Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: October 26, 2014

L. Kreeger
Cisco
D. Dutt
Cumulus Networks
T. Narten
IBM
D. Black
EMC

April 24, 2014

Network Virtualization NVE to NVA Control Protocol Requirements
draft-ietf-nvo3-nve-nva-cp-req-02

Abstract

The document "Problem Statement: Overlays for Network Virtualization" discusses the needs for network virtualization using overlay networks in highly virtualized data centers. The problem statement outlines a need for control protocols to facilitate running these overlay networks. This document outlines the high level requirements to be fulfilled by the control protocols related to building and managing the mapping tables and other state information used by the Network Virtualization Edge to transmit encapsulated packets across the underlying network.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

"Problem Statement: Overlays for Network Virtualization" [I-D.ietf-nvo3-overlay-problem-statement] discusses the needs for network virtualization using overlay networks in highly virtualized data centers and provides a general motivation for building such networks. "Framework for DC Network Virtualization" [I-D.ietf-nvo3-framework] provides a framework for discussing overlay networks generally and the various components that must work together in building such systems. "An Architecture for Overlay Networks (NVO3)" [I-D.ietf-nvo3-arch] presents a high-level architecture for building NVO3 Overlay networks. The reader is assumed to be familiar with these documents.

Section 4.5 of [I-D.ietf-nvo3-overlay-problem-statement] describes three separate work areas that fall under the general category of a control protocol for NVO3. This document focuses entirely on those aspects of the control protocol related to the building and distributing the mapping tables an NVE uses to tunnel traffic from one VM to another. Specifically, this document focuses on work areas 1 and 2 given in Section 4.5 of

[I-D.ietf-nvo3-overlay-problem-statement], and discussed in section 8 of [I-D.ietf-nvo3-arch]. Work areas 1 and 2 cover the interaction between an NVE and the Network Virtualization Authority (NVA) (work area 2) or operation of the NVA itself (work area 1). Requirements related to interaction between a hypervisor and NVE when the two entities reside on separate physical devices (work area 3) are covered in [I-D.kreeger-nvo3-hypervisor-nve-cp-req].

2. Terminology

This document uses the same terminology as found in [I-D.ietf-nvo3-framework] and [I-D.ietf-nvo3-arch]. This section defines additional terminology used by this document.

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

VN Alias: A string name for a VN as used by administrators and customers to name a specific VN. A VN Alias is a human-usable string that can be listed in contracts, customer forms, email, configuration files, etc. and that can be communicated easily vocally (e.g., over the phone). A VN Alias is independent of the underlying technology used to implement a VN and will generally not be carried in protocol fields of control protocols used in virtual networks. Rather, a VN Alias will be mapped into a VN Name where precision is required.

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

VN ID: A unique and compact identifier for a VN within the scope of a specific NVO3 administrative domain. It will generally be more efficient to carry VN IDs as fields in control protocols than VN Names or VN Aliases. There is a one-to-one mapping between a VN Name and a VN ID within an NVO3 Administrative Domain. Depending on the technology used to implement an overlay network, the VN ID could be used as the VN Context in the data plane, or would need to be mapped to a locally-significant context ID.

3. Control Plane Protocol Functionality

The NVO3 problem statement [I-D.ietf-nvo3-overlay-problem-statement], discusses the needs for a control plane protocol (or protocols) to populate each NVE with the state needed to perform its functions.

In one common scenario, an NVE provides overlay encapsulation/ decapsulation packet forwarding services to Tenant Systems that are co-resident with the NVE on the same End Device. For example, when the NVE is embedded within a hypervisor or a Network Service Appliance, as depicted in Figure 1 and Figure 2 below. Alternatively, a Tenant System may use an externally connected NVE. For example, an NVE residing on a physical Network Switch connected to the End Device, as depicted in Figure 3 and Figure 4 below.

There are two control plane aspects for an NVE. One is the protocol between the NVE and its NVA used to populate the NVE's mapping tables for tunneling traffic across the underlying network. Another is the protocol between an End Device (e.g. Hypervisor) and an external NVE used to promptly update the NVE of Tenant System Interface (TSI) status. This latter control plane aspect is not discussed in this document, but is covered in [I-D.kreeger-nvo3-hypervisor-nve-cp-req]. The functional requirements for the NVE to NVA control plane are the same regardless of whether the NVE is embedded within and End Device or in an external device as depicted in Figure 1 through Figure 4 below.

```
       Hypervisor
+-----------------------+
| +--+   +-------+---+  |
| |VM|---|       |   |  |
| +--+   |Virtual|NVE|----- Underlying
| +--+   |Switch |   |  |    Network
| |VM|---|       |   |  |
| +--+   +-------+---+  |
+-----------------------+
```

Hypervisor with an Embedded NVE.

Figure 1

```
    Network Service Appliance
   +---------------------------+
   | +------------+   +-----+  |
   | |Net Service |---|     |  |
   | |Instance    |   |     |  |
   | +------------+   | NVE |------ Underlying
   | +------------+   |     |  |    Network
   | |Net Service |---|     |  |
   | |Instance    |   |     |  |
   | +------------+   +-----+  |
   +---------------------------+
```

Network Service Appliance (physical or virtual) with an Embedded NVE.

Figure 2

```
        Hypervisor              Access Switch
   +------------------+       +-----+-------+
   | +--+   +-------+ |       |     |       |
   | |VM|---|       | | VLAN  |     |       |
   | +--+   |Virtual|---------+ NVE |       +--- Underlying
   | +--+   |Switch | | Trunk |     |       |    Network
   | |VM|---|       | |       |     |       |
   | +--+   +-------+ |       |     |       |
   +------------------+       +-----+-------+
```

Hypervisor with an External NVE.

Figure 3

```
    Network Service Appliance         Access Switch
   +--------------------------+      +-----+-------+
   | +------------+    |\     |      |     |       |
   | |Net Service |----| \    |      |     |       |
   | |Instance    |    |  \   | VLAN |     |       |
   | +------------+    |   |---------+ NVE |       +--- Underlying
   | +------------+    |   |  | Trunk|     |       |    Network
   | |Net Service |----|  /   |      |     |       |
   | |Instance    |    | /    |      |     |       |
   | +------------+    |/     |      |     |       |
   +--------------------------+      +-----+-------+
```

Physical Network Service Appliance with an External NVE.

Figure 4

To support an NVE, a control plane protocol is necessary to provide an NVE with the information it needs to maintain its own internal state necessary to carry out its forwarding functions as explained in detail below.

1. An NVE maintains a per-VN table of mappings from TSI (inner) addresses to Underlying Network (outer) addresses of remote NVEs.

2. An NVE maintains per-VN state for delivering tenant multicast and broadcast packets to other Tenant Systems. Such state could include a list of multicast addresses and/or unicast addresses on the Underlying Network for the NVEs associated with a particular VN.

3. End Devices (such as a Hypervisor or Network Service Appliance) utilizing an external NVE need to "attach to" and "detach from" an NVE. Specifically, a mechanism is needed to notify an NVE when a TSI attaches to or detaches from a specific VN. Such a mechanism would provide the necessary information to the NVE that it needs to provide service to a particular TSI. The details of such a mechanism are out-of-scope for this document and are covered in [I-D.kreeger-nvo3-hypervisor-nve-cp-req].

4. An NVE needs a mapping from each unique VN name to the VN Context value used within encapsulated data packets within the administrative domain that the VN is instantiated.

The NVE to NVA control protocol operates directly over the underlay network. The NVA is expected to be connected to the same underlay network as the NVEs.

Each NVE communicates with only a single logical NVA; However, the NVA can be centralized or distributed between multiple entities for redundancy purposes. When the NVA is made up of multiple entities, better resiliency may be achieved by physically separating them, which may require each entity to be connected to a different IP subnet of the underlay network. For this reason, each NVE should be allowed to be configured with more than one IP addresses for its logical NVA. NVEs should be able to switch between these IP addresses when it detects that the address it is currently using for the NVA is unreachable. How the NVA represents itself externally is discussed in section 7.3 of [I-D.ietf-nvo3-arch].

Note that a single device could contain both NVE and NVA functionality, but the functional interaction between the NVE and NVA within that device should operate similarly to when the NVE and NVA are implemented in separate devices.

3.1. Inner to Outer Address Mapping

When presented with a data packet to forward to a TSI within a VN, the NVE needs to know the mapping of the TSI destination (inner) address to the (outer) address on the Underlying Network of the remote NVE which can deliver the packet to the destination Tenant System. In addition, the NVE needs to know what VN Context to use when sending to a destination Tenant System.

A protocol is needed to provide this inner to outer mapping and VN Context to each NVE that requires it and keep the mapping updated in a timely manner. Timely updates are important for maintaining connectivity between Tenant Systems when one Tenant System is a VM.

Note that one technique that could be used to create this mapping without the need for a control protocol is via data plane learning; However, the learning approach requires packets to be flooded to all NVEs participating in the VN when no mapping exists. One goal of using a control protocol is to eliminate this flooding.

3.2. Underlying Network Multi-Destination Delivery Address(es)

Each NVE needs a way to deliver multi-destination packets (i.e. tenant broadcast/multicast) within a given VN to each remote NVE which has a destination TSI for these packets. Three possible ways of accomplishing this are:

- o Use the multicast capabilities of the Underlying Network.

- o Have each NVE replicate the packets and send a copy across the Underlying Network to each remote NVE currently participating in the VN.

- o Use one or more distribution servers that replicate the packets on the behalf of the NVEs.

Whichever method is used, a protocol is needed to provide on a per VN basis, one or more multicast addresses (assuming the Underlying Network supports multicast), and/or one or more unicast addresses of either the remote NVEs which are not multicast reachable, or of one or more distribution servers for the VN.

The protocol must also keep the list of addresses up to date in a timely manner as the set of NVEs for a given VN changes over time. For example, the set of NVEs for a VN could change as VMs power on/off or migrate to different hypervisors.

3.3. VN Connect/Disconnect Notification

For the purposes of this document, it is assumed that an NVE receives appropriate notifications when a TSI attaches to or detaches from a specific VN. The details of how that is done are orthogonal to the NVE-to-NVA control plane, so long as such notification provides the necessary information needed by the control plane. As one example, the attach/detach notification would presumably include a VN Name that identifies the specific VN to which the attach/detach operation applies to.

3.4. VN Name to VN ID Mapping

Once an NVE (embedded or external) receives a VN connect indication with a specified VN Name, the NVE must determine what VN Context value and other necessary information to use to forward Tenant System traffic to remote NVEs. In one approach, the NVE-to-NVA protocol uses VN Names directly when interacting, with the NVA providing such information as the VN Context (or VN ID) along with egress NVE's address. Alternatively, it may be desirable for the NVE-to-NVA protocol to use a more compact representation of the VN name, that is, a VN ID. In such a case, a specific NVE-to-NVA operation might be needed to first map the VN Name into a VN ID, with subsequent NVE-to-NVA operations utilizing the VN ID directly. Thus, it may be useful for the NVE-to-NVA protocol to support an operation that maps VN Names into VN IDs.

4. Control Plane Characteristics

NVEs are expected to be implemented within both hypervisors (or Network Service Appliances) and within access switches. Any resources used by these protocols (e.g. processing or memory) takes away resources that could be better used by these devices to perform their intended functions (e.g. providing resources for hosted VMs).

A large scale data center may contain hundreds of thousands of these NVEs (which may be several independent implementations); Therefore, any savings in per-NVE resources can be multiplied hundreds of thousands of times.

Given this, the control plane protocol(s) implemented by NVEs to provide the functionality discussed above should have the below characteristics.

1. Minimize the amount of state needed to be stored on each NVE. The NVE should only be required to cache state that it is actively using, and be able to discard any cached state when it is no longer required. For example, an NVE should only need to

maintain an inner-to-outer address mapping for destinations to which it is actively sending traffic as opposed to maintaining mappings for all possible destinations.

2. Fast acquisition of needed state. For example, when a TSI emits a packet destined to an inner address that the NVE does not have a mapping for, the NVE should be able to acquire the needed mapping quickly.

3. Fast detection/update of stale cached state information. This only applies if the cached state is actually being used. For example, when a VM moves such that it is connected to a different NVE, the inner to outer mapping for this VM's address that is cached on other NVEs must be updated in a timely manner (if they are actively in use). If the update is not timely, the NVEs will forward data to the wrong NVE until it is updated.

4. Minimize processing overhead. This means that an NVE should only be required to perform protocol processing directly related to maintaining state for the TSIs it is actively communicating with. For example, if the NVA provides unsolicited information to the NVEs, then one way to minimize the processing on the NVE is for it to subscribe for getting these mappings on a per VN basis. Consequently an NVE is not required to maintain state for all VNs within a domain. An NVE only needs to maintain state (or participate in protocol exchanges) about the VNs it is currently attached to. If the NVE obtains mappings on demand from the NVA, then it only needs to obtain the information relevant to the traffic flows that are currently active. This requirement is for the NVE functionality only. The network node that contains the NVE may be involved in other functionality for the underlying network that maintains connectivity that the NVE is not actively using (e.g., routing and multicast distribution protocols for the underlying network).

5. Highly scalable. This means scaling to hundreds of thousands of NVEs and several million VNs within a single administrative domain. As the number of NVEs and/or VNs within a data center grows, the protocol overhead at any one NVE should not increase significantly.

6. Minimize the complexity of the implementation. This argues for using the least number of protocols to achieve all the functionality listed above. Ideally a single protocol should be able to be used. The less complex the protocol is on the NVE, the more likely interoperable implementations will be created in a timely manner.

7. Extensible. The protocol should easily accommodate extension to meet related future requirements. For example, access control or QoS policies, or new address families for either inner or outer addresses should be easy to add while maintaining interoperability with NVEs running older versions.

8. Simple protocol configuration. A minimal amount of configuration should be required for a new NVE to be provisioned. Existing NVEs should not require any configuration changes when a new NVE is provisioned. Ideally NVEs should be able to auto configure themselves.

9. Do not rely on IP Multicast in the Underlying Network. Many data centers do not have IP multicast routing enabled. If the Underlying Network is an IP network, the protocol should allow for, but not require the presence of IP multicast services within the data center.

10. Flexible mapping sources. It should be possible for either NVEs themselves, or other third party entities (e.g. data center management or orchestration systems) to create inner to outer address mappings in the NVA. The protocol should allow for mappings created by an NVE to be automatically removed from all other NVEs if it fails or is brought down unexpectedly.

11. Secure. See the Security Considerations section below.

5. Security Considerations

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

The protocol(s) should protect the integrity of the mapping against both off-path and on-path attacks. It should authenticate the systems that are creating mappings, and rely on light weight security mechanisms to minimize the impact on scalability and allow for simple configuration.

Use of an overlay exposes virtual networks to attacks on the underlying network beyond attacks on the control protocol that is the subject of this draft. In addition to the directly applicable security considerations for the networks involved, the use of an overlay enables attacks on encapsulated virtual networks via the underlying network. Examples of such attacks include traffic injection into a virtual network via injection of encapsulated traffic into the underlying network and modifying underlying network traffic to forward traffic among virtual networks that should have no

connectivity. The control protocol should provide functionality to help counter some of these attacks, e.g., distribution of NVE access control lists for each virtual network to enable packets from non-participating NVEs to be discarded, but the primary security measures for the underlying network need to be applied to the underlying network. For example, if the underlying network includes connectivity across the public Internet, use of secure gateways (e.g., based on IPsec [RFC4301]) may be appropriate.

The inner to outer address mappings used for forwarding data towards a remote NVE could also be used to filter incoming traffic to ensure the inner address sourced packet came from the correct NVE source address, allowing access control to discard traffic that does not originate from the correct NVE. This destination filtering functionality should be optional to use.

6. Acknowledgements

Thanks to the following people for reviewing and providing feedback: Fabio Maino, Victor Moreno, Ajit Sanzgiri, Chris Wright.

7. Informative References

[I-D.ietf-nvo3-arch]
Black, D., Hudson, J., Kreeger, L., Lasserre, M., and T. Narten, "An Architecture for Overlay Networks (NVO3)", draft-ietf-nvo3-arch-01 (work in progress), February 2014.

[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-05 (work in progress), January 2014.

[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.

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

Appendix A. Change Log

A.1. Changes from draft-ietf-nvo3-nve-nva-cp-req-01 to -02

1. Added references to the architecture document [I-D.ietf-nvo3-arch].

2. Terminology: Usage of "TSI" in several places.

Authors' Addresses

Lawrence Kreeger
Cisco

Email: kreeger@cisco.com

Dinesh Dutt
Cumulus Networks

Email: ddutt@cumulusnetworks.com

Thomas Narten
IBM

Email: narten@us.ibm.com

David Black
EMC

Email: david.black@emc.com