An Architecture for
Data-Center Network Virtualization over Layer 3 (NVO3)

Abstract

   This document presents a high-level overview architecture for
   building data-center Network Virtualization over Layer 3 (NVO3)
   networks. 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 with clear inter-component interfaces and
   interactions. It should be possible to build and implement
   individual components in isolation and have them interoperate with
   other independently implemented 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 document is not an Internet Standards Track specification; it is
   published for informational purposes.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Not all documents
   approved by the IESG are a candidate for any level of Internet
   Standard; see Section 2 of RFC 7841.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   http://www.rfc-editor.org/info/rfc8014.

Copyright Notice

Table of Contents

1. Introduction

   This document presents a high-level architecture for building data-
   center Network Virtualization over Layer 3 (NVO3) networks. The
   architecture is given at a high level, which shows the major
   components of an overall system. An important goal is to divide the
   space into smaller individual components that can be implemented
   independently with clear inter-component interfaces and interactions.
   It should be possible to build and implement individual components in
   isolation and have them interoperate with other independently
   implemented 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.

   The motivation for overlay networks is given in "Problem Statement:
   Overlays for Network Virtualization" [RFC7364]. "Framework for Data
   Center (DC) Network Virtualization" [RFC7365] 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 that the NVO3 WG has
   focused on.

2. Terminology

   This document uses the same terminology as [RFC7365]. 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 NV Domains. NVAs can share
      information about an entire NV Domain, or just individual virtual
      networks.

   Tenant System Interface (TSI): The interface to a Virtual Network
      (VN) as presented to a Tenant System (TS, see [RFC7365]). The TSI
      logically connects to the NVE via a Virtual Access Point (VAP).
      To the Tenant System, the TSI is like a Network Interface Card
      (NIC); the TSI presents itself to a Tenant System as a normal
      network interface.

   VLAN: Unless stated otherwise, the terms "VLAN" and "VLAN Tag" are
      used in this document to denote a Customer VLAN (C-VLAN)
      [IEEE.802.1Q]; the terms are used interchangeably to improve
      readability.

3. Background

   Overlay networks are an approach for providing network virtualization
   services to a set of Tenant Systems (TSs) [RFC7365]. 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.
   Additional discussion of some NVO3 use cases can be found in
   [USECASES].

   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. The NVO3 architecture does not impose any restrictions to
   the application of policy controls even within a VN.

   A Network Virtualization Edge (NVE) [RFC7365] 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 VN for which it
   is providing service on behalf of a Tenant System.

```
   +--------+                                        +--------+
   | Tenant +--+                            +----| Tenant |
   | System |  |                            (')  | System |
   +--------+  |        ................    (   ) +--------+
              |   +-+--+  .                . +--+-+  (_)
              |   | NVE|--.                .--| NVE|   |
             +--|    |  .                . |    |---+
               +-+--+  .                . +--+-+
              /         .                .
             /          . L3 Overlay .   +--+-++--------+
   +--------+  /           .   Network   .   | NVE|| Tenant |
   | Tenant +--+        .                .- -|    || System |
   | System |           .                . +--+-++--------+
   +--------+           ................
                             |
                          +----+
                          | NVE|
                          |    |
                          +----+
                             |
                             |
                    ====================
                       |           |
                   +--------+   +--------+
                   | Tenant |   | Tenant |
                   | System |   | System |
                   +--------+   +--------+
```
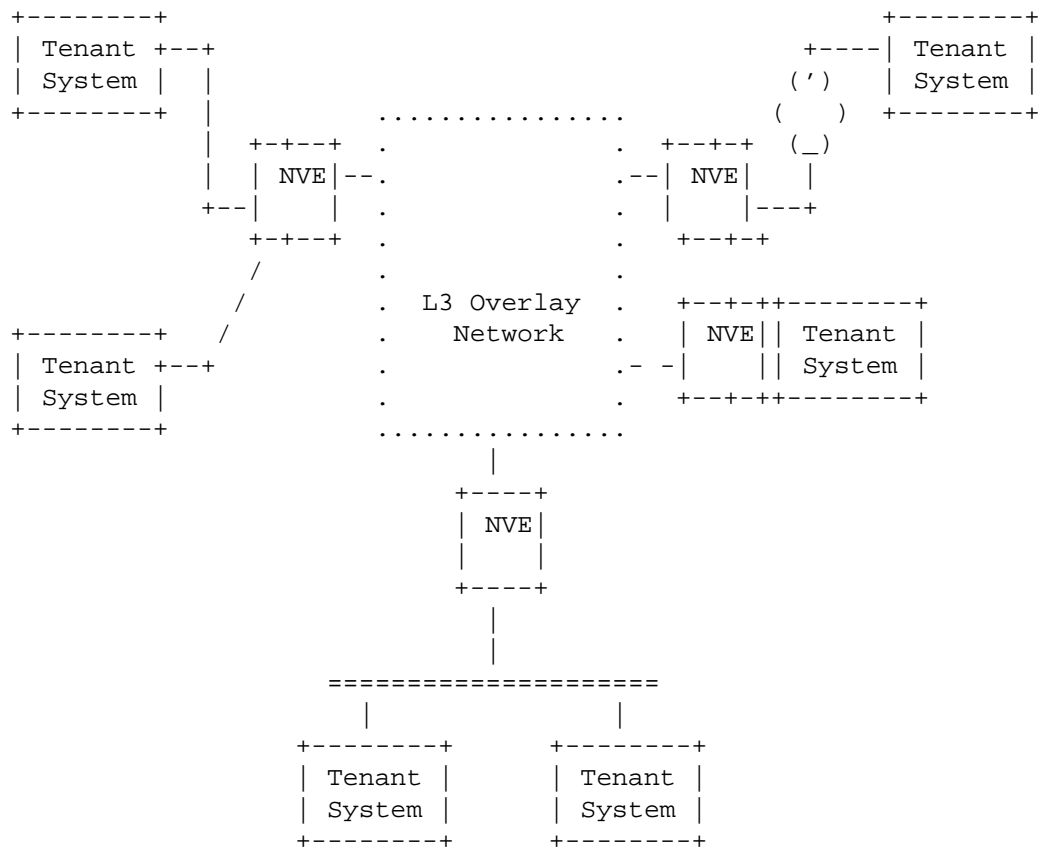
Figure 1: NVO3 Generic Reference Model

   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 Virtual Machine (VM) orchestration
   systems and their use of virtual networks.

3.1. VN Service (L2 and L3)

   A VN provides either Layer 2 (L2) or Layer 3 (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
   generally 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 the NVO3 architecture, tenant frames are tunneled to remote
   NVEs based on the Media Access Control (MAC) addresses of the frame
   headers as originated by the Tenant System. On the underlay, NVO3
   packets are forwarded between NVEs based on the outer addresses of
   tunneled packets.

   For L3 service, VNs are routed networks that transport IP datagrams,
   and a Tenant System is provided with a service that supports only IP
   traffic. Within the NVO3 architecture, 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 by the NVEs and overlay network. For
   L3 service, the Tenant System will be configured with an IP subnet
   that is effectively a point-to-point link, i.e., having only the
   Tenant System and a next-hop router address on it.

   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 or not an NVO3 network 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.

   NVO3 can also provide a combined L2 and L3 service to tenants. A
   combined service provides L2 service for intra-VN communication but
   also provides L3 service for L3 traffic entering or leaving the VN.
   Architecturally, the handling of a combined L2/L3 service within the
   NVO3 architecture is intended to match what is commonly done today in
   non-overlay environments by devices providing a combined bridge/
   router service. With combined service, the virtual network itself
   retains the semantics of L2 service, and all traffic is processed
   according to its L2 semantics. In addition, however, traffic
   requiring IP processing is also processed at the IP level.

The IP processing for a combined service can be implemented on a
standalone device attached to the virtual network (e.g., an IP
router) or implemented locally on the NVE (see Section 5.4 on
Distributed Inter-VN Gateways). For unicast traffic, NVE
implementation of a combined service may result in a packet being
delivered to another Tenant System attached to the same NVE (on
either the same or a different VN), tunneled to a remote NVE, or even
forwarded outside the NV Domain. For multicast or broadcast packets,
the combination of NVE L2 and L3 processing may result in copies of
the packet receiving both L2 and L3 treatments to realize delivery to
all of the destinations involved. This distributed NVE
implementation of IP routing results in the same network delivery
behavior as if the L2 processing of the packet included delivery of
the packet to an IP router attached to the L2 VN as a Tenant System,
with the router having additional network attachments to other
networks, either virtual or not.

3.1.1. VLAN Tags in L2 Service

An NVO3 L2 virtual network service may include encapsulated L2 VLAN
tags provided by a Tenant System but does not use encapsulated tags
in deciding where and how to forward traffic. Such VLAN tags can be
passed through so that Tenant Systems that send or expect to receive
them can be supported as appropriate.

The processing of VLAN tags that an NVE receives from a TS is
controlled by settings associated with the VAP. Just as in the case
with ports on Ethernet switches, a number of settings are possible.
For example, Customer VLAN Tags (C-TAGs) can be passed through
transparently, could always be stripped upon receipt from a Tenant
System, could be compared against a list of explicitly configured
tags, etc.

Note that there are additional considerations when VLAN tags are used
to identify both the VN and a Tenant System VLAN within that VN, as
described in Section 4.2.1.

3.1.2. Packet Lifetime Considerations

For L3 service, Tenant Systems should expect the IPv4 Time to Live
(TTL) or IPv6 Hop Limit in the packets they send to be decremented by
at least 1. For L2 service, neither the TTL nor the Hop Limit (when
the packet is IP) is modified. The underlay network manages TTLs and
Hop Limits in the outer IP encapsulation -- the values in these
fields could be independent from or related to the values in the same
fields of tenant IP packets.

3.2. Network Virtualization Edge (NVE) Background

   Tenant Systems connect to NVEs via a Tenant System Interface (TSI).
   The TSI logically connects to the NVE via a Virtual Access Point
   (VAP), and each VAP is associated with one VN 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 VN.

```
                      |        Data-Center Network (IP)        |
                      |                                        |
                 +----------------------------------------+
                      |                             |
                      |        Tunnel Overlay       |
        +-----------+--------+        +---------+-----------+
        | +---------+------+ |        | +-------+---------+ |
        | | Overlay Module | |        | | Overlay Module  | |
        | +--------+-------+ |        | +--------+-------+ |
        |          |         |        |          |         |
  NVE1  |          |         |        |          |         | NVE2
        |          |         |        |          |         |
        |  +-------+------+   |        |  +-------+------+   |
        |  | VNI1    VNI2 |   |        |  | VNI1    VNI2 |   |
        |  +-+---------+---+   |        |  +-+---------+--+   |
        |    | VAP1    | VAP2 |        |    | VAP1    | VAP2|
        +----+---------+------+        +----+---------+------+
             |         |                    |         |
             |\        |                    |         |
             | \       |                    |        /|
      -------+--\------+------------------+--------/-+--------
             |   \     |      Tenant      |      /   |
         TSI1 |TSI2\    | TSI3          TSI1  TSI2/   TSI3
             +---+ +---+ +---+          +---+ +---+   +---+
             |TS1| |TS2| |TS3|          |TS4| |TS5|   |TS6|
             +---+ +---+ +---+          +---+ +---+   +---+
```
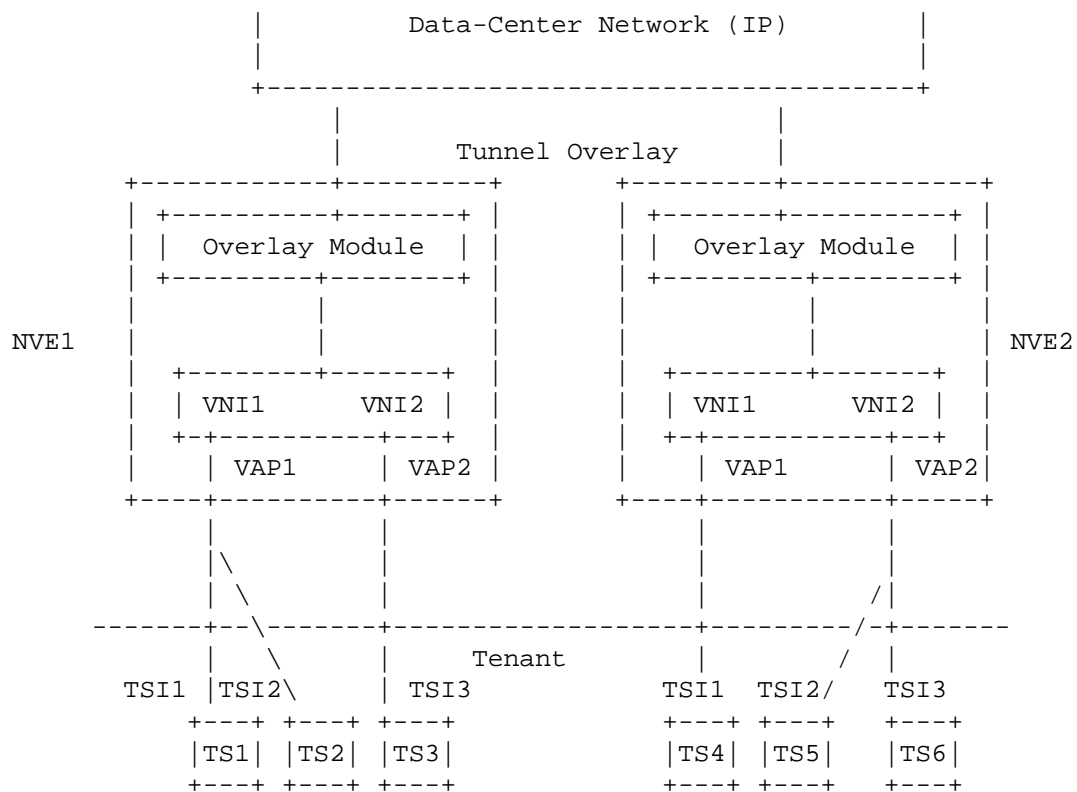
                       Figure 2: NVE Reference Model

   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 information such as the destination address of the NVE to
   tunnel a packet to and the Context ID that should be placed in the
   encapsulation header to identify the virtual network that a tunneled
   packet belongs to.

   On the side facing the data-center network, 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. When the source
   and destination Tenant System are on the same NVE, no encapsulation
   is needed and the NVE forwards traffic directly.

   Conceptually, the NVE is a single entity implementing the NVO3
   functionality. In practice, there are a number of different
   implementation scenarios, as described in detail in Section 4.

3.3. Network Virtualization Authority (NVA) Background

   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.
   While flooding works, it can lead to traffic hot spots and to
   problems in larger networks (e.g., excessive amounts of flooded
   traffic).

   Alternatively, to reduce the scope of where flooding must take place,
   or to eliminate it all together, 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 regard to its scope or
   how it is implemented. NVAs provide a service, and NVEs access that
   service via an NVE-NVA protocol as discussed in Section 8.

   Even when an NVA is present, Ethernet bridge MAC address 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.

   For the rest of this document, it is assumed that an NVA exists and
   will be used. NVAs are discussed in more detail in Section 7.

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 movement 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.

   Note also that although this section uses the terms "VM" and
   "hypervisor" throughout, the same issues apply to other
   virtualization approaches, including Linux Containers (LXC), BSD
   Jails, Network Service Appliances as discussed in Section 5.1, etc.
   From an NVO3 perspective, it should be assumed that where the
   document uses the term "VM" and "hypervisor", the intention is that
   the discussion also applies to other systems, where, e.g., the host
   operating system plays the role of the hypervisor in supporting
   virtualization, and a container plays the equivalent role as a VM.

   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 VM, and
   controls when a VM should be shut down 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 metadata associated with VM images can
   be useful from a network virtualization perspective. For example,
   the metadata 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. Examples of deployed VM
   orchestration systems include VMware's vCenter Server, Microsoft's
   System Center Virtual Machine Manager, and systems based on OpenStack
   and its associated plugins (e.g., Nova and Neutron). Each 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:

Facing the Tenant System: On the side facing the Tenant System, 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.

Facing the Data-Center Network: On the side facing the data-center
   network, an NVE interfaces with the data-center underlay network,
   sending and receiving tunneled 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
APIs are not in scope for the NVO3 WG at the time of publication of
this memo.

Implementing NVE functionality entirely on a server has the
disadvantage that server CPU resources must be spent implementing the
NVO3 functionality. Experimentation with overlay approaches and
previous experience with TCP and checksum adapter offloads suggest

that offloading certain NVE operations (e.g., encapsulation and
decapsulation operations) onto the physical network adapter can
produce performance advantages. As has been done with checksum and/
or TCP server offload and other optimization approaches, there may be
benefits to offloading common operations onto adapters where
possible. Just as important, the addition of an overlay header can
disable existing adapter offload capabilities that are generally not
prepared to handle the addition of a new header or other operations
associated with an NVE.

While the exact details of how to split the implementation of
specific NVE functionality between a server and its network adapters
are an implementation matter and outside the scope of IETF
standardization, the NVO3 architecture should be cognizant of and
support such separation. Ideally, it may even be possible to bypass
the hypervisor completely on critical data-path operations so that
packets between a Tenant System 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. An NVE running on a server (e.g., within a
hypervisor) could support NVO3 service towards the tenant but not
perform all NVE functions (e.g., encapsulation) directly on the
server; some of the actual NVO3 functionality could be implemented on
(i.e., offloaded to) an adjacent switch to which the server is
attached. While one could imagine a number of link types between a
server and the NVE, one simple deployment scenario would involve a
server and NVE separated by a simple L2 Ethernet link. A more
complicated scenario would have the server and NVE separated by a
bridged access network, such as when the NVE resides on a Top of Rack
(ToR) switch, with an embedded switch residing between servers and
the ToR switch.

For the split-NVE case, protocols will be needed that allow the
hypervisor and NVE to negotiate and set up 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 connects
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 [NVE-NVA].

4.2.1. Tenant VLAN Handling in Split-NVE Case

   Preserving tenant VLAN tags across an NVO3 VN, as described in
   Section 3.1.1, poses additional complications in the split-NVE case.
   The portion of the NVE that performs the encapsulation function needs
   access to the specific VLAN tags that the Tenant System is using in
   order to include them in the encapsulated packet. When an NVE is
   implemented entirely within the hypervisor, the NVE has access to the
   complete original packet (including any VLAN tags) sent by the
   tenant. In the split-NVE case, however, the VLAN tag used between
   the hypervisor and offloaded portions of the NVE normally only
   identifies the specific VN that traffic belongs to. In order to
   allow a tenant to preserve VLAN information from end to end between
   Tenant Systems in the split-NVE case, additional mechanisms would be
   needed (e.g., carry an additional VLAN tag by carrying both a C-TAG
   and a Service VLAN Tag (S-TAG) as specified in [IEEE.802.1Q] where
   the C-TAG identifies the tenant VLAN end to end and the S-TAG
   identifies the VN locally between each Tenant System and the
   corresponding NVE).

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 the 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
       tenant 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, a source NVE could use unicast replication to deliver
       traffic. In such a case, an NVE would need to know which remote
       NVEs are participating in the VN. An NVE could use both
       approaches, switching from one mode to the other depending on

factors such as bandwidth efficiency and group membership
sparseness. [FRAMEWORK-MCAST] discusses the subject of multicast
handling in NVO3 in further detail.

4. An NVE maintains necessary information to encapsulate outgoing
    traffic, including what type of encapsulation and what value to
    use for a Context ID to identify the VN 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 (i.e., TSI).

6. An NVE may find it convenient to maintain additional per-VN
    information such as QoS settings, Path MTU information, Access
    Control Lists (ACLs), etc.

4.4. Multihoming of NVEs

   NVEs may be multihomed. That is, an NVE may have more than one IP
   address associated with it on the underlay network. Multihoming
   happens in two different scenarios. First, an NVE may have multiple
   interfaces connecting it to the underlay. Each of those interfaces
   will typically have a different IP address, resulting in a specific
   Tenant Address (on a specific VN) being reachable through the same
   NVE but through more than one underlay IP address. Second, a
   specific Tenant System may be reachable through more than one NVE,
   each having one or more underlay addresses. In both cases, NVE
   address-mapping functionality needs to support one-to-many mappings
   and enable a sending NVE to (at a minimum) be able to fail over from
   one IP address to another, e.g., should a specific NVE underlay
   address become unreachable.

   Finally, multihomed NVEs introduce complexities when source unicast
   replication is used to implement tenant multicast as described in
   Section 4.3. Specifically, an NVE should only receive one copy of a
   replicated packet.

   Multihoming is needed to support important use cases. First, a bare
   metal server may have multiple uplink connections to either the same
   or different NVEs. Having only a single physical path to an upstream
   NVE, or indeed, having all traffic flow through a single NVE would be
   considered unacceptable in highly resilient deployment scenarios that
   seek to avoid single points of failure. Moreover, in today's
   networks, the availability of multiple paths would require that they
   be usable in an active-active fashion (e.g., for load balancing).

4.5. Virtual Access Point (VAP)

   The VAP is the NVE side of the interface between the NVE and the TS.
   Traffic to and from the tenant flows through the VAP. If an NVE runs
   into difficulties sending traffic received on the VAP, it may need to
   signal such errors back to the VAP. Because the VAP is an emulation
   of a physical port, its ability to signal NVE errors is limited and
   lacks sufficient granularity to reflect all possible errors an NVE
   may encounter (e.g., inability to reach a particular destination).
   Some errors, such as an NVE losing all of its connections to the
   underlay, could be reflected back to the VAP by effectively disabling
   it. This state change would reflect itself on the TS as an interface
   going down, allowing the TS to implement interface error handling
   (e.g., failover) in the same manner as when a physical interface
   becomes disabled.

5. Tenant System Types

   This section describes a number of special Tenant System types and
   how they fit into an NVO3 system.

5.1. Overlay-Aware Network Service Appliances

   Some Network Service Appliances [NVE-NVA] (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 multitenancy 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 multitenancy, a Network Service Appliance
   may explicitly support multitenancy. 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 may not be running a 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 NV 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 NVO3 (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 NVO3
   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., 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 may not support a hypervisor and will need an
   appropriate control-plane protocol to obtain the information needed
   to provide NVO3 service.

   Gateways handle several different use cases. For example, one use
   case consists of systems supporting overlays together with systems
   that do not (e.g., bare metal servers). 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.

5.3.1. Gateway Taxonomy

   As can be seen from the discussion above, there are several types of
   gateways that can exist in an NVO3 environment. This section breaks
   them down into the various types that could be supported. Note that
   each of the types below could be either implemented in a centralized
   manner or distributed to coexist with the NVEs.

5.3.1.1. L2 Gateways (Bridging)

   L2 Gateways act as Layer 2 bridges to forward Ethernet frames based
   on the MAC addresses present in them.

   L2 VN to Legacy L2: This type of gateway bridges traffic between L2
      VNs and other legacy L2 networks such as VLANs or L2 VPNs.

   L2 VN to L2 VN: The main motivation for this type of gateway is to
      create separate groups of Tenant Systems using L2 VNs such that
      the gateway can enforce network policies between each L2 VN.

5.3.1.2. L3 Gateways (Only IP Packets)

   L3 Gateways forward IP packets based on the IP addresses present in
   the packets.

   L3 VN to Legacy L2: This type of gateway forwards packets between L3
      VNs and legacy L2 networks such as VLANs or L2 VPNs. The original
      sender's destination MAC address in any frames that the gateway
      forwards from a legacy L2 network would be the MAC address of the
      gateway.

   L3 VN to Legacy L3: This type of gateway forwards packets between L3
      VNs and legacy L3 networks. These legacy L3 networks could be
      local to the data center, be in the WAN, or be an L3 VPN.

   L3 VN to L2 VN: This type of gateway forwards packets between L3 VNs
      and L2 VNs. The original sender's destination MAC address in any
      frames that the gateway forwards from a L2 VN would be the MAC
      address of the gateway.

   L2 VN to L2 VN: This type of gateway acts similar to a traditional
      router that forwards between L2 interfaces. The original sender's
      destination MAC address in any frames that the gateway forwards
      from any of the L2 VNs would be the MAC address of the gateway.

   L3 VN to L3 VN: The main motivation for this type of gateway is to
      create separate groups of Tenant Systems using L3 VNs such that
      the gateway can enforce network policies between each L3 VN.

5.4. Distributed Inter-VN Gateways

   The relaying of traffic from one VN to another deserves special
   consideration. Whether traffic is permitted to flow from one VN to
   another is a matter of policy and would not (by default) be allowed
   unless explicitly enabled. In addition, NVAs are the logical place
   to maintain policy information about allowed inter-VN communication.
   Policy enforcement for inter-VN communication can be handled in (at
   least) two different ways. Explicit gateways could be the central
   point for such enforcement, with all inter-VN traffic forwarded to
   such gateways for processing. Alternatively, the NVA can provide
   such information directly to NVEs by either providing a mapping for a
   target Tenant System (TS) on another VN or indicating that such
   communication is disallowed by policy.

   When inter-VN gateways are centralized, traffic between TSs on
   different VNs can take suboptimal paths, i.e., triangular routing
   results in paths that always traverse the gateway. In the worst
   case, traffic between two TSs connected to the same NVE can be hair-
   pinned through an external 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 supports distributed gateways for the case of
   inter-VN communication. 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.

   Distributed gateways could also be used to distribute other
   traditional router services to individual NVEs. The NVO3
   architecture does not preclude such implementations but does not
   define or require them as they are outside the scope of the NVO3
   architecture.

5.5. ARP and Neighbor Discovery

   Strictly speaking, for an L2 service, special processing of the
   Address Resolution Protocol (ARP) [RFC826] and IPv6 Neighbor
   Discovery (ND) [RFC4861] is not required. ARP requests are
   broadcast, and an NVO3 can deliver ARP requests to all members of a
   given L2 virtual network just as it does for any packet sent to an L2
   broadcast address. Similarly, ND requests are sent via IP multicast,
   which NVO3 can support by delivering via L2 multicast. However, as a
   performance optimization, an NVE can intercept ARP (or ND) requests
   from its attached TSs and respond to them directly using information
   in its mapping tables. Since an NVE will have mechanisms for
   determining the NVE address associated with a given TS, the NVE can
   leverage the same mechanisms to suppress sending ARP and ND requests
   for a given TS to other members of the VN. The NVO3 architecture
   supports such a capability.

6. NVE-NVE Interaction

   Individual NVEs will interact with each other for the purposes of
   tunneling and delivering traffic to remote TSs. At a minimum, a
   control protocol may be needed for tunnel setup and maintenance. For
   example, tunneled traffic may need to be encrypted or integrity
   protected, in which case it will be necessary to set up appropriate
   security associations between NVE peers. It may also be desirable to
   perform tunnel maintenance (e.g., continuity checks) on a tunnel in
   order to detect when a remote NVE becomes unreachable. Such generic
   tunnel setup and maintenance functions are not generally
   NVO3-specific. Hence, the NVO3 architecture expects to leverage
   existing tunnel maintenance protocols rather than defining new ones.

   Some NVE-NVE interactions may be specific to NVO3 (in particular, be
   related to information kept in mapping tables) and agnostic to the
   specific tunnel type being used. For example, when tunneling traffic
   for TS-X to a remote NVE, it is possible that TS-X is not presently
   associated with the remote NVE. Normally, this should not happen,
   but there could be race conditions where the information an NVE has
   learned from the NVA is out of date relative to actual conditions.
   In such cases, the remote NVE could return an error or warning
   indication, allowing the sending NVE to attempt a recovery or
   otherwise attempt to mitigate the situation.

   The NVE-NVE interaction could signal a range of indications, for
   example:

   o "No such TS here", upon a receipt of a tunneled packet for an
      unknown TS

   o   "TS-X not here, try the following NVE instead" (i.e., a redirect)

   o   "Delivered to correct NVE but could not deliver packet to TS-X"

   When an NVE receives information from a remote NVE that conflicts
   with the information it has in its own mapping tables, it should
   consult with the NVA to resolve those conflicts. In particular, it
   should confirm that the information it has is up to date, and it
   might indicate the error to the NVA so as to nudge the NVA into
   following up (as appropriate). While it might make sense for an NVE
   to update its mapping table temporarily in response to an error from
   a remote NVE, any changes must be handled carefully as doing so can
   raise security considerations if the received information cannot be
   authenticated. That said, a sending NVE might still take steps to
   mitigate a problem, such as applying rate limiting to data traffic
   towards a particular NVE or TS.

7. Network Virtualization Authority (NVA)

   Before sending traffic 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 can obtain such information from a Network Virtualization
   Authority (NVA).

   The NVA is the entity that is expected to provide address mapping and
   other information to NVEs. NVEs can 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 regard to its scope or how it is implemented.

7.1. How an NVA Obtains Information

   There are two primary ways in which an NVA can obtain the address
   dissemination information it manages: from the VM orchestration
   system and/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 from 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 TSs connected to that NVE and can distribute
information to the NVE about the VNs those TSs 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.

7.2. 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 the Border
Gateway Protocol (BGP) [RFC4364] (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.

7.3. NVA External Interface

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 to ensure 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.

In the NVO3 architecture, an NVA is accessed through one or more IP
addresses (or an 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 respond
more quickly than another, e.g., due to network conditions, load on
the server, etc.

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 could use some randomization input to select among
equal-priority NVAs. Such a priority scheme facilitates failover and
load balancing, for example, by allowing a network operator to
specify a set of primary and backup NVAs.

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. Key questions with
such an approach are 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.

8. NVE-NVA Protocol

   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-NVA protocol is needed. The NVE-NVA protocol provides two
   functions. First, it allows an NVE to obtain information about the
   location and status of other TSs with which it needs to communicate.
   Second, the NVE-NVA protocol provides a way for NVEs to provide
   updates to the NVA about the TSs 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-NVA protocol might seem straightforward,
   the existence of existing VM orchestration systems complicates the
   choices an NVE has for interacting with the NVA.

8.1. NVE-NVA Interaction Models

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

   o NVEs embedded within the same server as the hypervisor 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 implementations can 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.

   The NVO3 architecture will focus on support for the second model
   above. Existing virtualization environments are already using the
   first model, but they are not sufficient to cover the case of

   standalone gateways -- such gateways may not support virtualization
   and do not interface with existing VM orchestration systems.

8.2. Direct NVE-NVA Protocol

   An NVE can interact directly with an NVA via an NVE-NVA protocol.
   Such a protocol can be either independent of the NVA internal
   protocol or an extension of it. Using a purpose-specific protocol
   would provide 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 would be prudent to assume that once an NVE has been
   implemented and deployed, it may be challenging to get subsequent NVE
   extensions and changes implemented and deployed, whereas an NVA (and
   its associated internal protocols) is more likely to evolve over time
   as experience is gained from usage and upgrades will involve fewer
   nodes.

   Requirements for a direct NVE-NVA protocol can be found in [NVE-NVA].

8.3. Propagating Information Between NVEs and NVAs

   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 information from the NVA
   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 relevant to 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 (or TSs for broadcast
   or multicast traffic), an NVE must know where the remote TS(s)
   currently reside. 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.

There are scenarios where an NVE may wish to query the NVA about
individual mappings within a 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 shut down, 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 to 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 TSs 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.
However, if it needs to communicate with a destination for which it
does not have mapping information, it will need to be able to query
the NVA on demand for the missing information on a per-destination
basis.

The NVO3 architecture will need to support a range of operations
between the NVE and NVA. Requirements for those operations can be
found in [NVE-NVA].

9. 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, an
administrator managing multiple data centers 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 that is part of a different NV Domain, it
still interacts only with its NVA, even when obtaining mappings for
NVEs associated with a different NV Domain.

Figure 3 shows a scenario where two separate NV Domains (A and B)
share information about a VN. VM1 and VM2 both connect to the same
VN, even though the two VMs are in separate NV Domains. There are
two cases to consider. In the first case, NV Domain B 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 A and B may not share a
common address space (i.e., traversal through a NAT device is
required), 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 the VN is available and direct that traffic between
the two nodes be forwarded via IP-G (an IP Gateway). 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.

```
                     xxxxxx                        xxxx        +-----+
    +-----+    xxxxxx     xxxxx            xxxxxx    xxxxx   | VM2 |
    | VM1 |     xx            xx            xxx          xx  |-----|
    |-----|   xx               x            xx            x |NVE-B|
    |NVE-A|   x                x  +----+  x                x +-----+
    +--+--+   x   NV Domain A  x  |IP-G|--x                x    |
      +-------x               xx--+  | x                  xx    |
            x                 x    +----+ x     NV Domain B  x    |
        +---x             xx         xx                    x---+
        |     xxxx      xx        +->xx                     xx
        |       xxxxxxxxx         |  xx                    xx
     +---+-+                      |    xx                 xx
     |NVA-1|                   +--+--+   xx             xxx
     +-----+                   |NVA-2|    xxxx     xxxx
                               +-----+        xxxxx
```
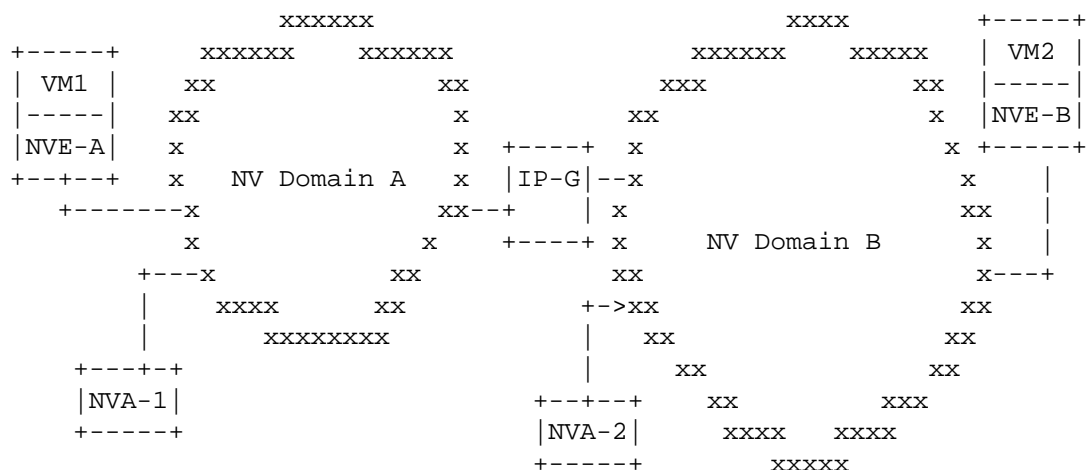
                Figure 3: VM1 and VM2 in Different NV Domains

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 NV 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 VN is
intended to span multiple NVAs.

Reasons for using a federated model include:

o Provide isolation among 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 for intra-
   NVA 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 trade-offs 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.

9.1. 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.

10. Control Protocol Work Areas

   The NVO3 architecture consists of two major distinct entities: NVEs
   and NVAs. In order to provide isolation and independence between
   these two entities, the NVO3 architecture calls for well-defined
   protocols for interfacing between them. For an individual NVA, the
   architecture calls for a logically centralized 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 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; thus, there is little
   benefit in standardizing the internal structure of an NVA.

   Individual NVAs peer with each other in a federated manner. The NVO3
   architecture calls for a well-defined interface between NVAs.

   Finally, a hypervisor-NVE protocol is needed to cover the split-NVE
   scenario described in Section 4.2.

11. NVO3 Data-Plane Encapsulation

   When tunneling tenant traffic, NVEs add an encapsulation header to
   the original tenant packet. The exact encapsulation to use for NVO3
   does not seem to be critical. The main requirement is that the
   encapsulation support a Context ID of sufficient size. A number of
   encapsulations already exist that provide a VN Context of sufficient
   size for NVO3. For example, Virtual eXtensible Local Area Network
   (VXLAN) [RFC7348] has a 24-bit VXLAN Network Identifier (VNI).
   Network Virtualization using Generic Routing Encapsulation (NVGRE)
   [RFC7637] 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.

12. Operations, Administration, and Maintenance (OAM)

   The simplicity of operating and debugging overlay networks will be
   critical for successful deployment.

   Overlay networks are based on tunnels between NVEs, so the
   Operations, Administration, and Maintenance (OAM) [RFC6291] framework
   for overlay networks can draw from prior IETF OAM work for tunnel-
   based networks, specifically L2VPN OAM [RFC6136]. RFC 6136 focuses
   on Fault Management and Performance Management as fundamental to
   L2VPN service delivery, leaving the Configuration Management,
   Accounting Management, and Security Management components of the Open
   Systems Interconnection (OSI) Fault, Configuration, Accounting,
   Performance, and Security (FCAPS) taxonomy [M.3400] for further
   study. This section does likewise for NVO3 OAM, but those three
   areas continue to be important parts of complete OAM functionality
   for NVO3.

   The relationship between the overlay and underlay networks is a
   consideration for fault and performance management -- a fault in the
   underlay may manifest as fault and/or performance issues in the
   overlay. Diagnosing and fixing such issues are complicated by NVO3
   abstracting the underlay network away from the overlay network (e.g.,
   intermediate nodes on the underlay network path between NVEs are
   hidden from overlay VNs).

   NVO3-specific OAM techniques, protocol constructs, and tools are
   needed to provide visibility beyond this abstraction to diagnose and
   correct problems that appear in the overlay. Two examples are
   underlay-aware traceroute [TRACEROUTE-VXLAN] and ping protocol
   constructs for overlay networks [VXLAN-FAILURE] [NVO3-OVERLAY].

   NVO3-specific tools and techniques are best viewed as complements to
   (i.e., not as replacements for) single-network tools that apply to
   the overlay and/or underlay networks. Coordination among the
   individual network tools (for the overlay and underlay networks) and
   NVO3-aware, dual-network tools is required to achieve effective
   monitoring and fault diagnosis. For example, the defect detection
   intervals and performance measurement intervals ought to be
   coordinated among all tools involved in order to provide consistency
   and comparability of results.

   For further discussion of NVO3 OAM requirements, see [NVO3-OAM].

13. Summary

   This document presents the overall architecture for NVO3. The
   architecture calls for three main areas of protocol work:

   1. A hypervisor-NVE protocol to support split-NVEs as discussed in
       Section 4.2

   2. An NVE-NVA protocol for disseminating VN information (e.g., inner
       to outer address mappings)

   3. An NVA-NVA protocol for exchange of information about specific
       virtual networks between federated NVAs

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

14. Security Considerations

   The data plane and control plane described in this architecture will
   need to address potential security threats.

   For the data plane, tunneled application traffic may need protection
   against being misdelivered, being modified, or having its content
   exposed to an inappropriate third party. In all cases, encryption
   between authenticated tunnel endpoints (e.g., via use of IPsec
   [RFC4301]) and enforcing policies that control which endpoints and
   VNs are permitted to exchange traffic can be used to mitigate risks.

   For the control plane, a combination of authentication and encryption
   can be used between NVAs, between the NVA and NVE, as well as between
   different components of the split-NVE approach. All entities will
   need to properly authenticate with each other and enable encryption
   for their interactions as appropriate to protect sensitive
   information.

   Leakage of sensitive information about users or other entities
   associated with VMs whose traffic is virtualized can also be covered
   by using encryption for the control-plane protocols and enforcing
   policies that control which NVO3 components are permitted to exchange
   control-plane traffic.

   Control-plane elements such as NVEs and NVAs need to collect
   performance and other data in order to carry out their functions.
   This data can sometimes be unexpectedly sensitive, for example,
   allowing non-obvious inferences of activity within a VM. This
   provides a reason to minimize the data collected in some environments
   in order to limit potential exposure of sensitive information. As

   noted briefly in RFC 6973 [RFC6973] and RFC 7258 [RFC7258], there is
   an inevitable tension between being privacy sensitive and taking into
   account network operations in NVO3 protocol development.

   See the NVO3 framework security considerations in RFC 7365 [RFC7365]
   for further discussion.

15. Informative References

   [FRAMEWORK-MCAST]
              Ghanwani, A., Dunbar, L., McBride, M., Bannai, V., and R.
              Krishnan, "A Framework for Multicast in Network
              Virtualization Overlays", Work in Progress,
              draft-ietf-nvo3-mcast-framework-05, May 2016.

   [IEEE.802.1Q]
              IEEE, "IEEE Standard for Local and metropolitan area
              networks--Bridges and Bridged Networks", IEEE 802.1Q-2014,
              DOI 10.1109/ieeestd.2014.6991462,
              <http://ieeexplore.ieee.org/servlet/
              opac?punumber=6991460>.

   [M.3400]   ITU-T, "TMN management functions", ITU-T
              Recommendation M.3400, February 2000,
              <https://www.itu.int/rec/T-REC-M.3400-200002-I/>.

   [NVE-NVA] Kreeger, L., Dutt, D., Narten, T., and D. Black, "Network
              Virtualization NVE to NVA Control Protocol Requirements",
              Work in Progress, draft-ietf-nvo3-nve-nva-cp-req-05, March
              2016.

   [NVO3-OAM] Chen, H., Ed., Ashwood-Smith, P., Xia, L., Iyengar, R.,
              Tsou, T., Sajassi, A., Boucadair, M., Jacquenet, C.,
              Daikoku, M., Ghanwani, A., and R. Krishnan, "NVO3
              Operations, Administration, and Maintenance Requirements",
              Work in Progress, draft-ashwood-nvo3-oam-requirements-04,
              October 2015.

   [NVO3-OVERLAY]
              Kumar, N., Pignataro, C., Rao, D., and S. Aldrin,
              "Detecting NVO3 Overlay Data Plane failures", Work in
              Progress, draft-kumar-nvo3-overlay-ping-01, January 2014.

   [RFC826]   Plummer, D., "Ethernet Address Resolution Protocol: Or
              Converting Network Protocol Addresses to 48.bit Ethernet
              Address for Transmission on Ethernet Hardware", STD 37,
              RFC 826, DOI 10.17487/RFC0826, November 1982,
              <http://www.rfc-editor.org/info/rfc826>.

   [RFC4301] Kent, S. and K. Seo, "Security Architecture for the
              Internet Protocol", RFC 4301, DOI 10.17487/RFC4301,
              December 2005, <http://www.rfc-editor.org/info/rfc4301>.

   [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private
              Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February
              2006, <http://www.rfc-editor.org/info/rfc4364>.

   [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman,
              "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861,
              DOI 10.17487/RFC4861, September 2007,
              <http://www.rfc-editor.org/info/rfc4861>.

   [RFC6136] Sajassi, A., Ed. and D. Mohan, Ed., "Layer 2 Virtual
              Private Network (L2VPN) Operations, Administration, and
              Maintenance (OAM) Requirements and Framework", RFC 6136,
              DOI 10.17487/RFC6136, March 2011,
              <http://www.rfc-editor.org/info/rfc6136>.

   [RFC6291] Andersson, L., van Helvoort, H., Bonica, R., Romascanu,
              D., and S. Mansfield, "Guidelines for the Use of the "OAM"
              Acronym in the IETF", BCP 161, RFC 6291,
              DOI 10.17487/RFC6291, June 2011,
              <http://www.rfc-editor.org/info/rfc6291>.

   [RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J.,
              Morris, J., Hansen, M., and R. Smith, "Privacy
              Considerations for Internet Protocols", RFC 6973,
              DOI 10.17487/RFC6973, July 2013,
              <http://www.rfc-editor.org/info/rfc6973>.

   [RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an
              Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May
              2014, <http://www.rfc-editor.org/info/rfc7258>.

   [RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger,
              L., Sridhar, T., Bursell, M., and C. Wright, "Virtual
              eXtensible Local Area Network (VXLAN): A Framework for
              Overlaying Virtualized Layer 2 Networks over Layer 3
              Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014,
              <http://www.rfc-editor.org/info/rfc7348>.

   [RFC7364] Narten, T., Ed., Gray, E., Ed., Black, D., Fang, L.,
              Kreeger, L., and M. Napierala, "Problem Statement:
              Overlays for Network Virtualization", RFC 7364,
              DOI 10.17487/RFC7364, October 2014,
              <http://www.rfc-editor.org/info/rfc7364>.

   [RFC7365] Lasserre, M., Balus, F., Morin, T., Bitar, N., and Y.
              Rekhter, "Framework for Data Center (DC) Network
              Virtualization", RFC 7365, DOI 10.17487/RFC7365, October
              2014, <http://www.rfc-editor.org/info/rfc7365>.

   [RFC7637] Garg, P., Ed. and Y. Wang, Ed., "NVGRE: Network
              Virtualization Using Generic Routing Encapsulation",
              RFC 7637, DOI 10.17487/RFC7637, September 2015,
              <http://www.rfc-editor.org/info/rfc7637>.

   [TRACEROUTE-VXLAN]
              Nordmark, E., Appanna, C., Lo, A., Boutros, S., and A.
              Dubey, "Layer-Transcending Traceroute for Overlay Networks
              like VXLAN", Work in Progress, draft-nordmark-nvo3-
              transcending-traceroute-03, July 2016.

   [USECASES]
              Yong, L., Dunbar, L., Toy, M., Isaac, A., and V. Manral,
              "Use Cases for Data Center Network Virtualization Overlay
              Networks", Work in Progress, draft-ietf-nvo3-use-case-15,
              December 2016.

   [VXLAN-FAILURE]
              Jain, P., Singh, K., Balus, F., Henderickx, W., and V.
              Bannai, "Detecting VXLAN Segment Failure", Work in
              Progress, draft-jain-nvo3-vxlan-ping-00, June 2013.

Authors' Addresses

   David Black
   Dell EMC

   Email: david.black@dell.com

   Jon Hudson
   Independent

   Email: jon.hudson@gmail.com

   Lawrence Kreeger
   Independent

   Email: lkreeger@gmail.com

   Marc Lasserre
   Independent

   Email: mmlasserre@gmail.com

   Thomas Narten
   IBM

   Email: narten@us.ibm.com