform the provisioning system, log the error, and stop processing the associate message. This validation may include authorization checks. The authentication and authorization may be implicit when

the activate message is a dataplane frame (e.g., a "gratuitous" ARP or RARP).

B.2: Check if the VID in the activate message is zero. If so, log the error, and stop processing the activate message.

B.3: Use the VID and port P to look up the VNID from a previous associate message. If there is no mapping table state for that VID and port, log the error and stop processing the activate message.

B.4: If forwarding is not enabled for <VID, port, network address> activate it, mapping VID -> VNID on this port (P) for traffic sent to and received from r-NVEs.

B.5: If the activate message is a dataplane frame that requires forwarding beyond the NVE, (e.g., a "gratuitous" ARP or RARP), use the activated forwarding to send the frame onward via the virtual network identified by the VNID.

5.2.3. Terminating a VM

On receiving a request from the provisioning system to terminate execution of a VM (powering off the VM, whether or not the VM's image is retained on storage), the server sends a dissociate message to the l-NVE with the hold time set to zero. The dissociate message contains the operation, authentication, VNID, encapsulation type, and VM addresses. On receiving the dissociate message on port P from server S, each NVE device L does the following:

D.1: Validate the authentication (if present). If not, inform the provisioning system, log the error, and stop processing the associate message.

D.2: Communicate with the control plane to withdraw the VM's addresses. If the hold time is as non-zero, wait until the hold time expires before proceeding to the next step.

D.3: Delete the VM's addresses from the mapping table and delete any VM-specific network policies associated with any of the VM addresses. If a mapping tuple contains no VM addresses as a result delete that tuple. If the mapping table contains no entries for the VNID involved after deleting the tuple, optionally delete any network policies for the VNID.

D.4: Respond to S saying that the operation was successful.

At step D.2, the control plane is responsible for not disrupting network operation if the addresses are in use at another l-NVE. Also, l-NVEs cannot rely on receiving dissociate messages for all terminated VMs, as a server crash may implicitly terminate a VM

before a dissociate message can be sent.

[5.2.4.](#) Migrating a VM

Consider a VM that is being migrated from server S (connected to l-NVE device L) to server S' (connected to l-NVE device L'). This section assumes shared storage, so that both S and S' have access to the VM's storage. The sequence of steps for a successful VM migration is:

- M.1: S' gets a request to prepare to receive a copy of the VM from S.
- M.2: S gets a request to copy the VM to S'.
- M.3: The copy of the VM (memory, configuration state, etc.) occurs while the VM continues to execute.
- M.4: When that copy has made sufficient progress, S pauses the VM, and completes the copy, including the VM's execution state.
- M.5: S' gets a request to resume the paused VM.
- M.6: After that resume has succeeded, S then proceeds to terminate the paused VM on S, see [section 5.2.3](#), but this operation may specify a non-zero hold time during which traffic received may be forwarded to the VM's new location.

Steps M.1 and M.2 initiate the copy of the VM. During step M.3, S' sends an "associate" message to L' for each of the VM's network addresses (S' receives information about these addresses as part of the VM copy). Step M.4 occurs when the VM copy has made sufficient progress that the pause required to transfer the VM's execution from S to S' is sufficiently short. At step M.4, or M.5 at the latest, S' sends an "activate" message to L' for each of the VM's interfaces. At Step M.6, S sends a "dissociate" message to L for each of the VM's network addresses, optionally with a non-zero hold time.

From the DCVPN's view, there are two important overlaps in the apparent network location of the VM's addresses:

- o The VM's addresses are associated with both L and L' between steps M.3 and M.6.

- o The VM's addresses are activated at L' during step M.4 or step M.5 at the latest (e.g., if activate is a dataplane operation based on traffic sent at that step); both of these typically occur before these addresses are dissociated at L during step M.6

The DCVPN control plane must work correctly in the presence of these overlaps, and in particular must not:

- o Fail to activate the VM's network addresses at L' because they have not yet been withdrawn at L, or
- o Disruptively withdraw the VM's network addresses from use at step M.6 of a migration when the VM continues to execute on a different server.

An additional scenario that is important for migration is that the source and destination servers, S and S', may share a common l-NVE, i.e., L and L' are the same. In this scenario there is no need for remote interaction of that l-NVE with other NVEs, but that NVE must be aware of the possibility of a new association of the VM's addresses with a different port and the need to promptly activate them on that port even though they have not (yet) been dissociated from their original port.

[5.3.](#) Signaling Protocols

There are multiple protocols that can be used to signal the above messages. One could invent a new protocol for this purpose, or reuse existing protocols, among them LLDP, XMPP, HTTP REST, and VDP [[VDP](#)], a new protocol standardized for the purposes of signaling a VM's network parameters from server to l-NVE. Multiple factors influence the choice of protocol(s); this draft's focus is on what needs to be signaled, leaving choices of how the information is signaled, and specific encodings for other drafts to consider.

[6.](#) Interfacing with DCVPN Control Planes

The control plane for a DCVPN manages the creation/deletion, membership and span of the DCVPN ([I-D.ietf-nvo3-overlay-problem-statement], [[I-D.kreeger-nvo3-overlay-cp](#)]). Such a control plane needs to work with the server-to-nve signaling in a coordinated manner, to ensure that address changes at a local NVE are reflected appropriately in remote NVEs. The details of such coordination are specified in separate documents.

[7.](#) Security Considerations

Internet-Draft

Signaling VM Activity to NVE

April 2013

[8.](#) IANA Considerations

[9.](#) Acknowledgments

Many thanks to Amit Shukla for his help with the details of EVB and his insight into data center issues. Many thanks to members of the nvo3 WG for their comments, including Yingjie Gu.

[10.](#) Informative References

[I-D.ietf-l2vpn-evpn]

Sajassi, A., Aggarwal, R., Henderickx, W., Balus, F., Isaac, A., and J. Uttaro, "BGP MPLS Based Ethernet VPN", [draft-ietf-l2vpn-evpn-03](#) (work in progress), February 2013.

[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.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.

[I-D.mahalingam-dutt-dcops-vxlan]

Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", [draft-mahalingam-dutt-dcops-vxlan-03](#) (work in progress), February 2013.

[I-D.sridharan-virtualization-nvgre]

Sridharan, M., Greenberg, A., Venkataramaiah, N., Wang, Y., Duda, K., Ganga, I., Lin, G., Pearson, M., Thaler, P., and C. Tumuluri, "NVGRE: Network Virtualization using Generic Routing Encapsulation", [draft-sridharan-virtualization-nvgre-02](#) (work in progress), February 2013.

Kompella, et al.

Expires October 31, 2013

[Page 17]

Internet-Draft

Signaling VM Activity to NVE

April 2013

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", [RFC 4364](#), February 2006.

[RFC4761] Kompella, K. and Y. Rekhter, "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling", [RFC 4761](#), January 2007.

[RFC4762] Lasserre, M. and V. Kompella, "Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling", [RFC 4762](#), January 2007.

[VDP] IEEE, "Edge Virtual Bridging (IEEE Std 802.1Qbg-2012)", July 2012.

Authors' Addresses

Kireeti Kompella
Juniper Networks
1194 N. Mathilda Ave.
Sunnyvale, CA 94089
US

Email: kireeti@juniper.net

Yakov Rekhter
Juniper Networks
1194 N. Mathilda Ave.
Sunnyvale, CA 94089
US

Email: yakov@juniper.net

Thomas Morin
France Telecom - Orange Labs
2, avenue Pierre Marzin
Lannion 22307
France

Email: thomas.morin@orange.com

Kompella, et al.	Expires October 31, 2013	[Page 18]
------------------	--------------------------	-----------

Internet-Draft	Signaling VM Activity to NVE	April 2013
----------------	------------------------------	------------

David L. Black
EMC Corporation
176 South St.
Hopkinton, MA 01748

Email: david.black@emc.com

