Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2013

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

October 22, 2012

Signaling Virtual Machine Activity to the Network Virtualization Edge
draft-kompella-nvo3-server2nve-01

Abstract

This document proposes a simplified approach for provisioning the networking 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. Furthermore, it is 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 April 25, 2013.

Table of Contents

1. Introduction

To create a Virtual Machine (VM) on a server in a data center, one must specify parameters for the CPU, 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. Similar considerations apply to live migration and terminating VMs. This document proposes mechanisms whereby a server can be provisioned with all of the paramters 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 subsections capture the abstract sequence of steps for VM creation, live migration and deletion.

1.1. VM Creation

This subsection describes an abstract sequence of steps involved in creating a VM and make it operational. 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 CPU, network, storage, and appliance parameters required for the VM;

2. deciding which server, network, storage and 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 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 appliances (firewalls, load balancers and "middle boxes").

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).

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. 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.

This document focuses on the case where the network elements in Step 4 are not co-resident with the server, and shows how the provisioning in Step 4 can be replaced by signaling between server and network, using information from Step 3. This document also shows how Step 4 can interact seamlessly with some of the realizations of Steps 5 and 6.

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 consistes 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 set a 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 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. deactivating the VM's network parameters at the source server;

10. deprovisioning the VM from the network element(s) to which the source server is connected; and

11. deleting the VM at the source server.

While shown as a numbered sequence above, some of these steps may be concurrent (e.g., moving the VM and associated network changes).

Step 1 is primarily information gathering. For Steps 2, 3, 10 and 11, 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., a gratuitous ARP or RARP from the destination server may accomplish all four steps) or other techniques.

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 10 can be replaced by signaling between server and network, using information from Step 3. This document also shows how Step 4 can interact seamlessly with some of the realizations of Steps 5 and 6.

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. deactivating the VM's network parameters at the server;

3. deprovisioning the VM from the network element(s) to which the server is connected; and

4. deleting the VM from the server (the VM's image may remain in storage for reuse).

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

Steps 1, 2 and 4 are handled by the server, based on instructions from the provisioning system. For Step 3, 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.

This document focuses on the case where the network elements in Step 3 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. Conventions and Acronyms Used

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

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

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

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

rNVE: 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 lNVE

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 (sometimes called a Group Identifier)

VSI: Virtual Station Interface

VXLAN: Virtual eXtensible Local Area Network

3. Virtual Networks

The goal of provisioning networks 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, a 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 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 lNVE 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. In Step 6, the r-NVE(s) are informed of the presence of the new VM and obtain its addresses.

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 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 is typically by a "gratuitous ARP".

While VLANs are familiar and well-understood, they fail to scale on several dimensions. Underlying VLANs is a Layer 2 infrastructure. The number of independent VLANs in a Layer 2 domain is limited by the size of the VLAN tag. Data plane techniques (flooding and broadcast) are another source of serious concern as the overall size of the network grows.

3.2. Future Mode of Operation

There are several 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 these are not the goal of this document,

a secondary goal of this document is to show how the signaling that replaces Step 4 can seamlessly interact with several of these 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, not to be confused with the notion of VLANs, which are a data center-wide concept. Data plane tags between l-NVE and r-NVE depends on the encapsulation mechanism among the NVE; the l-NVE is expected to map between VIDs and intra-NVE tags in both directions.

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 elements live within the server; in this case, Steps 3 and 4 are "single-touch" in that the provisioning system only needs to talk to the server, and both CPU and network parameters can be applied by the server. However, in other cases, the l-NVE is separate from the server, requiring that the provisioning system talk independently to both the server and lNVE. This scenario, which we call "distributed local NVE", is the one considered in this document. This document resurrects "single-touch" provisioning in the distributed lNVE case.

The approach here is to provision the server, then have the server signal the requisite parameters to the l-NVE. Such an 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 that the provisioning system needs to have; in particular, the provisioning system does not have to maintain a full, up-to-date map of server to network connectivity. Furthermore, it is more resilient to topology changes in server-network connectivity 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 being aware of the change.

While the current document focuses on provisioning networking parameters via signaling, future extensions may address the provisioning of storage and middle-box parameters in a similar fashion. Companion documents will describe how NVEs to which peer

VMs are connected can get the required networking information via signaling rather than by provisioning and/or other means.

5. Signaling

5.1. Preliminaries

There are three common 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 "deactivating" the VM's association with the DCVPN and then "dissociating" the VM from its DCVPN. Moving a VM consists of associating it with its DCVPN in its new location, then dissociating it from its old location. . The deactivation operation is often implicit in another operation, but is called out here for symmetry and completeness.

5.2. VM Operations

5.2.1. Network Parameters

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

operation: one of pre-associate, 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 lNVE to distinguish DCVPN traffic; the value zero in an associate or pre-associate operation is a request to the l-NVE to assign an unused VID. These specifications are meant to provide extensibility by allowing the VID to be a VLAN-id, but also any another means of locally multiplexing traffic betwen the server and the nve. In the case where the NVE is implemented on the server, the VID can be the a local name of a virtual network interface.

table type: realization of DCVPN on NVE (see below).

address entries: addresses for VM on server

policy: VM-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 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 VM being distinct from the VID allocated to other VMs connected to the same DCVPN on a same NVE port; this behavior will result in traffic between to/from the VM to always transit through the NVE, even from/to VMs of a same DCVPN

Activate and deactivate are dataplane operations that reference the VID, and additionally provide authentication, table type and address entries information. When an activate is realized via a "gratuitous ARP" in the data plane, the VID is in the Ethernet header, and all of the other parameters are obtained by mapping the VID and the port on which the frame containing it was received to information established by a prior associate operation.

Realizations of DCVPNs include, among others, E-VPNs ([I-D.ietf-l2vpn-evpn]), IP VPNs ([RFC4364]), NVGRE ([I-D.sridharan-virtualization-nvgre], TRILL ([RFC6325]), VPLS ([RFC4761], [RFC4762]), and VXLAN ([I-D.mahalingam-dutt-dcops-vxlan]). The table type implicitly defines whether forwarding at the NVE for the DCVPN is at Layer 2 or Layer 3 or both.

Typically, for the pre-associate and associate messages, all the information except hold time would be needed. For the dissociate message, all the above information except VID and table type would be needed.

Operations are stateful, that is, they remain in place until superceded by another operation. For example, on receiving an associate message, an NVE is expected to create and maintain the DCVPN table for a VM until the NVE receives a dissociate message to remove the table. 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 all state installed in response to messages from the other.

In the descriptions below, we assume that the NVE layer provides 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 skip 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 containg 4-tuples of the following form will prove useful to the NVE:

<VID, port, VNID, VM address entries>

The procedures below assume that the NVE systematically reorders the provided VM address entries before inserting or looking up entries in this mamping table.

Note that valid values of VID 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. Entries in this table have multiple uses:

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

5.2.2. Creating a VM

When a VM is instantiated on a server, it is assigned a VNID, VM addresses and a table 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. 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:

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 is the associate message:

* if this flag is not set:

+ Check if the VID in the associate message is zero (i.e., an allocation request); if so, look up the VID for <VNID, P, VM address entries> ; if there is none, allocate a new VID

+ If the VID in the associate message is non-zero, look up <VNID, P, VM address entries> for an already allocated VID. If the looup is successful, associate the resulting VID with <VNID, P, VM address entries>. If the result is zero, associate the VID with <VNID, P, VM address entries>. Otherwise, the provided VID does not match the one in use for <VNID, P>, 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), 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, P, VM address entries>. If not, the provided VID does not match the per-address-VID-constraint, so respond to S with an error, and stop processing the associate message.

A.3: Add the <VID, P, VM address entries> -> VNID mapping to the mapping table

A.4: If a table of appropriate type (as signaled) for VNID does not already exist, create it, and add the VM's addresses to it.

A.5: Commmunicate with the control plane to advertise the VM's addresses, and also to get the addresses of other VMs in the DCVPN. Populate the table with the VM's addresses and any addresses learned from the control plane (some control planes may not provide all or even any of the other addresses in the DCVPN at this point).

A.6: Finally, respond to S with the VID for <VNID, P, VM address entries>, and also saying that the operation was successful.

After a successful associate, the network has been provisioned (at least in the local NVE) for the VM's 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.

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 VNID, log the error and stop processing the activate message.

B.4: If forwarding is not enabled for <VID, P, VM address entries> activate it, mapping VID -> VNID.

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

5.2.3. Terminating a VM

On receiving a request from the provisioning system to terminate a VM, 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, table 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: Delete the VM's addresses from the mapping table and delete any VM-specific network policies associated with any of the VM addresses. If the VNID table is empty after deleting the VM's addresses, optionally delete the table and any network policies for the VNID.

D.3: Respond to S saying that the operation was successful.

5.2.4. Migrating a VM

NOTE: This sub section has not been updated from the -00 version of this draft; it will be updated in the forthcoming -02 version. The set of VM migration steps are known to be incomplete, material on concurrent actions and race conditions (based on list discussion) should be added and new step PA.5 is anticipated to need

generalization to encompass control planes that may not push all addressing changes to all relevant rNVEs. Please ignore the text in this subsection and beyond - this document is a draft and the authors are working on it.

Let's say that a VM is to be migrated from server S (connected to lNVE device L) to server S' (connected to lNVE device L'). The sequence of steps for 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: S then gets a request to terminate the VM on S.

M.4: Finally, S' gets a request to start up the VM on S'.

At Step M.1, S' initiates the move, and also sends a pre-associate message to L', including the pre-associate information. The processing of a pre-associate message (PA.1 to PA.7) for L' is the same as that of an associate message (A.1 to A.6), with the following change to step 5.

PA.5: Commmunicate with each rNVE device to advertise the VM's addresses but as non-preferred destinations(*). Also get the addresses of other VMs in the DCVPN. Populate the table with the VM's addresses and addresses learned from each rNVE.

(*) See Section 6 for some mechanisms for doing this. This is necessary so that L' does not attract traffic to the VM's new location before the migration is complete, yet L knows ahead of time how to send traffic to L' (Step D.2), minimizing traffic loss to the VM when migration is complete.

At step M.2, S initiates the VM copy. If at any time L hears advertisements from L' about how to communicate with the VM in its new location (as unpreferred destinations), L stores that information for use in step D.2.

At step M.3, S terminates the running of the VM on itself, and sends a dissociate message to L with a non-zero hold time (either what the provisioning system sends, or a default value). L processes the dissociate message as above.

5.3. Signaling Protocols

There are several options for protocols to use to signal the above messages. One could invent a new protocol for this purpose. One could 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 lNVE. Several factors influence the choice of protocol(s); at this time, the focus is on what needs to be signaled, leaving for later the choice of how the information is signaled, and specific encodings.

5.4. Liveness

Procedures to handle failures of the server or of the NVE will be covered in a further revision.

6. Interfacing with DCVPN Control Planes

The control plane for a DCVPN manages the creation/deletion, membership and span of the DCVPN ([I-D.narten-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 will be specified in a companion document.

7. Security Considerations

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.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[VDP] IEEE 802.1 Working Group, "Edge Virtual Bridging (802.1Qbg) (work in progress)", 2012.

10.2. 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-01 (work in progress), July 2012.

[I-D.kreeger-nvo3-overlay-cp]
Kreeger, L., Dutt, D., Narten, T., Black, D., and M. Sridhavan, "Network Virtualization Overlay Control Protocol Requirements", draft-kreeger-nvo3-overlay-cp-01 (work in progress), July 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-02 (work in progress), August 2012.

[I-D.narten-nvo3-overlay-problem-statement]
Narten, T., Black, D., Dutt, D., Fang, L., Gray, E., Kreeger, L., Napierala, M., and M. Sridhavan, "Problem Statement: Overlays for Network Virtualization", draft-narten-nvo3-overlay-problem-statement-04 (work in progress), August 2012.

[I-D.sridharan-virtualization-nvgre]
Sridhavan, 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-01 (work in progress), July 2012.

[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.

[RFC6325] Perlman, R., Eastlake, D., Dutt, D., Gai, S., and A. Ghanwani, "Routing Bridges (RBridges): Base Protocol Specification", RFC 6325, July 2011.

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

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

Email: david.black@emc.com