

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: April 25, 2014

D. Black
EMC
J. Hudson
Brocade
L. Kreeger
Cisco
M. Lasserre
Alcatel-Lucent
T. Narten
IBM
October 22, 2013

**An Architecture for Overlay Networks (NV03)
draft-narten-nvo3-arch-01**

Abstract

This document presents a high-level overview architecture for building overlay networks in NV03. The architecture is given at a high-level, showing the major components of an overall system. An important goal is to divide the space into individual smaller components that can be implemented independently and with clear interfaces and interactions with other components. It should be possible to build and implement individual components in isolation and have them work with other components with no changes to other components. That way implementers have flexibility in implementing individual components and can optimize and innovate within their respective components without requiring changes to other components.

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, 2014.

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 3
- 2. Terminology 3
- 3. Background 4
 - 3.1. VN Service (L2 and L3) 5
 - 3.2. Network Virtualization Edge (NVE) 6
 - 3.3. Network Virtualization Authority (NVA) 7
 - 3.4. VM Orchestration Systems 8
- 4. Network Virtualization Edge (NVE) 9
 - 4.1. NVE Co-located With Server Hypervisor 9
 - 4.2. Split-NVE 10
 - 4.3. NVE State 11
- 5. Tenant System Types 12
 - 5.1. Overlay-Aware Network Service Appliances 12
 - 5.2. Bare Metal Servers 12
 - 5.3. Gateways 13
 - 5.4. Distributed Gateways 13
- 6. Network Virtualization Authority 14
 - 6.1. How an NVA Obtains Information 14
 - 6.2. Internal NVA Architecture 15
 - 6.3. NVA External Interface 15
- 7. NVE-to-NVA Protocol 17
 - 7.1. NVE-NVA Interaction Models 17
 - 7.2. Direct NVE-NVA Protocol 18
 - 7.3. Propagating Information Between NVEs and NVAs 19
- 8. Federated NVAs 20
 - 8.1. Inter-NVA Peering 22
- 9. Control Protocol Work Areas 23
- 10. NV03 Data Plane Encapsulation 23
- 11. Operations and Management 24
- 12. Summary 24
- 13. Acknowledgments 24

<u>14.</u> IANA Considerations	<u>24</u>
<u>15.</u> Security Considerations	<u>24</u>
<u>16.</u> Informative References	<u>24</u>
<u>Appendix A.</u> Change Log	<u>26</u>
<u>A.1.</u> Changes From -00 to -01	<u>26</u>
Authors' Addresses	<u>26</u>

1. Introduction

This document presents a high-level architecture for building overlay networks in NV03. The architecture is given at a high-level, showing the major components of an overall system. An important goal is to divide the space into smaller individual components that can be implemented independently and with clear interfaces and interactions with other components. It should be possible to build and implement individual components in isolation and have them work with other components with no changes to other components. That way implementers have flexibility in implementing individual components and can optimize and innovate within their respective components without necessarily requiring changes to other components.

The motivation for overlay networks is given in [[I-D.ietf-nvo3-overlay-problem-statement](#)]. "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. This document differs from the framework document in that it doesn't attempt to cover all possible approaches within the general design space. Rather, it describes one particular approach.

This document is intended to be a concrete strawman that can be used for discussion within the IETF NV03 WG on what the NV03 architecture should look like.

2. Terminology

This document uses the same terminology as [[I-D.ietf-nvo3-framework](#)]. In addition, the following terms are used:

NV Domain A Network Virtualization Domain is an administrative construct that defines a Network Virtualization Authority (NVA), the set of Network Virtualization Edges (NVEs) associated with that NVA, and the set of virtual networks the NVA manages and supports. NVEs are associated with a (logically centralized) NVA, and an NVE supports communication for any of the virtual networks in the domain.

NV Region A region over which information about a set of virtual networks is shared. The degenerate case of a single NV Domain corresponds to an NV region corresponding to that domain. The more interesting case occurs when two or more NV Domains share information about part or all of a set of virtual networks that they manage. Two NVAs share information about particular virtual networks for the purpose of supporting connectivity between tenants located in different NVA Domains. NVAs can share information about an entire NV domain, or just individual virtual networks.

Tenant System Identifier (TSI) Interface to a Virtual Network as presented to a Tenant System. The TSI logically connects to the NVE via a Virtual Access Point (VAP). To the Tenant System, the TSI is like a NIC; the TSI presents itself to a Tenant System as a normal network interface.

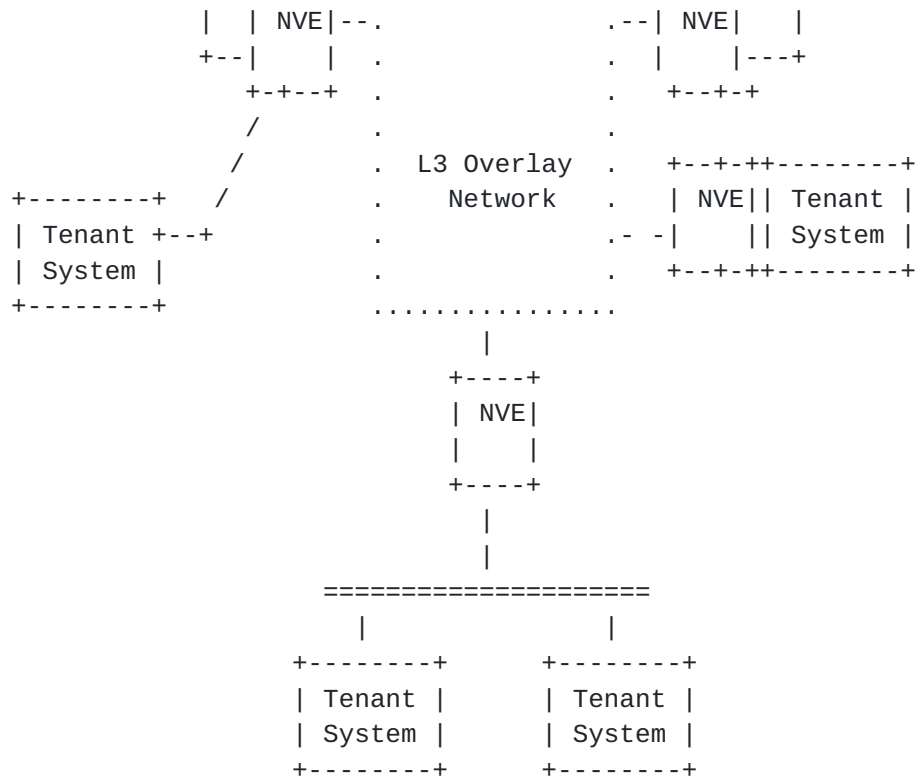
3. Background

Overlay networks are an approach for providing network virtualization services to a set of Tenant Systems (TSs) [[I-D.ietf-nvo3-framework](#)]. With overlays, data traffic between tenants is tunneled across the underlying data center's IP network. The use of tunnels provides a number of benefits by decoupling the network as viewed by tenants from the underlying physical network across which they communicate.

Tenant Systems connect to Virtual Networks (VNs), with each VN having associated attributes defining properties of the network, such as the set of members that connect to it. Tenant Systems connected to a virtual network typically communicate freely with other Tenant Systems on the same VN, but communication between Tenant Systems on one VN and those external to the VN (whether on another VN or connected to the Internet) is carefully controlled and governed by policy.

A Network Virtualization Edge (NVE) [[I-D.ietf-nvo3-framework](#)] is the entity that implements the overlay functionality. An NVE resides at the boundary between a Tenant System and the overlay network as shown in Figure 1. An NVE creates and maintains local state about each Virtual Network for which it is providing service on behalf of a Tenant System.





The dotted line indicates a network connection (i.e., IP).

Figure 1: NV03 Generic Reference Model

The following subsections describe key aspects of an overlay system in more detail. [Section 3.1](#) describes the service model (Ethernet vs. IP) provided to Tenant Systems. [Section 3.2](#) describes NVEs in more detail. [Section 3.3](#) introduces the Network Virtualization Authority, from which NVEs obtain information about virtual networks. [Section 3.4](#) provides background on VM orchestration systems and their use of virtual networks.

3.1. VN Service (L2 and L3)

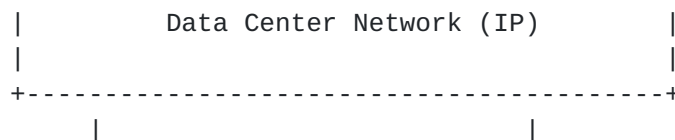
A Virtual Network provides either L2 or L3 service to connected tenants. For L2 service, VNs transport Ethernet frames, and a Tenant System is provided with a service that is analogous to being connected to a specific L2 C-VLAN. L2 broadcast frames are delivered to all (and multicast frames delivered to a subset of) the other Tenant Systems on the VN. To a Tenant System, it appears as if they are connected to a regular L2 Ethernet link. Within NV03, tenant frames are tunneled to remote NVEs based on the MAC addresses of the frame headers as originated by the Tenant System. On the underlay, NV03 packets are forwarded between NVEs based on the outer addresses of tunneled packets.

For L3 service, VNs transport IP datagrams, and a Tenant System is provided with a service that only supports IP traffic. Within NV03, tenant frames are tunneled to remote NVEs based on the IP addresses of the packet originated by the Tenant System; any L2 destination addresses provided by Tenant Systems are effectively ignored.

L2 service is intended for systems that need native L2 Ethernet service and the ability to run protocols directly over Ethernet (i.e., not based on IP). L3 service is intended for systems in which all the traffic can safely be assumed to be IP. It is important to note that whether NV03 provides L2 or L3 service to a Tenant System, the Tenant System does not generally need to be aware of the distinction. In both cases, the virtual network presents itself to the Tenant System as an L2 Ethernet interface. An Ethernet interface is used in both cases simply as a widely supported interface type that essentially all Tenant Systems already support. Consequently, no special software is needed on Tenant Systems to use an L3 vs. an L2 overlay service.

3.2. Network Virtualization Edge (NVE)

Tenant Systems connect to NVEs via a Tenant System Interface (TSI). The TSI logically connects to the NVE via a Virtual Access Point (VAP) as shown in Figure 2. To the Tenant System, the TSI is like a NIC; the TSI presents itself to a Tenant System as a normal network interface. On the NVE side, a VAP is a logical network port (virtual or physical) into a specific virtual network. Note that two different Tenant Systems (and TSIs) attached to a common NVE can share a VAP (e.g., TS1 and TS2 in Figure 2) so long as they connect to the same Virtual Network.



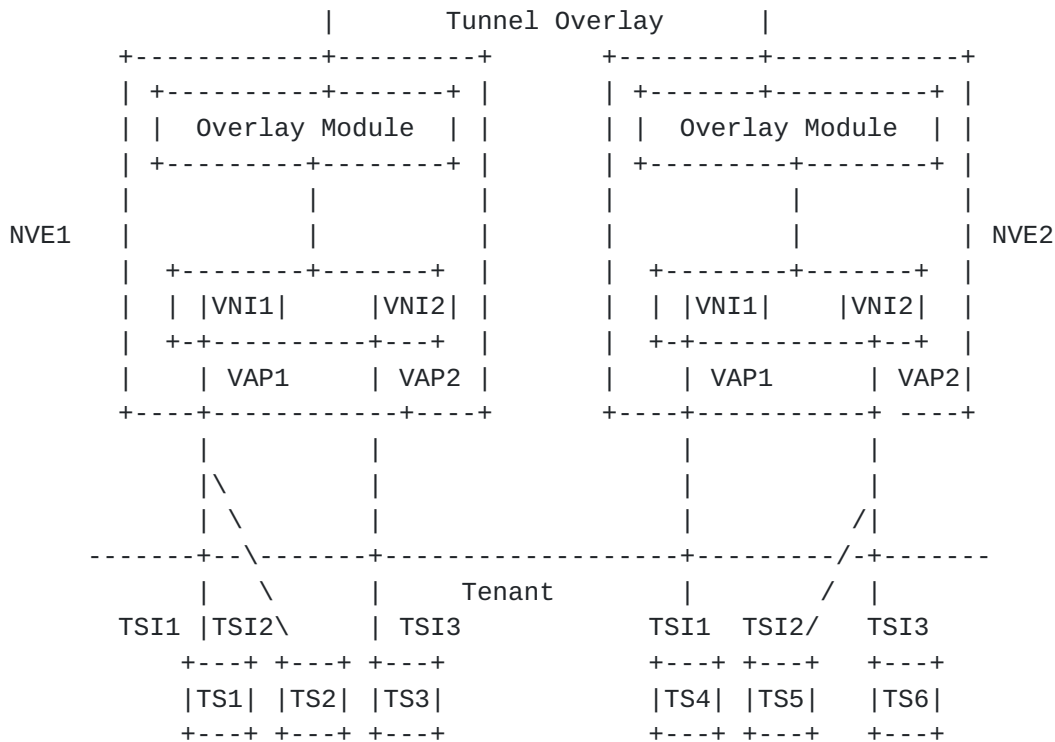


Figure 2: NVE Reference Model

The Overlay Module performs the actual encapsulation and decapsulation of tunneled packets. The NVE maintains state about the virtual networks it is a part of so that it can provide the Overlay Module with such information as the destination address of the NVE to tunnel a packet to, or the Context ID that should be placed in the encapsulation header to identify the virtual network a tunneled packet belong to.

On the data center network side, the NVE sends and receives native IP traffic. When ingressing traffic from a Tenant System, the NVE identifies the egress NVE to which the packet should be sent, adds an overlay encapsulation header, and sends the packet on the underlay network. When receiving traffic from a remote NVE, an NVE strips off the encapsulation header, and delivers the (original) packet to the appropriate Tenant System.

Conceptually, the NVE is a single entity implementing the NV03 functionality. In practice, there are a number of different implementation scenarios, as described in detail in [Section 4](#).

3.3. Network Virtualization Authority (NVA)

Address dissemination refers to the process of learning, building and distributing the mapping/forwarding information that NVEs need in

order to tunnel traffic to each other on behalf of communicating Tenant Systems. For example, in order to send traffic to a remote Tenant System, the sending NVE must know the destination NVE for that Tenant System.

One way to build and maintain mapping tables is to use learning, as 802.1 bridges do [[IEEE-802.1Q](#)]. When forwarding traffic to multicast or unknown unicast destinations, an NVE could simply flood traffic everywhere. While flooding works, it can lead to traffic hot spots and can lead to problems in larger networks.

Alternatively, NVEs can make use of a Network Virtualization Authority (NVA). An NVA is the entity that provides address mapping and other information to NVEs. NVEs interact with an NVA to obtain any required address mapping information they need in order to properly forward traffic on behalf of tenants. The term NVA refers to the overall system, without regards to its scope or how it is implemented. NVAs provide a service, and NVEs access that service via an NVE-to-NVA protocol.

Even when an NVA is present, learning could be used as a fallback mechanism, should the NVA be unable to provide an answer or for other reasons. This document does not consider flooding approaches in detail, as there are a number of benefits in using an approach that depends on the presence of an NVA.

NVAs are discussed in more detail in [Section 6](#).

3.4. VM Orchestration Systems

VM Orchestration systems manage server virtualization across a set of servers. Although VM management is a separate topic from network virtualization, the two areas are closely related. Managing the creation, placement, and movements of VMs also involves creating, attaching to and detaching from virtual networks. A number of existing VM orchestration systems have incorporated aspects of virtual network management into their systems.

When a new VM image is started, the VM Orchestration system determines where the VM should be placed, interacts with the hypervisor on the target server to load and start the server and controls when a VM should be shutdown or migrated elsewhere. VM Orchestration systems also have knowledge about how a VM should connect to a network, possibly including the name of the virtual network to which a VM is to connect. The VM orchestration system can pass such information to the hypervisor when a VM is instantiated. VM orchestration systems have significant (and sometimes global) knowledge over the domain they manage. They typically know on what

servers a VM is running, and meta data associated with VM images can be useful from a network virtualization perspective. For example, the meta data may include the addresses (MAC and IP) the VMs will use and the name(s) of the virtual network(s) they connect to.

VM orchestration systems run a protocol with an agent running on the hypervisor of the servers they manage. That protocol can also carry information about what virtual network a VM is associated with. When the orchestrator instantiates a VM on a hypervisor, the hypervisor interacts with the NVE in order to attach the VM to the virtual networks it has access to. In general, the hypervisor will need to communicate significant VM state changes to the NVE. In the reverse direction, the NVE may need to communicate network connectivity information back to the hypervisor. Example VM orchestration systems in use today include VMware's vCenter Server or Microsoft's System Center Virtual Machine Manager. Both can pass information about what virtual networks a VM connects to down to the hypervisor. The protocol used between the VM orchestration system and hypervisors is generally proprietary.

It should be noted that VM orchestration systems may not have direct access to all networking related information a VM uses. For example, a VM may make use of additional IP or MAC addresses that the VM management system is not aware of.

4. Network Virtualization Edge (NVE)

As introduced in [Section 3.2](#) an NVE is the entity that implements the overlay functionality. This section describes NVEs in more detail. An NVE will have two external interfaces:

Tenant Facing: On the tenant facing side, an NVE interacts with the hypervisor (or equivalent entity) to provide the NVO3 service. An NVE will need to be notified when a Tenant System "attaches" to a virtual network (so it can validate the request and set up any state needed to send and receive traffic on behalf of the Tenant System on that VN). Likewise, an NVE will need to be informed when the Tenant System "detaches" from the virtual network so that it can reclaim state and resources appropriately.

DCN Facing: On the data center network facing side, an NVE interfaces with the data center underlay network, sending and receiving tunneled IP packets to and from the underlay. The NVE may also run a control protocol with other entities on the network, such as the Network Virtualization Authority.

4.1. NVE Co-located With Server Hypervisor

When server virtualization is used, the entire NVE functionality will typically be implemented as part of the hypervisor and/or virtual switch on the server. In such cases, the Tenant System interacts with the hypervisor and the hypervisor interacts with the NVE. Because the interaction between the hypervisor and NVE is implemented entirely in software on the server, there is no "on-the-wire" protocol between Tenant Systems (or the hypervisor) and the NVE that needs to be standardized. While there may be APIs between the NVE and hypervisor to support necessary interaction, the details of such an API are not in-scope for the IETF to work on.

Implementing NVE functionality entirely on a server has the disadvantage that server CPU resources must be spent implementing the NV03 functionality. Experimentation with overlay approaches and previous experience with TCP and checksum adapter offloads suggests that offloading certain NVE operations (e.g., encapsulation and decapsulation operations) onto the physical network adaptor can produce performance improvements. As has been done with checksum and /or TCP server offload and other optimization approaches, there may be benefits to offloading common operations onto adaptors where possible. Just as important, the addition of an overlay header can disable existing adaptor offload capabilities that are generally not prepared to handle the addition of a new header or other operations associated with an NVE.

While the details of how to split the implementation of specific NVE functionality between a server and its network adaptors is outside the scope of IETF standardization, the NV03 architecture should support such separation. Ideally, it may even be possible to bypass the hypervisor completely on critical data path operations so that packets between a TS and its VN can be sent and received without having the hypervisor involved in each individual packet operation.

4.2. Split-NVE

Another possible scenario leads to the need for a split NVE implementation. A hypervisor running on a server could be aware that NV03 is in use, but have some of the actual NV03 functionality implemented on an adjacent switch to which the server is attached. While one could imagine a number of link types between a server and the NVE, the simplest deployment scenario would involve a server and NVE separated by a simple L2 Ethernet link, across which LLDP runs. A more complicated scenario would have the server and NVE separated by a bridged access network, such as when the NVE resides on a ToR, with an embedded switch residing between servers and the ToR.

While the above talks about a scenario involving a hypervisor, it should be noted that the same scenario can apply to Network Service

Appliances as discussed in [Section 5.1](#). In general, when this document discusses the interaction between a hypervisor and NVE, the discussion applies to Network Service Appliances as well.

For the split NVE case, protocols will be needed that allow the hypervisor and NVE to negotiate and setup the necessary state so that traffic sent across the access link between a server and the NVE can be associated with the correct virtual network instance. Specifically, on the access link, traffic belonging to a specific Tenant System would be tagged with a specific VLAN C-TAG that identifies which specific NVO3 virtual network instance it belongs to. The hypervisor-NVE protocol would negotiate which VLAN C-TAG to use for a particular virtual network instance. More details of the protocol requirements for functionality between hypervisors and NVEs can be found in [[I-D.kreeger-nvo3-hypervisor-nve-cp](#)].

4.3. NVE State

NVEs maintain internal data structures and state to support the sending and receiving of tenant traffic. An NVE may need some or all of the following information:

1. An NVE keeps track of which attached Tenant Systems are connected to which virtual networks. When a Tenant System attaches to a virtual network, the NVE will need to create or update local state for that virtual network. When the last Tenant System detaches from a given VN, the NVE can reclaim state associated with that VN.
2. For tenant unicast traffic, an NVE maintains a per-VN table of mappings from Tenant System (inner) addresses to remote NVE (outer) addresses.
3. For tenant multicast (or broadcast) traffic, an NVE maintains a per-VN table of mappings and other information on how to deliver multicast (or broadcast) traffic. If the underlying network supports IP multicast, the NVE could use IP multicast to deliver tenant traffic. In such a case, the NVE would need to know what IP underlay multicast address to use for a given VN. Alternatively, if the underlying network does not support multicast, an NVE could use serial unicast to deliver traffic. In such a case, an NVE would need to know which destinations are subscribers to the tenant multicast group. An NVE could use both approaches, switching from one mode to the other depending on such factors as bandwidth efficiency and group membership sparseness.

4. An NVE maintains necessary information to encapsulate outgoing traffic, including what type of encapsulation and what value to use for a Context ID within the encapsulation header.
5. In order to deliver incoming encapsulated packets to the correct Tenant Systems, an NVE maintains the necessary information to map incoming traffic to the appropriate VAP and Tenant System.
6. An NVE may find it convenient to maintain additional per-VN information such as QoS settings, Path MTU information, ACLs, etc.

5. Tenant System Types

This section describes a number of special Tenant System types and how they fit into an NV03 system.

5.1. Overlay-Aware Network Service Appliances

Some Network Service Appliances [[I-D.ietf-nvo3-nve-nva-cp-req](#)] (virtual or physical) provide tenant-aware services. That is, the specific service they provide depends on the identity of the tenant making use of the service. For example, firewalls are now becoming available that support multi-tenancy where a single firewall provides virtual firewall service on a per-tenant basis, using per-tenant configuration rules and maintaining per-tenant state. Such appliances will be aware of the VN an activity corresponds to while processing requests. Unlike server virtualization, which shields VMs from needing to know about multi-tenancy, a Network Service Appliance explicitly supports multi-tenancy. In such cases, the Network Service Appliance itself will be aware of network virtualization and either embed an NVE directly, or implement a split NVE as described in [Section 4.2](#). Unlike server virtualization, however, the Network Service Appliance will not be running a traditional hypervisor and the VM Orchestration system may not interact with the Network Service Appliance. The NVE on such appliances will need to support a control plane to obtain the necessary information needed to fully participate in an NV03 Domain.

5.2. Bare Metal Servers

Many data centers will continue to have at least some servers operating as non-virtualized (or "bare metal") machines running a traditional operating system and workload. In such systems, there will be no NVE functionality on the server, and the server will have no knowledge of NV03 (including whether overlays are even in use). In such environments, the NVE functionality can reside on the first-hop physical switch. In such a case, the network administrator would

(manually) configure the switch to enable the appropriate NV03 functionality on the switch port connecting the server and associate that port with a specific virtual network. Such configuration would typically be static, since the server is not virtualized, and once configured, is unlikely to change frequently. Consequently, this scenario does not require any protocol or standards work.

5.3. Gateways

Gateways on VNs relay traffic onto and off of a virtual network. Tenant Systems use gateways to reach destinations outside of the local VN. Gateways receive encapsulated traffic from one VN, remove the encapsulation header, and send the native packet out onto the data center network for delivery. Outside traffic enters a VN in a reverse manner.

Gateways can be either virtual (i.e., implemented as a VM) or physical (i.e., as a standalone physical device). For performance reasons, standalone hardware gateways may be desirable in some cases. Such gateways could consist of a simple switch forwarding traffic from a VN onto the local data center network, or could embed router functionality. On such gateways, network interfaces connecting to virtual networks will (at least conceptually) embed NVE (or split-NVE) functionality within them. As in the case with Network Service Appliances, gateways will not support a hypervisor and will need an appropriate control plane protocol to obtain the information needed to provide NV03 service.

Gateways handle several different use cases. For example, a virtual network could consist of systems supporting overlays together with legacy Tenant Systems that do not. Gateways could be used to connect legacy systems supporting, e.g., L2 VLANs, to specific virtual networks, effectively making them part of the same virtual network. Gateways could also forward traffic between a virtual network and other hosts on the data center network or relay traffic between different VNs. Finally, gateways can provide external connectivity such as Internet or VPN access.

5.4. Distributed Gateways

The relaying of traffic from one VN to another deserves special consideration. The previous section described gateways performing this function. If such gateways are centralized, traffic between TSeS on different VNs can take suboptimal paths, i.e., triangular routing results in paths that always traverse the gateway. As an optimization, individual NVEs can be part of a distributed gateway that performs such relaying, reducing or completely eliminating triangular routing. In a distributed gateway, each ingress NVE can

perform such relaying activity directly, so long as it has access to the policy information needed to determine whether cross-VN communication is allowed. Having individual NVEs be part of a distributed gateway allows them to tunnel traffic directly to the destination NVE without the need to take suboptimal paths.

The NVO3 architecture should [must? or just say it does?] support distributed gateways. Such support requires that NVO3 control protocols include mechanisms for the maintenance and distribution of policy information about what type of cross-VN communication is allowed so that NVEs acting as distributed gateways can tunnel traffic from one VN to another as appropriate.

6. Network Virtualization Authority

Before sending to and receiving traffic from a virtual network, an NVE must obtain the information needed to build its internal forwarding tables and state as listed in [Section 4.3](#). An NVE obtains such information from a Network Virtualization Authority.

The Network Virtualization Authority (NVA) is the entity that provides address mapping and other information to NVEs. NVEs interact with an NVA to obtain any required information they need in order to properly forward traffic on behalf of tenants. The term NVA refers to the overall system, without regards to its scope or how it is implemented.

6.1. How an NVA Obtains Information

There are two primary ways in which an NVA can obtain the address dissemination information it manages. The NVA can obtain information either from the VM orchestration system, or directly from the NVEs themselves.

On virtualized systems, the NVA may be able to obtain the address mapping information associated with VMs from the VM orchestration system itself. If the VM orchestration system contains a master database for all the virtualization information, having the NVA obtain information directly to the orchestration system would be a natural approach. Indeed, the NVA could effectively be co-located with the VM orchestration system itself. In such systems, the VM orchestration system communicates with the NVE indirectly through the hypervisor.

However, as described in [Section 4](#) not all NVEs are associated with hypervisors. In such cases, NVAs cannot leverage VM orchestration protocols to interact with an NVE and will instead need to peer directly with them. By peering directly with an NVE, NVAs can obtain

information about the TSeS connected to that NVE and can distribute information to the NVE about the VN's those TSeS are associated with. For example, whenever a Tenant System attaches to an NVE, that NVE would notify the NVA that the TS is now associated with that NVE. Likewise when a TS detaches from an NVE, that NVE would inform the NVA. By communicating directly with NVEs, both the NVA and the NVE are able to maintain up-to-date information about all active tenants and the NVEs to which they are attached.

6.2. Internal NVA Architecture

For reliability and fault tolerance reasons, an NVA would be implemented in a distributed or replicated manner without single points of failure. How the NVA is implemented, however, is not important to an NVE so long as the NVA provides a consistent and well-defined interface to the NVE. For example, an NVA could be implemented via database techniques whereby a server stores address mapping information in a traditional (possibly replicated) database. Alternatively, an NVA could be implemented in a distributed fashion using an existing (or modified) routing protocol to maintain and distribute mappings. So long as there is a clear interface between the NVE and NVA, how an NVA is architected and implemented is not important to an NVE.

A number of architectural approaches could be used to implement NVAs themselves. NVAs manage address bindings and distribute them to where they need to go. One approach would be to use BGP (possibly with extensions) and route reflectors. Another approach could use a transaction-based database model with replicated servers. Because the implementation details are local to an NVA, there is no need to pick exactly one solution technology, so long as the external interfaces to the NVEs (and remote NVAs) are sufficiently well defined to achieve interoperability.

6.3. NVA External Interface

[note: the following section discusses various options that the WG has not yet expressed an opinion on. Discussion is encouraged.]

Conceptually, from the perspective of an NVE, an NVA is a single entity. An NVE interacts with the NVA, and it is the NVA's responsibility for ensuring that interactions between the NVE and NVA result in consistent behavior across the NVA and all other NVEs using the same NVA. Because an NVA is built from multiple internal components, an NVA will have to ensure that information flows to all internal NVA components appropriately.

One architectural question is how the NVA presents itself to the NVE. For example, an NVA could be required to provide access via a single IP address. If NVEs only have one IP address to interact with, it would be the responsibility of the NVA to handle NVA component failures, e.g., by using a "floating IP address" that migrates among NVA components to ensure that the NVA can always be reached via the one address. Having all NVA accesses through a single IP address, however, adds constraints to implementing robust failover, load balancing, etc.

[Note: the following is a strawman proposal.]

In the NVO3 architecture, an NVA is accessed through one or more IP addresses (i.e. IP address/port combination). If multiple IP addresses are used, each IP address provides equivalent functionality, meaning that an NVE can use any of the provided addresses to interact with the NVA. Should one address stop working, an NVE is expected to failover to another. While the different addresses result in equivalent functionality, one address may be more responsive than another, e.g., due to network conditions, load on the server, etc.

[Note: should we support the following?] To provide some control over load balancing, NVA addresses may have an associated priority. Addresses are used in order of priority, with no explicit preference among NVA addresses having the same priority. To provide basic load-balancing among NVAs of equal priorities, NVEs use some randomization input to select among equal-priority NVAs. Such a priority scheme facilitates failover and load balancing, for example, allowing a network operator to specify a set of primary and backup NVAs.

[note: should we support the following? It would presumably add considerable complexity to the NVE.] It may be desirable to have individual NVA addresses responsible for a subset of information about an NV Domain. In such a case, NVEs would use different NVA addresses for obtaining or updating information about particular VNs or TS bindings. A key question with such an approach is how information would be partitioned, and how an NVE could determine which address to use to get the information it needs.

Another possibility is to treat the information on which NVA addresses to use as cached (soft-state) information at the NVEs, so that any NVA address can be used to obtain any information, but NVEs are informed of preferences for which addresses to use for particular information on VNs or TS bindings. That preference information would be cached for future use to improve behavior - e.g., if all requests for a specific subset of VNs are forwarded to a specific NVA component, the NVE can optimize future requests within that subset by sending them directly to that NVA component via its address.

7. NVE-to-NVA Protocol

[Note: this and later sections are a bit sketchy and need work. Discussion is encouraged.]

As outlined in [Section 4.3](#), an NVE needs certain information in order to perform its functions. To obtain such information from an NVA, an NVE-to-NVA protocol is needed. The NVE-to-NVA protocol provides two functions. First it allows an NVE to obtain information about the location and status of other TSEs with which it needs to communicate. Second, the NVE-to-NVA protocol provides a way for NVEs to provide updates to the NVA about the TSEs attached to that NVE (e.g., when a TS attaches or detaches from the NVE), or about communication errors encountered when sending traffic to remote NVEs. For example, an NVE could indicate that a destination it is trying to reach at a destination NVE is unreachable for some reason.

While having a direct NVE-to-NVA protocol might seem straightforward, the existence of existing VM orchestration systems complicates the choices an NVE has for interacting with the NVA.

7.1. NVE-NVA Interaction Models

An NVE interacts with an NVA in at least two (quite different) ways:

- o NVEs supporting VMs and hypervisors can obtain necessary information entirely through the hypervisor-facing side of the NVE. Such an approach is a natural extension to existing VM orchestration systems supporting server virtualization because an existing protocol between the hypervisor and VM Orchestration system already exists and can be leveraged to obtain any needed information. Specifically, VM orchestration systems used to create, terminate and migrate VMs already use well-defined (though typically proprietary) protocols to handle the interactions between the hypervisor and VM orchestration system. For such systems, it is a natural extension to leverage the existing orchestration protocol as a sort of proxy protocol for handling the interactions between an NVE and the NVA. Indeed, existing implementation already do this.
- o Alternatively, an NVE can obtain needed information by interacting directly with an NVA via a protocol operating over the data center underlay network. Such an approach is needed to support NVEs that are not associated with systems performing server virtualization (e.g., as in the case of a standalone gateway) or where the NVE needs to communicate directly with the NVA for other reasons.

[Note: The following paragraph is included to stimulate discussion, and the WG will need to decide what direction it wants to take.]

The WG The NV03 architecture should support both of the above models, as in practice, it is likely that both models will coexist in practice and be used simultaneously in a deployment. Existing virtualization environments are already using the first model. But they are not sufficient to cover the case of standalone gateways -- such gateways do not support virtualization and do not interface with existing VM orchestration systems. Also, A hybrid approach might be desirable in some cases where the first model is used to obtain the information, but the latter approach is used to validate and further authenticate the information before using it.

7.2. Direct NVE-NVA Protocol

An NVE can interact directly with an NVA via an NVE-to-NVA protocol. Such a protocol can be either independent of the NVA internal protocol, or an extension of it. Using a dedicated protocol provides architectural separation and independence between the NVE and NVA. The NVE and NVA interact in a well-defined way, and changes in the NVA (or NVE) do not need to impact each other. Using a dedicated protocol also ensures that both NVE and NVA implementations can evolve independently and without dependencies on each other. Such independence is important because the upgrade path for NVEs and NVAs is quite different. Upgrading all the NVEs at a site will likely be

more difficult in practice than upgrading NVAs because of their large number - one on each end device. In practice, it is assumed that an NVE will be implemented once, and then (hopefully) not again, whereas an NVA (and its associated protocols) are more likely to evolve over time as experience is gained from usage.

Requirements for a direct NVE-NVA protocol can be found in [[I-D.ietf-nvo3-nve-nva-cp-req](#)]

7.3. Propagating Information Between NVEs and NVAs

[Note: This section has been completely redone to move away from the push/pull discussion at an abstract level.]

Information flows between NVEs and NVAs in both directions. The NVA maintains information about all VNs in the NV Domain, so that NVEs do not need to do so themselves. NVEs obtain from the NVA information about where a given remote TS destination resides. NVAs in turn obtain information from NVEs about the individual TSs attached to those NVEs.

While the NVA could push information about every virtual network to every NVE, such an approach scales poorly and is unnecessary. In practice, a given NVE will only need and want to know about VNs to which it is attached. Thus, an NVE should be able to subscribe to updates only for the virtual networks it is interested in receiving updates for. The NVO3 architecture supports a model where an NVE is not required to have full mapping tables for all virtual networks in an NV Domain.

Before sending unicast traffic to a remote TS, an NVE must know where the remote TS currently resides. When a TS attaches to a virtual network, the NVE obtains information about that VN from the NVA. The NVA can provide that information to the NVE at the time the TS attaches to the VN, either because the NVE requests the information when the attach operation occurs, or because the VM orchestration system has initiated the attach operation and provides associated mapping information to the NVE at the same time. A similar process can take place with regards to obtaining necessary information needed for delivery of tenant broadcast or multicast traffic.

There are scenarios where an NVE may wish to query the NVA about individual mappings within an VN. For example, when sending traffic to a remote TS on a remote NVE, that TS may become unavailable (e.g., because it has migrated elsewhere or has been shutdown, in which case the remote NVE may return an error indication). In such situations, the NVE may need to query the NVA to obtain updated mapping information for a specific TS, or verify that the information is

still correct despite the error condition. Note that such a query could also be used by the NVA as an indication that there may be an inconsistency in the network and that it should take steps to verify that the information it has about the current state and location of a specific TS is still correct.

For very large virtual networks, the amount of state an NVE needs to maintain for a given virtual network could be significant. Moreover, an NVE may only be communicating with a small subset of the TSEs on such a virtual network. In such cases, the NVE may find it desirable to maintain state only for those destinations it is actively communicating with. In such scenarios, an NVE may not want to maintain full mapping information about all destinations on a VN. Should it then need to communicate with a destination for which it does not have mapping information, however, it will need to be able to query the NVA on demand for the missing information on a per-destination basis.

The NV03 architecture will need to support a range of operations between the NVE and NVA. Requirements for those operations can be found in [\[I-D.ietf-nvo3-nve-nva-cp-req\]](#).

8. Federated NVAs

An NVA provides service to the set of NVEs in its NV Domain. Each NVA manages network virtualization information for the virtual networks within its NV Domain. An NV domain is administered by a single entity.

In some cases, it will be necessary to expand the scope of a specific VN or even an entire NV domain beyond a single NVA. For example, multiple data centers managed by the same administrator may wish to operate all of its data centers as a single NV region. Such cases are handled by having different NVAs peer with each other to exchange mapping information about specific VNs. NVAs operate in a federated manner with a set of NVAs operating as a loosely-coupled federation of individual NVAs. If a virtual network spans multiple NVAs (e.g., located at different data centers), and an NVE needs to deliver tenant traffic to an NVE at a remote NVA, it still interacts only with its NVA, even when obtaining mappings for NVEs associated with domains at a remote NVA.

Figure Figure 3 shows a scenario where two separate NV Domains (1 and 2) share information about Virtual Network "1217". VM1 and VM1 both connect to the same Virtual Network (1217), even though the two VMs are in separate NV Domains. There are two cases to consider. In the first case, NV Domain B (NVB) does not allow NVE-A to tunnel traffic directly to NVE-B. There could be a number of reasons for this. For

example, NV Domains 1 and 2 may not share a common address space (i.e., require traversal through a NAT device), or for policy reasons, a domain might require that all traffic between separate NV Domains be funneled through a particular device (e.g., a firewall). In such cases, NVA-2 will advertise to NVA-1 that VM1 on virtual network 1217 is available, and direct that traffic between the two nodes go through IP-G. IP-G would then decapsulate received traffic from one NV Domain, translate it appropriately for the other domain and re-encapsulate the packet for delivery.

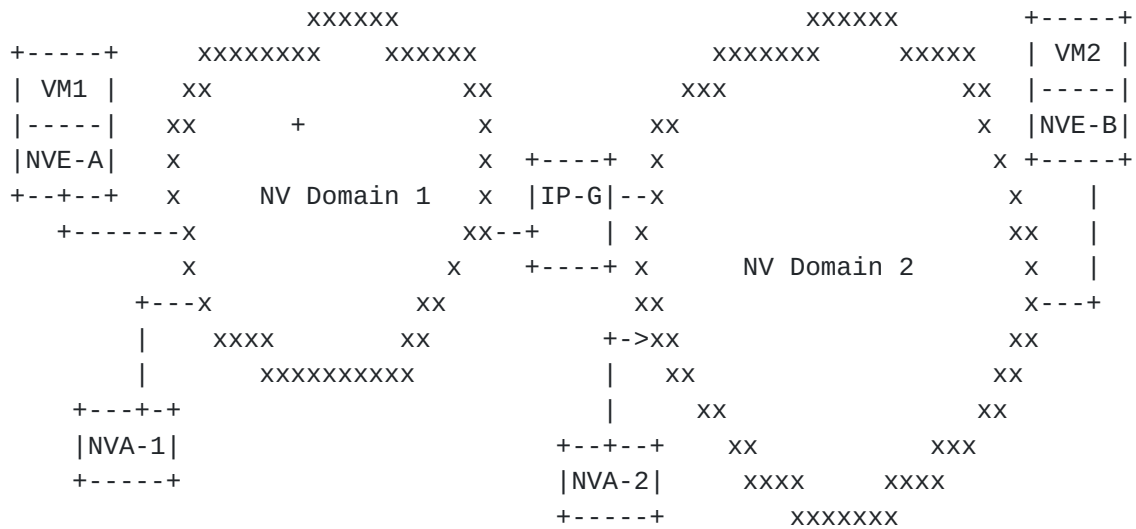


Figure 3: VM1 and VM2 are in different NV Domains.

NVAs at one site share information and interact with NVAs at other sites, but only in a controlled manner. It is expected that policy and access control will be applied at the boundaries between different sites (and NVAs) so as to minimize dependencies on external NVAs that could negatively impact the operation within a site. It is an architectural principle that operations involving NVAs at one site not be immediately impacted by failures or errors at another site. (Of course, communication between NVEs in different NVO3 domains may be impacted by such failures or errors.) It is a strong requirement that an NVA continue to operate properly for local NVEs even if external communication is interrupted (e.g., should communication between a local and remote NVA fail).

At a high level, a federation of interconnected NVAs has some analogies to BGP and Autonomous Systems. Like an Autonomous System, NVAs at one site are managed by a single administrative entity and do not interact with external NVAs except as allowed by policy. Likewise, the interface between NVAs at different sites is well defined, so that the internal details of operations at one site are largely hidden to other sites. Finally, an NVA only peers with other

NVAs that it has a trusted relationship with, i.e., where a virtual network is intended to span multiple NVAs.

[Note: the following are motivations for having a federated NVA model and are intended for discussion. Depending on discussion, these may be removed from future versions of this document.] Reasons for using a federated model include:

- o Provide isolation between NVAs operating at different sites at different geographic locations.
- o Control the quantity and rate of information updates that flow (and must be processed) between different NVAs in different data centers.
- o Control the set of external NVAs (and external sites) a site peers with. A site will only peer with other sites that are cooperating in providing an overlay service.
- o Allow policy to be applied between sites. A site will want to carefully control what information it exports (and to whom) as well as what information it is willing to import (and from whom).
- o Allow different protocols and architectures to be used to for intra- vs. inter-NVA communication. For example, within a single data center, a replicated transaction server using database techniques might be an attractive implementation option for an NVA, and protocols optimized for intra-NVA communication would likely be different from protocols involving inter-NVA communication between different sites.
- o Allow for optimized protocols, rather than using a one-size-fits all approach. Within a data center, networks tend to have lower-latency, higher-speed and higher redundancy when compared with WAN links interconnecting data centers. The design constraints and tradeoffs for a protocol operating within a data center network are different from those operating over WAN links. While a single protocol could be used for both cases, there could be advantages to using different and more specialized protocols for the intra- and inter-NVA case.

8.1. Inter-NVA Peering

To support peering between different NVAs, an inter-NVA protocol is needed. The inter-NVA protocol defines what information is exchanged between NVAs. It is assumed that the protocol will be used to share addressing information between data centers and must scale well over WAN links.

9. Control Protocol Work Areas

The NV03 architecture consists of two major distinct entities: NVEs and NVAs. In order to provide isolation and independence between these two entities, the NV03 architecture calls for well defined protocols for interfacing between them. For an individual NVA, the architecture calls for a single conceptual entity, that could be implemented in a distributed or replicated fashion. While the IETF may choose to define one or more specific architectural approaches to building individual NVAs, there is little need for it to pick exactly one approach to the exclusion of others. An NVA for a single domain will likely be deployed as a single vendor product and thus their is little benefit in standardizing the internal structure of an NVA.

Individual NVAs peer with each other in a federated manner. The NV03 architecture calls for a well-defined interface between NVAs.

Finally, a hypervisor-to-NVE protocol is needed to cover the split-NVE scenario described in [Section 4.2](#).

10. NV03 Data Plane Encapsulation

When tunneling tenant traffic, NVEs add encapsulation header to the original tenant packet. The exact encapsulation to use for NV03 does not seem to be critical. The main requirement is that the encapsulation support a Context ID of sufficient size [[I-D.ietf-nvo3-dataplane-requirements](#)]. A number of encapsulations already exist that provide a VN Context of sufficient size for NV03. For example, VXLAN [[I-D.mahalingam-dutt-dcops-vxlan](#)] has a 24-bit VXLAN Network Identifier (VNI). NVGRE [[I-D.sridharan-virtualization-nvgre](#)] has a 24-bit Tenant Network ID (TNI). MPLS-over-GRE provides a 20-bit label field. While there is widespread recognition that a 12-bit VN Context would be too small (only 4096 distinct values), it is generally agreed that 20 bits (1 million distinct values) and 24 bits (16.8 million distinct values) are sufficient for a wide variety of deployment scenarios.

[Note: the following paragraph is included for WG discussion. Future versions of this document may omit this text.]

While one might argue that a new encapsulation should be defined just for NV03, no compelling requirements for doing so have been identified yet. Moreover, optimized implementations for existing encapsulations are already starting to become available on the market (i.e., in silicon). If the IETF were to define a new encapsulation format, it would take at least 2 (and likely more) years before optimized implementations of the new format would become available in products. In addition, a new encapsulation format would not likely

displace existing formats, at least not for years. Thus, there seems little reason to define a new encapsulation. However, it does make sense for NVO3 to support multiple encapsulation formats, so as to allow NVEs to use their preferred encapsulations when possible. This implies that the address dissemination protocols must also include an indication of supported encapsulations along with the address mapping details.

11. Operations and Management

The simplicity of operating and debugging overlay networks will be critical for successful deployment. Some architectural choices can facilitate or hinder OAM. Related OAM drafts include [[I-D.ashwood-nvo3-operational-requirement](#)].

12. Summary

This document provides a start at a general architecture for overlays in NVO3. The architecture calls for three main areas of protocol work:

1. A hypervisor-to-NVE protocol to support Split NVEs as discussed in [Section 4.2](#).
2. An NVE to NVA protocol for address dissemination.
3. An NVA-to-NVA protocol for exchange of information about specific virtual networks between NVAs.

It should be noted that existing protocols or extensions of existing protocols are applicable.

13. Acknowledgments

Helpful comments and improvements to this document have come from Lizhong Jin, Dennis (Xiaohong) Qin and Lucy Yong.

14. IANA Considerations

This memo includes no request to IANA.

15. Security Considerations

Yep, kind of sparse. But we'll get there eventually. :-)

16. Informative References

[[I-D.ashwood-nvo3-operational-requirement](#)]

Ashwood-Smith, P., Iyengar, R., Tsou, T., Sajassi, A., Boucadair, M., Jacquenet, C., and M. Daikoku, "NV03 Operational Requirements", [draft-ashwood-nvo3-operational-requirement-03](#) (work in progress), July 2013.

[I-D.ietf-nvo3-dataplane-requirements]

Bitar, N., Lasserre, M., Balus, F., Morin, T., Jin, L., and B. Khasnabish, "NV03 Data Plane Requirements", [draft-ietf-nvo3-dataplane-requirements-01](#) (work in progress), July 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-03](#) (work in progress), July 2013.

[I-D.ietf-nvo3-nve-nva-cp-req]

Kreeger, L., Dutt, D., Narten, T., and D. Black, "Network Virtualization NVE to NVA Control Protocol Requirements", [draft-ietf-nvo3-nve-nva-cp-req-00](#) (work in progress), July 2013.

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

Narten, T., Gray, E., Black, D., Fang, L., Kreeger, L., and M. Napierala, "Problem Statement: Overlays for Network Virtualization", [draft-ietf-nvo3-overlay-problem-statement-04](#) (work in progress), July 2013.

[I-D.kreeger-nvo3-hypervisor-nve-cp]

Kreeger, L., Narten, T., and D. Black, "Network Virtualization Hypervisor-to-NVE Overlay Control Protocol Requirements", [draft-kreeger-nvo3-hypervisor-nve-cp-01](#) (work in progress), February 2013.

[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-05](#) (work in progress), October 2013.

[I-D.sridharan-virtualization-nvgre]

Sridharan, M., Greenberg, A., Wang, Y., Garg, P., Venkataramiah, N., Duda, K., Ganga, I., Lin, G., Pearson, M., Thaler, P., and C. Tumuluri, "NVGRE: Network Virtualization using Generic Routing Encapsulation", [draft-sridharan-virtualization-nvgre-03](#) (work in progress), August 2013.

[IEEE-802.1Q]

IEEE 802.1Q-2011, ., "IEEE standard for local and metropolitan area networks: Media access control (MAC) bridges and virtual bridged local area networks, ", August 2011.

Appendix A. Change Log

A.1. Changes From -00 to -01

1. Editorial and clarity improvements.
2. Replaced "push vs. pull" section with section more focussed on triggers where an event implies or triggers some action.
3. Clarified text on co-located NVE to show how offloading NVE functionality onto adaptors is desirable.
4. Added new section on distributed gateways.
5. Expanded Section on NVA external interface, adding requirement for NVE to support multiple IP NVA addresses.

Authors' Addresses

David Black
EMC

Email: david.black@emc.com

Jon Hudson
Brocade
120 Holger Way
San Jose, CA 95134
USA

Email: jon.hudson@gmail.com

Lawrence Kreeger
Cisco

Email: kreeger@cisco.com

Marc Lasserre
Alcatel-Lucent

Email: marc.lasserre@alcatel-lucent.com

Thomas Narten
IBM

Email: narten@us.ibm.com