Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 12, 2015

F. Baker, Ed.
C. Marino
I. Wells
R. Agarwalla
S. Jeuk
G. Salgueiro
Cisco Systems

February 8, 2015

A Model for IPv6 Operation in OpenStack
draft-baker-openstack-ipv6-model-02

Abstract

This is an overview of a network model for OpenStack, designed to dramatically simplify scalable network deployment and operations.

Requirements Language

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

Design Principle

Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away.

Antoine de Saint-Exupery

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 12, 2015.

Table of Contents

1. Introduction

OpenStack, and its issues.

1.1. What is OpenStack?

OpenStack is a cloud computing orchestration solution developed using an open source community process. It consists of a collection of 'projects', each implementing the creation, control, and administration of tenant resources. There are separate OpenStack projects for managing compute, storage and network resources.

Neutron is the project that manages OpenStack networking. It exposes a northbound API to the other OpenStack projects for programmatic control over tenant network connectivity. The southbound interface is implemented as one or more device driver plugins that are built to interact with specific devices in the network. This approach provides the flexibility to deploy OpenStack networking using a range of alternative techniques.

An OpenStack tenant, in the Kilo and earlier releases, is required to create what OpenStack identifies as a 'Neutron Network' connecting their virtual machines. This Network is instantiated via the plugins as either a layer 2 network, a layer 3 network, or as an overlay network. The actual implementation is unknown to the tenant. The technology used to provide these networks is selected by the OpenStack operator based upon the requirements of the cloud deployment.

The tenant also is required, in the Kilo and earlier releases, to specify an 'IP Subnet' for each Network. This specification is made by providing a CIDR prefix for IPv4 address allocation via DHCP or for IPv6 address allocation via DHCP or SLAAC. This address range may be from within the address range of the datacenter (non-overlapping), or overlapping [RFC1918] addresses. Tenants may create multiple Networks, each with its own Subnet.

An OpenStack Subnet is a logical layer 2 network and requires layer 3 routing for packets to exit the Subnet. This is achieved by attaching the Subnet to a Neutron Router. The Neutron router implements Network Address Translation for external traffic from tenant networks as well for providing connectivity to tenant networks from the outside. Using Linux utilities, OpenStack can support overlapping RFC 1918 addresses between tenants.

OpenStack Subnets are typically implemented as VLANs in a datacenter. When tenant scalability requirement grow large, an overlay approach is typically used. Because of the difficulties in scaling and administering large layer 2 and/or overlay networks, some OpenStack integrations chose not to provide isolated Subnets and simply offer tenants a layer 3 based network alternative.

OpenStack uses Layer 3 and Layer 2 Linux utilities on hosts to provide protection against IP/MAC spoofing and ARP poisoning.

1.2. OpenStack Scaling Issues

One of the fundamental requirements of OpenStack Networking (Neutron) is to provide scalable, isolated tenant networks. Today this is achieved via L2 segmentation using either a) standard 802.1Q VLANs or b) an overlay approach based on one of several L2 over L3 encapsulation techniques available today such as 802.1ad, VXLAN, STT or NVGRE.

However, these approaches still struggle to provide scalable, transparent, manageable, high performance, isolated tenant networks. VLAN's don't scale beyond 4096 (2^12) networks and have complex trunking requirements when tenants span host and racks. IEEE 802.1ad

(QinQ) partially solves that, but adds another limit - at most 2^12 tenants, each of which may have 2^12 VLANs. IP Encapsulation introduces additional complexity on host computers running hypervisors as well as impact performance of tenant applications running on virtual machines. Overlay based isolation techniques may also impair traditional network monitoring and performance management tools. Moreover, when these isolated (L2) networks require external access to other networks or the public Internet, they require even more complex solutions to accommodate overlapping IP prefixes and network address translation (NAT).

As more capabilities are built on to these layer 2 based 'virtual' networks, complexity continues to grow.

This draft presents a new Layer 3 based approach to OpenStack networking using IPv6 that can be deployed natively on IPv6 networks. It will be shown that this approach can provide tenant isolation without the limitations of existing alternatives, as well as deliver high performance networks transparently using a simplified tenant network connectivity model, without the overhead of encapsulation or managing overlapping IP addresses and address translations. We note that some large content providers, notably Google and Facebook [FaceBook-IPv6], are going in exactly this direction.

2. Requirements

In this section, we attempt to list critical requirements.

2.1. Design approach

As a design approach, we presume an IPv6-only data center in a world that might have IPv4 or IPv6 clients outside of it. This design explicitly does not depend on VLANs, QinQ, VXLAN, MPLS, Segment Routing, LISP, IP/IP or GRE tunnels, or any other supporting encapsulation. Data center operators remain free to use any of those tools, but they are not required. If we can do everything required for OpenStack networking with IPv6 alone, these other networking technologies may be used as optimizations. If we are unable to satisfy the OpenStack requirements that also is something we wish to know and understand.

OpenStack is designed to be used by many cloud users or 'tenants'. Scalable, secure and isolated tenant networks are a requirement for building a multi-tenant cloud datacenter. The OpenStack administrator/operator can design and configure a cloud environment to provide network isolation using the approach described in this document, alone, or in combination with any of the above network technologies . However, all the details of the underlying technology

and implementation details are completely transparent to the tenant itself.

2.2. Multiple Data Centers

A common requirement in network and data center operations is reliability, serviceability, and maintainability of their operations in the presence of an outage. At minimum, this implies multihoming in the sense of having multiple upstream ISPs; in many cases, it also implies multiple and at times duplicate data centers, and tenants stretched or able to be readily moved or recreated across multiple data centers.

2.3. Large Data Centers

Microsoft Azure [Microsoft-Azure] has purchased a 100 acre piece of land for the construction of a single data center. In terms of physical space, that is enough for a data center with about half a million 19' RETMA racks.

With even modest virtual machine density, infrastructure at this scale easily exhausts the 16M available RFC 1918 private addresses (i.e. 10.0.0.0/8) and explains the recent efforts by webscale cloud providers to deploy IPv6 throughout their new datacenters.

2.4. Multi-tenancy

While it is possible that a single tenant would require a 100 acre data center, it would be unusual. In most such data centers, one would expect a large number of tenants.

2.5. Isolation

Isolation is required between tenants, and at times between tenants hierarchically related to larger tenants.

2.5.1. Inter-tenant isolation

A 'tenant' is defined as a set of resources under common administrative control. It may be appropriate for tenants to communicate with each other within the context of an application or relationships among their owners or operators. However, unless specified otherwise, tenants are intended to operate as if they were on their own company's premises and be isolated from one another.

2.5.2. Intra-tenant isolation

There are often security compartments within a corporate network, just as there are security barriers between companies. As a result, there is a recursive isolation requirement: it must be possible to isolate an identified part of a tenant (which we also think of as a tenant) from another part of the same tenant.

2.6. Operational Simplicity

To the extent possible (and, for operators, the concept will bring a smile), operation of a multi-location multi-tenant data center, and the design of an application that runs in one, should be simple and uncoupled.

As discussed in [RFC3439], this requires that the operational model required to support a tenant with only two physical machines, or virtual machines in the same physical chassis, should be the same as that required to support a tenant running a million machines in a federated multiple data center application. Additionally, this same operational model should scale from running a single tenant up to many thousands of tenants.

2.7. Address space

As described in Section 1.1, currently, an OpenStack tenant is required to specify a Subnet's CIDR prefix for IP address allocation. With this proposal, this is no longer required.

2.8. Data Center Federation

It must be possible to extend the architecture across multiple data centers. These data centers may be operated by distinct entities, with security policies that apply to their interconnection.

2.9. Path MTU Issues

An issue in virtualized data center architectures is Path MTU Discovery [RFC1981] implementation. Implementing Path MTU requires the ICMPv6 [RFC4443] Packet Too Big message to get from the originating router or middleware to the indicated host, which is in this case virtual and potentially hidden within a tunnel. This is a special case of the issues raised in [RFC2923].

3. Models

3.1. Configuration Model

In the OpenStack model, the cloud computing user, or tenant, is building something Edward Yourdon might call a 'structured design' for the application they are building. In the 1960's, when Yourdon started specifying process and data flow diagrams, these were job steps in a deck of Job Control Language cards; in OpenStack, they are multiple, individual machines, virtual or physical, running parts of a structured application.

In these, one might find a load balancer that receives and distributes requests to request processors, a set of stored data processing applications, and the storage they depend on. What is important to the OpenStack tenant is that 'this' and 'that' communicate, potentially using or not using multicast communications, and don't communicate with 'the other'. Typically unnecessary is any and all information regarding how this communication actually needs to occur (i.e. placement of routers, switches, and IP subnets, prefixes, etc.).

An IPv6 based networking model simplifies the configuration of tenant connectivity requirements. Global reachability eliminates the need for network address translation devices as well as tenant-specified Subnet prefixes (Section 2.7), although tenant-specified ULA prefixes or prefixes from the owner of the tenant's address space are usable with it. With the exception of network security functions, no network devices need to be specified or configured to provide connectivity.

3.2. Data Center Model

The premises of the routing and addressing models are that

- The address tells the routing system what topological location to deliver a packet to, and within that, what interface to deliver it to, and

- The routing system should deliver traffic to a resource if and only if the sender is authorized to communicate with that resource.

- Contrary to the OpenStack Neutron Networking Model, tunnels are not necessary to provide tenant network isolation; we include resources in a tenant network by a Role-based Access Control model, but address the tenant resources within the data center in a manner that scales for the data center.

We expect to find the data center to be composed of some minimal unit of connectivity and maintenance, such as a rack or row, and equipped with one or more Top-of-Rack or End-of-Row switch(es); each configured with at least one subnet prefix, perhaps one per such switch. For the purposes of this note, these will be called Racks and Top-of-Rack switches, and when applied to other architectures the appropriate translation needs to be imposed.

Figure 1 describes a relatively typical rack design. It is a simple fat-tree architecture, with every device in a pair, so that any failure has an immediate hot backup. There are other common designs, such as those that consider each rack to be in a 'row' and in a 'column', with one or more distribution switches in each.

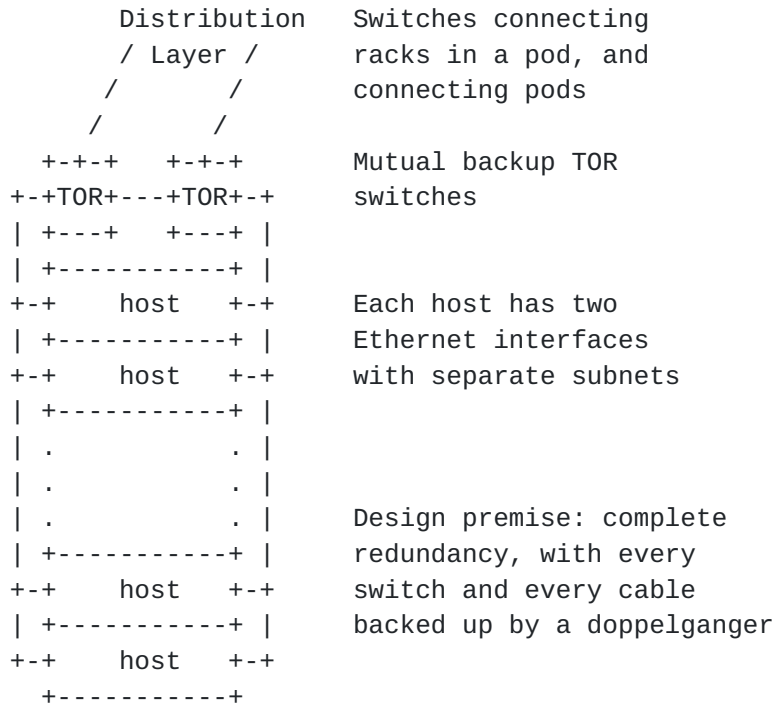

Figure 1: Typical Rack Design

3.2.1. Tenant Address Model

Tenant resources need to be told, by configuration or naming, the addresses of resources they communicate with. This is true regardless of their location or relationship to a given tenant. In environments with well-known addresses, this becomes complex and unscalable. This was learned very early with Internet hostnames; a single 'hostfile' was maintained by a central entity and updated daily, which quickly became unwieldy. The result was the development of the Domain Name System; the level of indirection between names and addresses improved scalability. It also facilitated ongoing

maintenance. If a service needed multiple servers, or a server needed to change its address, that was trivially solved by changing the DNS Resource Record; every resource that needed the new address would obtain it the next time it queried the DNS. It has also facilitated the IPv4/IPv6 transition; a resource that has an IPv6 address is given a AAAA record in addition to, or to replace, its IPv4 A record.

Similarly, today's reliance on NAPT technology frequently limits the capabilities of an application. It works reasonably well for a client accessing a client/server application when the protocol does not carry addressing information. If there is an expectation that one resource's private address will be meaningful to a peer, such as when an SIP client presents its address in SDP or an HTTP server presents an address in a redirection, either the resource needs to understand the difference between an 'inside' and an 'outside' address and know which is which, or it needs a traversal algorithm that changes the addresses. For peer-to-peer applications, this ultimately means providing a network design in which those issues don't apply.

IPv6 provides global addresses, enough of them that there is no real expectation of running out any time soon, making these issues go away. In addition, with the IPv4 address space running out, both globally and within today's large datacenters, there aren't necessarily addresses available for an IPv4 application to use, even as a floating IP address.

Hence, the model we propose is that a resource in a tenant is told the addresses of the other resources with which it communicates. They are IPv6 addresses, and the data center takes care to ensure that inappropriate communications do not take place.

3.2.1.1. Use of Global Unicast Addresses by Tenants

A unicast address in an IP network identifies a topological location, by association with an IP prefix (which might be for a subnet or any aggregate of subnets). It also identifies a single interface located within that subnet, which may or may not be instantiated at the time. We assume that there is a subnet associated with a top-of-rack switch or whatever its counterpart would be in a given network design, and that the physical and virtual machines located in that rack have addresses in that subnet. This is the same prefix that is used by the datacenter administrator.

3.2.1.2. Unique Local Addresses

A common requirement is that tenants have the use of some form of private address space. In an IPv6 network, a Unique Local IPv6 Unicast Address [RFC4193] may be used to accomplish this. In this case, however, the addresses will need to be explicitly assigned to physical or virtual machines used by the tenant, perhaps using DHCP or YANG, where a standard IPv6 address could be allocated using SLAAC, DHCPv6, or other technologies.

The value of this is that the distinction between a Global Address and a Unique Local Address is a corner case in the data center; a ULA will not generally be useful when communicating outside the data center, but within the data center it is rational. Tenants have no routing information or other awareness of the prefix. This is not intended for use behind a NAPT; resources that need accessibility to or from resources outside the tenant, and especially outside the data center, need global addresses.

3.2.1.3. Multicast Domains

Multicast capability is a capability enjoyed by some groups of resources, that enables them to send a single message and have it delivered to multiple destinations roughly simultaneously. At the link layer, this means sending a message once that is received by a specified set of recipient resources using hardware capabilities. IP multicast can be implemented on a LAN as specified in [RFC4291], and can also cross multiple subnets directly, using routing protocols such as Protocol Independent Multicast [RFC4601] [RFC4602] [RFC4604] [RFC4605] [RFC4607]. In IPv6, the model would be that when a group of resources is created with a multicast capability, it is allocated one or more source-specific transient group addresses as defined in section 2.7 of that RFC.

3.2.1.4. IPv4 Interaction Model

OpenStack IPv4 Neutron uses "floating IPv4 addresses" - global or public IPv4 addresses and Network Address Translation - to enable remote resources to connect to tenant private network endpoints. Tenant end points can connect out to remote resources through an "External Default Gateway". Both of these depend on NAPT (DNAT/SNAT) [RFC2391] to ensure that IPv4 end points are able communicate and at the same time ensure tenant isolation.

If IPv6 is deployed in a data center, there are fundamentally two ways a tenant can interact with IPv4 peers:

- it can run existing IPv4 OpenStack technology in parallel with the IPv6 deployment, or

- It can have a translator at the data center edge (such as described in [I-D.ietf-v6ops-siit-dc]) that associates an IPv4 address or address plus port with an IPv6 address or address plus port. The IPv4 address, in this model, becomes a floating IPv4 address attached to an internal IPv6 address. The 'data center edge' is, by definition, a system that has IPv4 reachability to at least the data center's upstream ISP and all IPv4 systems in the data center, IPv6 connectivity to all of the IPv6 systems in the data center, and (if the upstream offers IPv6 service) IPv6 connectivity to the upstream as well.

The first model is complex, if for no other reason than that there are two fundamental models in use, one with various encapsulations hiding overlapping address space and one with non-overlapping address space.

To simplify the network, as noted in Section 2.1, we suggest that the data center be internally IPv6-only, and IPv4 be translated to IPv6 at the data center edge. The advantage is that it enables IPv4 access while that remains in use, and as IPv6 takes over, it reduces the impact of vestigial support for IPv4.

The SIIT Translation model in [I-D.ietf-v6ops-siit-dc] has IPv4 traffic come to an translator [RFC6145][RFC6146] having a pre-configured translation, resulting in an IPv6 packet indistinguishable from the packet the remote resource might have sent had it been IPv6-capable, with one exception. The IPv6 destination address is that of the endpoint (the same address advertised in a AAAA record), but the source address is an IPv4-Embedded IPv6 Address [RFC6052] with the IPv4 address of the sender embedded in a prefix used by the translator.

Access to external IPv4 resources is provided in the same way: an DNS64 [RFC6147] server is implemented that contains AAAA records with an IPv4-Embedded IPv6 Address [RFC6052] with the IPv4 address of the remote resource embedded in a prefix used by the translator.

This follows the Framework for IPv4/IPv6 Translation [RFC6144], making the internal IPv4 address a floating IP address attached to an internal IPv6 address, and the external 'dial-out' address indistinguishable from a native IPv6 address.

3.2.1.5. Legacy IPv4 OpenStack

The other possible model, applicable to IPv4-only devices, is to run a legacy OpenStack environment inside IPv6 tunnels. This preserves the data center IPv6-only, and enables IPv4-only applications, notably those whose licenses tie them to IPv4 addresses, to run. However, it adds significant overhead in terms of encapsulation size and network management complexity.

3.2.2. Use of Global Addresses by the Data Center

Every rack and physical host requires an IP prefix that is reachable by the OpenStack operator. This will normally be a global IPv6 unicast address. For scalability purposes, as isolation is handled separately, this is normally the same prefix as is used by tenants in the rack.

3.3. Inter-tenant security services

In this model, the a label is used to identify a set of virtual or physical systems under common ownership and administration that are authorized to communicate freely among themselves - a tenant. Tenants are not generally authorized to communicate with each other, but interactions between specified tenants may be authorized, and specific systems may be authorized to communicate generally.

The fundamental premise is that the vSwitch can determine whether a VM is authorized to send or receive a given message. It does so by finding the label in a message being sent or received and comparing it to a locally-held authorization policy. This policy would indicate that the VM is permitted to send or receive messages containing one of a small list of labels. In the case of a label contained in the IID of an IPv6 address, it would also need to verify the prefix used in the address, as this type of policy would be specific to an IPv6 prefix.

A set of possible choices that were considered is to be found in Appendix B. The key questions are a list of considerations, presented in no particular order:

- In what way does the approach the IPv6 Path MTU?
- How does the address come into being?
- What security implications apply? For example, how hard would it be for a VM to spoof the source address or attack a random destination?

- What is the service offered? Can, for example, policy be applied at
  - The sender of a datagram?
  - The receiver of a datagram?
  - An arbitrary point between the sender and receiver?
  - At the data center edge, on arriving or departing traffic other data centers?
  - At the data center edge, on arriving or departing traffic random locations?
- In what way does the approach the IPv6 Path MTU?

3.4. IPv6 Tenant Isolation using the Label

Neutron today already implements a form of Network Ingress Filtering [RFC2827]. It prevents the VM from emitting traffic with an unauthorized MAC, IPv4, or IPv6 source address.

In addition to this, in this model Neutron prevents the VM from transmitting a network packet with an unauthorized label value. The VM MAY be configured with and authorized to use one of a short list of authorized label values, as opposed to simply having its choice overridden; in that case, Neutron verifies the value and overwrites one not in the list.

When a hypervisor is about to deliver an IPv6 packet to a VM, it checks the label value against a list of values that the VM is permitted to receive. If it contains an unauthorized value, the hypervisor discards the packet rather than deliver it. If the Flow Label is in use, Neutron zeros the label prior to delivery.

The intention is to hide the label value from malware potentially found in the VM, and enable the label to be used as a form of first and last hop security. This provides basic tenant isolation, if the label is assigned as a tenant identifier, and may be used more creatively such as to identify a network management application as separate from a managed resource.

3.5. Isolation in Routing

This concept has the weakness that if a packet is not dropped at its source, it is dropped at its destination. It would be preferable for

the packet to be dropped in flight, such as at the top-of-rack switch or an aggregation router.

Concepts discussed in IS-IS LSP Extendibility [I-D.baker-ipv6-isis-dst-flowlabel-routing][RFC5120][RFC5308] and OSPFv3 LSA Extendibility [I-D.baker-ipv6-ospf-dst-flowlabel-routing] [I-D.ietf-ospf-ospfv3-lsa-extend][RFC5340] may be used to isolate tenants in the routing of the data center backbone. This is not strictly necessary, if Section 3.4 is uniformly and correctly implemented. It does, however, present a second defense against misconfiguration, as the filter becomes ubiquitous in the data center and as scalable as routing.

4. BCP 38 Ingress Filtering

As noted in Section 3.4, Neutron today implements a form of Network Ingress Filtering [RFC2827]. It prevents the VM from emitting traffic with an unauthorized MAC, IPv4, or IPv6 source address.

In IPv6, this is readily handled when the address or addresses used by a VM are selected by the OpenStack operator. It may then configure a per-VM filter with the addresses it has chosen, following logic similar to the Source Address Validation Solution for DHCP [I-D.ietf-savi-dhcp] or SEND [RFC7219]. This is also true of IPv6 Stateless Address Autoconfiguration (SLAAC) [RFC4862] when the MAC address is known and not shared.

However, when SLAAC is in use and either the MAC address is unknown or SLAAC's Privacy Extensions [RFC4941][RFC7217], are in use, Neutron will need to implement the provisions of FCFS SAVI: First-Come, First-Served Source Address Validation [RFC6620] in order to learn the addresses that a VM is using and include them in the per-VM filter.

5. Moving virtual machines

This design supports these kinds of required layer 2 networks with the additional use of a layer 2 over layer 3 encapsulation and tunneling protocol, such as VXLAN [RFC7348]. The important point here being that these overlays are used to address specific tenant network requirements and NOT deployed to remove the scalability limitations of OpenStack networking.

There are at least three ways VM movement can be accomplished:

- Recreation of the VM
- VLAN Modification

- Live Migration of a Running Virtual Machine

5.1. Recreation of the VM

The simplest and most reliable is to

1. Create a new VM in the new location,
2. Add its address to the DNS Resource Record for the name, allowing new references to the name to send transactions there,
3. Remove the old address from the DNS Resource Record (including the SIIT translation, if one exists), ending the use of the old VM for new transactions,
4. Wait for the period of the DNS Resource Record's lifetime (including the SIIT translation, if one exists), as it will get new requests throughout that interval,
5. Wait for the for the old VM to finish any outstanding transactions, and then
6. Kill the old VM.

This is obviously not movement of an existing VM, but preservation of the same number and function of VMs by creation of a new VM and killing the old.

5.2. Live Migration of a Running Virtual Machine

At http://blogs.vmware.com/vsphere/2011/02/vmotion-whats-going-on-under-the-covers.html, VMWare describes its capability, called vMotion, in the following terms:

1. Shadow VM created on the destination host.
2. Copy each memory page from the source to the destination via the vMotion network. This is known as preCopy.
3. Perform another pass over the VM's memory, copying any pages that changed during the last preCopy iteration.
4. Continue this iterative memory copying until no changed pages (outstanding to be-copied pages) remain or 100 seconds elapse.
5. Stun the VM on the source and resume it on the destination.

In a native-address environment, we add three steps:

1. Shadow VM created on the destination host.

2. Copy each memory page from the source to the destination via the vMotion network. This is known as preCopy.

3. Perform another pass over the VM's memory, copying any pages that changed during the last preCopy iteration.

4. Continue this iterative memory copying until no changed pages (outstanding to be-copied pages) remain or 100 seconds elapse.

5. Stitch routing for the old address.

6. Stun the VM on the source and resume it on the destination.

7. Renumber the VM as instructed in [RFC4192].

8. Unstitch routing for the old address.

If the VM is moved within the same subnet (which usually implies the same rack), there is no stitching or renumbering apart from ensuring that the MAC address moves with the VM. When the VM moves to a different subnet, however, we need to restitch routing, at least temporarily. This obviously calls for some definitions.

Stitching Routing: The VM is potentially in communication with two sets of peers: VMs in the same subnet, and VMs in different subnets.

* The router in the new subnet is instructed to advertise a host route (/128) to the moved VM, and to install a static route to the old address with the VM's address in the new subnet as its next hop address. Traffic from VMs from other subnets will now follow the host route to the VM in its new location.

* The router in the old subnet is instructed to direct LAN traffic to the VM's MAC Address to its IPv6 forwarding logic. Traffic from other VMs in the old subnet will now follow the host route to the moved VM.

Renumbering: This step is optional, but is good hygiene if the VM will be there a while. If the VM will reside in its new location only temporarily, it can be skipped.

Note that every IPv6 address, unlike an IPv4 address, has a lifetime. At least in theory, when the lifetime expires, neighbor relationships with the address must be extended or the address removed from the system. The Neighbor Discovery [RFC4861] process

in the subnet router will periodically emit a Router Advertisement; the VM will gain an IPv6 address in the new subnet at that time if not earlier. As described in [RFC4192], DNS should be changed to report the new address instead of the old. The DNS lifetime and any ambient sessions using the old address are now allowed to expire. That this point, any new sessions will be using the new address, and the old is vestigial.

Waiting for sessions using the address to expire can take an arbitrarily long interval, because the session generally has no knowledge of the lifetime of the IPv6 address.

Unstitching Routing: This is the reverse process of stitching. If the VM is renumbered, when the old address becomes vestigial, the address will be discarded by the VM; if the VM is subsequently taken out of service, it has the same effect. At that point, the host route is withdrawn, and the MAC address in the old subnet router's tables is removed.

6. OpenStack implications

6.1. Configuration implications

1. Neutron MUST be configured with a pre-determined default label value for each tenant virtual network Section 3.4.

2. Neutron MAY be configured with a set of authorized label values for each tenant virtual network Section 3.4.

3. A virtual tenant network MAY be configured with a set of authorized label values Section 3.4.

4. Neutron MUST be configured with one or more label values per virtual tenant network that the network is permitted to receive Section 3.4.

6.2. vSwitch implications

On messages transmitted by a virtual machine

Label Correctness: As described in Section 3.3, ensure that the label in the packet is one that the VM is authorized to use. Exactly what label is in view is a deferred, and potentially configurable, option. Again Depending on configuration, the vSwitch may overwrite whatever value is there, or may ratify that the value there is as specified in a VM-specific list.

Source Address Validation: As described in Section 4, force the source address to be among those the VM is authorized to use. The VM may simultaneously be authorized to use several addresses.

Destination Address Validation: OpenStack for IPv4 permits a NAT translation, called a 'floating IP address', to enable a VM to communicate outside the domain; without that, it cannot. For IPv6, the destination address should be permitted by some access list, which may permit all addresses, or addresses matching one or more CIDR prefixes such as permitted multicast addresses, and the prefix of the data center.

On messages received for delivery to a virtual machine

Label Authorization: As described in Section 3.4, the vSwitch only delivers a packet to a VM if the VM is authorized to receive it. The VM may have been authorized to receive several such labels.

Each approach in the appendix discusses filtering.

7. IANA Considerations

This document does not ask IANA to do anything.

8. Security Considerations

In Section 2.5 and Section 3.3, this specification considers inter-tenant and intra-tenant network isolation. It is intended to contribute to the security of a network, much like encapsulation in a maze of tunnels or VLANs might, but without the complexities and overhead of the management of such resources. This does not replace the use of IPSec, SSH, or TLS encryption or the use authentication technologies; if these would be appropriate in an on-premises corporate data center, they remain appropriate in a multi-tenant data center regardless of the isolation technology. However, one can think of this as a simple inter-tenant firewall based on the concepts of role-based access control; if it can be readily determined that a sender is not authorized to communicate with a receiver, such a transmission is prevented.

9. Privacy Considerations

This specification places no personally identifying information in an unencrypted part of a packet.

10. Acknowledgements

This document grew out of a discussion among the authors and contributors.

11. Contributors

Puneet Konghot
Cisco Systems
San Jose, California 95134
USA
Email: pkonghot@cisco.com

Shannon McFarland
Cisco Systems
Boulder, Colorado 80301
USA
Email: shmcfarl@cisco.com

Figure 2

12. References

12.1. Normative References

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

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

12.2. Informative References

[FaceBook-IPv6]
Pepelnjak, I., "Facebook Is Close to Having an IPv6-only Data Center http://blog.ipspace.net/2014/03/facebook-is-close-to-having-ipv6-only.html", March 2014.

[I-D.baker-ipv6-isis-dst-flowlabel-routing]
Baker, F., "Using IS-IS with Role-Based Access Control", draft-baker-ipv6-isis-dst-flowlabel-routing-00 (work in progress), February 2013.

[I-D.baker-ipv6-ospf-dst-flowlabel-routing]
Baker, F., "Using OSPFv3 with Role-Based Access Control", draft-baker-ipv6-ospf-dst-flowlabel-routing-02 (work in progress), May 2013.

[I-D.ietf-6man-default-iids]
Gont, F., Cooper, A., Thaler, D., and W. Will, "Recommendation on Stable IPv6 Interface Identifiers", draft-ietf-6man-default-iids-01 (work in progress), October 2014.

[I-D.ietf-ospf-ospfv3-lsa-extend]
Lindem, A., Mirtorabi, S., Roy, A., and F. Baker, "OSPFv3 LSA Extendibility", draft-ietf-ospf-ospfv3-lsa-extend-03 (work in progress), May 2014.

[I-D.ietf-savi-dhcp]
Bi, J., Wu, J., Yao, G., and F. Baker, "SAVI Solution for DHCP", draft-ietf-savi-dhcp-26 (work in progress), May 2014.

[I-D.ietf-v6ops-siit-dc]
tore, t., "SIIT-DC: Stateless IP/ICMP Translation for IPv6 Data Centre Environments", draft-ietf-v6ops-siit-dc-00 (work in progress), December 2014.

[I-D.previdi-6man-segment-routing-header]
Previdi, S., Filsfils, C., Field, B., and I. Leung, "IPv6 Segment Routing Header (SRH)", draft-previdi-6man-segment-routing-header-04 (work in progress), November 2014.

[Microsoft-Azure]
Sverdlik, Y., "Report: Microsoft Buys 100 Acres of Iowa land for Data Center http://www.datacenterknowledge.com/archives/2014/08/01/report-microsoft-buys-100-acres-iowa-land-data-center/", August 2014.

[RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, August 1996.

[RFC2205] Braden, B., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

[RFC2391] Srisuresh, P. and D. Gan, "Load Sharing using IP Network Address Translation (LSNAT)", RFC 2391, August 1998.

[RFC2710] Deering, S., Fenner, W., and B. Haberman, "Multicast Listener Discovery (MLD) for IPv6", RFC 2710, October 1999.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, May 2000.

[RFC2923] Lahey, K., "TCP Problems with Path MTU Discovery", RFC 2923, September 2000.

[RFC3439] Bush, R. and D. Meyer, "Some Internet Architectural Guidelines and Philosophy", RFC 3439, December 2002.

[RFC3697] Rajahalme, J., Conta, A., Carpenter, B., and S. Deering, "IPv6 Flow Label Specification", RFC 3697, March 2004.

[RFC4192] Baker, F., Lear, E., and R. Droms, "Procedures for Renumbering an IPv6 Network without a Flag Day", RFC 4192, September 2005.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, October 2005.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

[RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.

[RFC4601] Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", RFC 4601, August 2006.

[RFC4602] Pusateri, T., "Protocol Independent Multicast - Sparse Mode (PIM-SM) IETF Proposed Standard Requirements Analysis", RFC 4602, August 2006.

[RFC4604] Holbrook, H., Cain, B., and B. Haberman, "Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast", RFC 4604, August 2006.

[RFC4605] Fenner, B., He, H., Haberman, B., and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP /MLD Proxying")", RFC 4605, August 2006.

[RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, August 2006.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, September 2007.

[RFC5120] Przygienda, T., Shen, N., and N. Sheth, "M-ISIS: Multi Topology (MT) Routing in Intermediate System to Intermediate Systems (IS-ISs)", RFC 5120, February 2008.

[RFC5308] Hopps, C., "Routing IPv6 with IS-IS", RFC 5308, October 2008.

[RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", RFC 5340, July 2008.

[RFC5548] Dohler, M., Watteyne, T., Winter, T., and D. Barthel, "Routing Requirements for Urban Low-Power and Lossy Networks", RFC 5548, May 2009.

[RFC5673] Pister, K., Thubert, P., Dwars, S., and T. Phinney, "Industrial Routing Requirements in Low-Power and Lossy Networks", RFC 5673, October 2009.

[RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, October 2010.

[RFC6144] Baker, F., Li, X., Bao, C., and K. Yin, "Framework for IPv4/IPv6 Translation", RFC 6144, April 2011.

[RFC6145] Li, X., Bao, C., and F. Baker, "IP/ICMP Translation Algorithm", RFC 6145, April 2011.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011.

[RFC6147] Bagnulo, M., Sullivan, A., Matthews, P., and I. van Beijnum, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", RFC 6147, April 2011.

[RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, November 2011.

[RFC6620] Nordmark, E., Bagnulo, M., and E. Levy-Abegnoli, "FCFS SAVI: First-Come, First-Served Source Address Validation Improvement for Locally Assigned IPv6 Addresses", RFC 6620, May 2012.

[RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, April 2014.

[RFC7219] Bagnulo, M. and A. Garcia-Martinez, "SEcure Neighbor Discovery (SEND) Source Address Validation Improvement (SAVI)", RFC 7219, May 2014.

[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, August 2014.

[UCC] Jeuk, S., Szefer, J., and S. Zhou, "Towards Cloud, Service and Tenant Classification for Cloud Computing", IEEE Xplore Digital Library: Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium, May 2014.

Appendix A. Change Log

Initial Version: October 2014

First update: January 2015

Appendix B. Alternative Labels considered

B.1. IPv6 Flow Label

The IPv6 flow label may be used to identify a tenant or part of a tenant, and to facilitate access control based on the flow label value. The flow label is a flat 20 bits, facilitating the designation of 2^20 (1,048,576) tenants without regard to their

location. 1,048,576 is less than infinity, but compared to current data centers is large, and much simpler to manage.

Note that this usage differs from the current IPv6 Flow Label Specification [RFC6437]. It also differs from the use of a flow label recommended by the IPv6 Specification [RFC2460], and the respective usages of the flow label in the Resource ReSerVation Protocol [RFC2205] and the previous IPv6 Flow Label Specification [RFC3697], and the projected usage in Low-Power and Lossy Networks [RFC5548][RFC5673]. Within a target domain, the usage may be specified by the domain. That is the viewpoint taken in this specification.

B.1.1. Metaconsiderations

B.1.1.1. Service offered

To Be Supplied

B.1.1.2. Pros and Cons

B.1.1.2.1. The case in favor of this approach

To Be Supplied

B.1.1.2.2. The case against

To Be Supplied

B.1.1.3. Filtering considerations

To Be Supplied

B.2. Federated Identity

B.2.1. Introduction

In the course of developing draft-baker-ipv6-openstack-model, it was determined that a way was needed to encode a federated identity for use in Role-Based Access Control. This appendix describes an IPv6 [RFC2460] option that could be carried in the Hop-by-Hop or Destination Options Header. The format of an option is defined in section 4.2 of that document, and the Hop-by-Hop and Destination Options are defined in sections 4.3 and 4.6 of that document respectively.

A 'Federated Identity', in the words of the Wikipedia, 'is the means of linking an electronic identity and attributes, stored across

multiple distinct identity management systems.' In this context, it is a fairly weak form of that; it is intended for quick interpretation in an access list at the Internet layer as opposed to deep analysis for login or other security purposes at the application layer, and rather than identifying an individual or a system, it identifies a set of systems whose members are authorized to communicate freely among themselves and may also be authorized to communicate with other identified sets of systems. Either two systems are authorized to communicate or they are not, and unauthorized traffic can be summarily discarded. The identifier is defined in a hierarchical fashion, for flexibility and scalability.

'Role-Based Access Control', in this context, applies to groups of virtual or physical hosts, not individuals. In the simplest case, the several tenants of a multi-tenant data center might be identified, and authorized to communicate only with other systems within the same 'tenant' or with identified systems in other tenants that manage external access. One could imagine a company purchasing cloud services from multiple data center operators, and as a result wanting to identify the systems in its tenant in one cloud service as being authorized to communicate with the systems its tenant of the other. One could further imagine a given department within that company being authorized to speak only with itself and an identified set of other departments within the same company. To that end, when a datagram is sent, it is tagged with the federated identify of the sender (e.g., {datacenter, client, department}), and the receiving system filters traffic it receives to limit itself to a specific set of authorized communicants.

B.2.2. Federated identity Option

The option is defined as a sequence of numbers that identify relevant parties hierarchically. The specific semantics (as in, what number identifies what party) are beyond the scope of this specification, but they may be interpreted as being successively more specific; as shown in Figure 3, the first might identify a cloud operator, the second, if present, might identify a client of that operator, and the third, if present, might identify a subset of that client's systems. In an application entirely used by Company A, there might be only one number, and it would identify sets of systems important to Company A such as business units. If Company A uses the services of a multi-tenant data center #1, it might require that there be two numbers, identifying Company A and its internal structure. If Company A uses the services of both multi-tenant data centers #1 and #2, and they are federated, the identifier might need to identify the data center, the client, and the structure of the client.

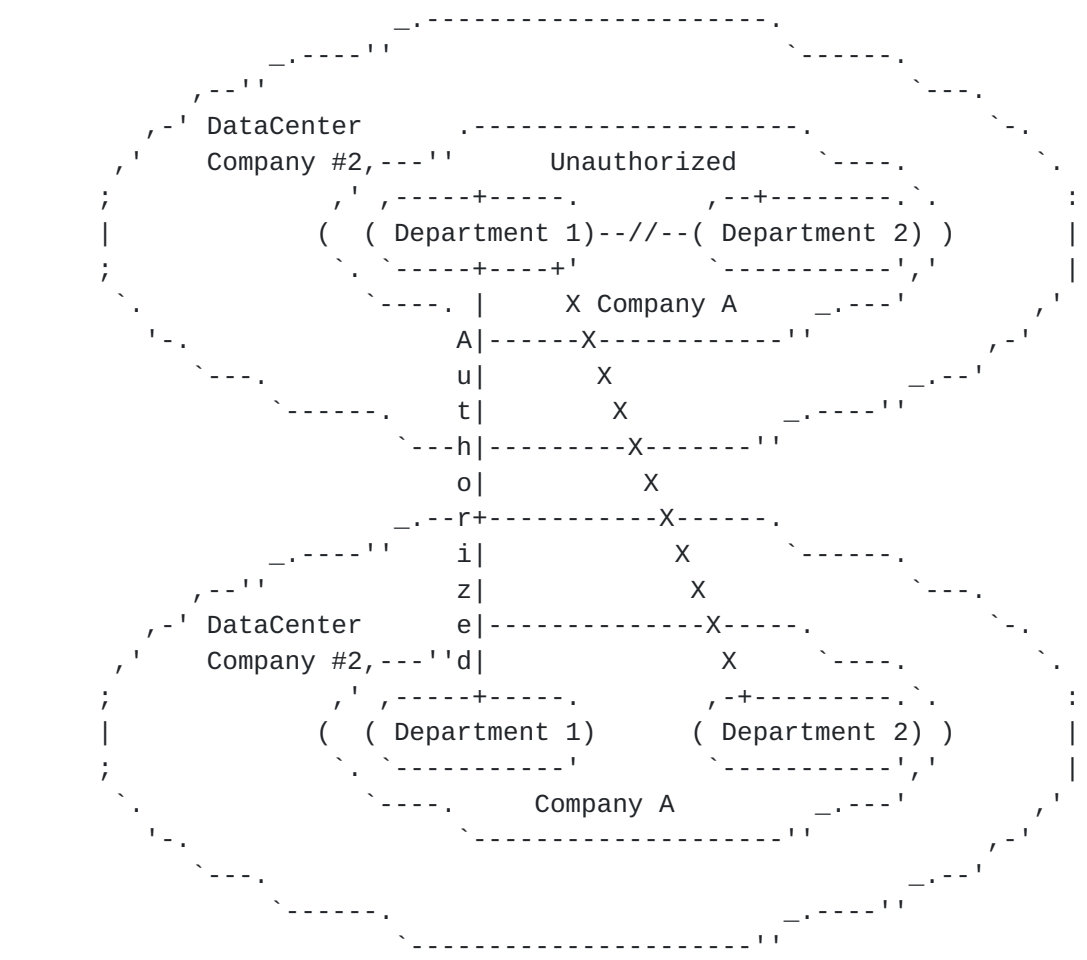

Figure 3: Use case: Identifying authorized communicatants in an RBAC environment

B.2.2.1. Option Format

A number (Figure 4) is represented as a base 128 number whose coefficients are stored in the lower 7 bits of a string of bytes. The upper bit of each byte is zero, except in the final byte, in which case it is 1. The most significant coefficient of a non-zero number is never zero.

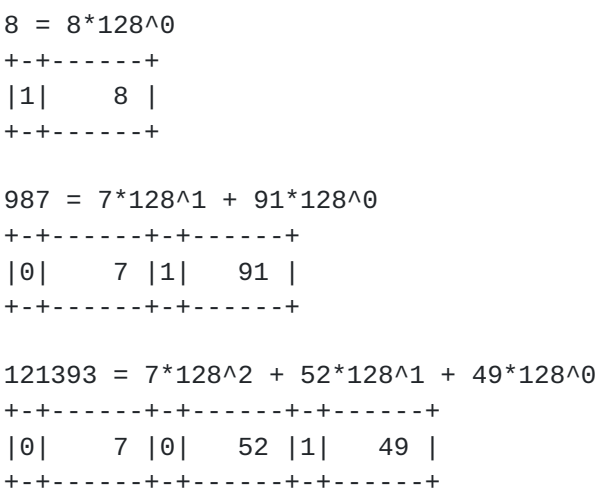

Figure 4: Sample numbers

The identifier {8, 987, 121393} looks like

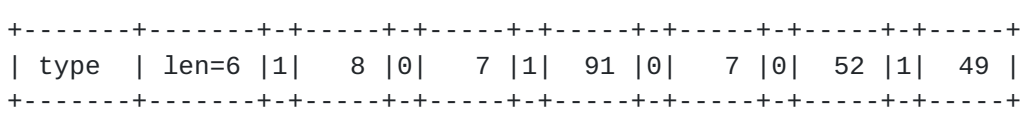

Figure 5

B.2.2.1.1. Use in the Destination Options Header

In an environment in which the validation of the option only occurs in the receiving system or its hypervisor, this option is best placed in the Destination Options Header.

B.2.2.1.2. Use in the Hop-by-Hop Header

In an environment in which the validation of the option occurs in transit, such as in a firewall or other router, this option is best placed in the Hop-by-Hop Header.

B.2.3. Metaconsiderations

B.2.3.1. Service offered

To Be Supplied

B.2.3.2. Pros and Cons

B.2.3.2.1. The case in favor of this approach

To Be Supplied

B.2.3.2.2. The case against

To Be Supplied

B.2.3.3. Filtering considerations

To Be Supplied

B.3. Universal Cloud Classification

B.3.1. Introduction

Cloud environments suffer from ambiguity in identifying their services and tenants. Traffic from different cloud providers cannot be distinguished easily on the Internet. Filters are simply not able to obtain the provider, service and tenant identities from network packets without leveraging other more latency intense inspection methods. This appendix describes the Universal Cloud Classification (UCC) [UCC] approach as a way to identify cloud providers, their services and tenants on the network layer. It introduces a Cloud-ID, Service-ID and Tenant-ID. The IDs are incorporated into an IPv6 extension header and can be used for different use-cases both within and outside a Cloud Environment. The format of the IDs and their characteristics are defined in Appendix B.3.2 of the document and the extension header is defined in Appendix B.3.3.

Applications and users are defined in many different ways in cloud environments, therefore ambiguity is multifold:

1. The first ambiguity is described by how a service is defined in cloud environments. Here, an application within a Cloud Provider is called a service. However, the cloud providers network can not distinguish services from services run on top of other services. Distinguishing sub-services hosted by a service becomes critical when applying network services to specific sub-services.

2. Secondly, a tenant in a cloud provider can have different meanings. Here, tenant is used to define a consumer of a cloud service. At the same time a service run on top of another service can be considered a tenant of that particular service. These ambiguities make it extremely difficult to uniquely identify services and their tenants in cloud environments. This

multi-layered service and tenant relationship is one of the most complex tasks to handle using existing technologies.

A service can be defined as a group of entities offering a specific function to a tenant within a Cloud Provider.

B.3.2. Universal Cloud Classification Options

Three IDs are defined that classify a Tenant specific to the service used within a certain Cloud Provider. The Cloud-ID, Service-ID and Tenant-ID are defined hierarchically and support service-stacking. The IDs are based on the 'Digital Object Identifier' scheme and support incorporating metadata per ID. The ID can be of variable length but Cloud-ID, Service-ID and Tenant-ID are proposed within a 4 byte, 6 byte and 6 byte tuple respectively.

B.3.2.1. Cloud ID

The Cloud ID is a globally unique ID that is managed by a registrar similar to DNS. It is 4 bytes in sizes and defined with a 10 bit location part and a 22 bit provider ID.

B.3.2.2. Service ID

The Service ID is used to identify a service both within and outside a cloud environment. It is a 6 byte long ID that is separated into several sub-IDs defining the data center, service and an option field. The Data Center location is defined by 8 bits, the Service is 32 bits long and the Option field provides another 8 bits. The option bits can be used to incorporate information used for en-route or destination tasks.

B.3.2.3. Tenant ID

The Tenant-ID is classifying consumers (tenants) of Cloud Services. It is a 6 byte long ID that is defined and managed by the Cloud Provider. Similar to the Service-ID the Tenant-ID incorporates metadata specific to that tenant. The MetaData field is of variable length and can be defined by the Cloud Provider as needed.

B.3.3. UCC Extension Header

The UCC proposal [UCC] defines an IPv6 hop-by-hop extension header to incorporate the Cloud-ID, Service-ID and Tenant-ID. Each ID area also includes bits to define enroute behavior for devices understanding/not-understanding the newly defined hop-by-hop extension header. This is useful for legacy devices on the Internet

to avoid drops the packet but forward it without processing the added IPv6 extension header.

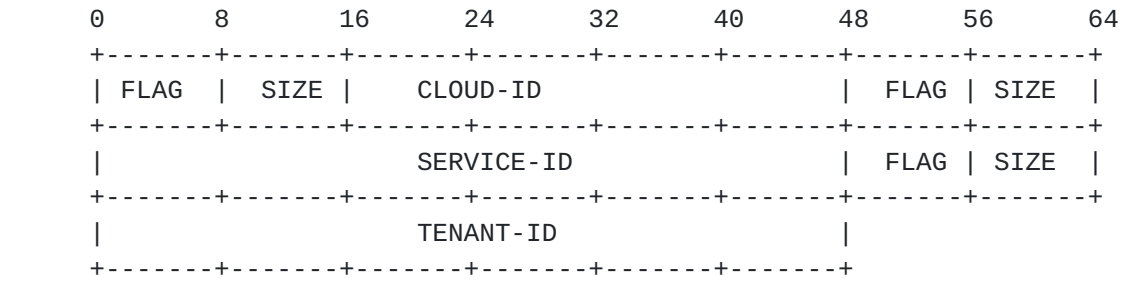

Figure 6: UCC IPv6 hop-by-hop extension header

The overall extension header size is 22 bytes.

B.4. Policy List in Segment Routing Header

The model here suggests using the Policy List described in the IPv6 Segment Routing Header [I-D.previdi-6man-segment-routing-header]. Technically, it would violate that specification, as the Policy List is described as containing a set of optional addresses representing specific nodes in the SR path, where in this case it would be a 128 bit number identifying the tenant or other set of communicating nodes.

B.4.1. Metaconsiderations

B.4.1.1. Service offered

To Be Supplied

B.4.1.2. Pros and Cons

B.4.1.2.1. The case in favor of this approach

To Be Supplied

B.4.1.2.2. The case against

To Be Supplied

B.4.1.3. Filtering considerations

To Be Supplied

B.5. RFC 4291 Interface Identifier (IID)

The approach starts from the observation that Openstack assigns the MAC address used by a VM, and can assign it according to any algorithm it chooses. A desire has been expressed to put the tenant identifier into the IPv6 address. This would put it into the IID in that address without modifying the VM OS or the virtual switch.

B.5.1. Address Format

The proposed address format is identical to the IPv6 EUI-48 based Address [RFC4291], and is derived from the MAC address space specified in SLAAC [RFC4862]. However, the MAC address provided by the OpenStack Controller differs from an IEEE 802.3 MAC Address.

Walking through the details, an IEEE 802.3 MAC Address (Figure 7) consists of two single bit fields, a 22 bit Organizationally Unique Identifier (OUI), and a serial number or other NIC-specific identifier. The intention is to create a globally unique address, so that the NIC may be used on any LAN in the world without colliding with other addresses.

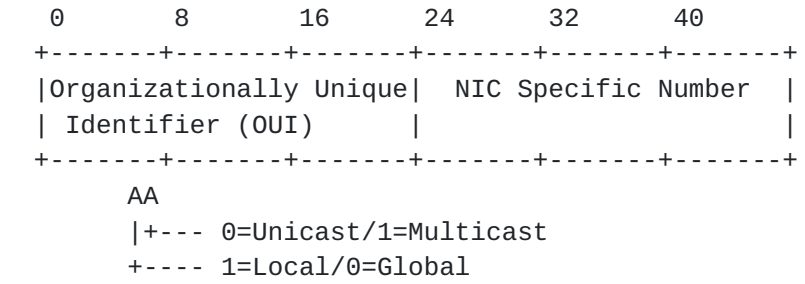

Figure 7: Ethernet MAC Address as specified by IEEE 802.3

[RFC4291] describes a transformation from that address (which it refers to as an EUI-48 address) to the IID of an IPv6 Address (Figure 8).

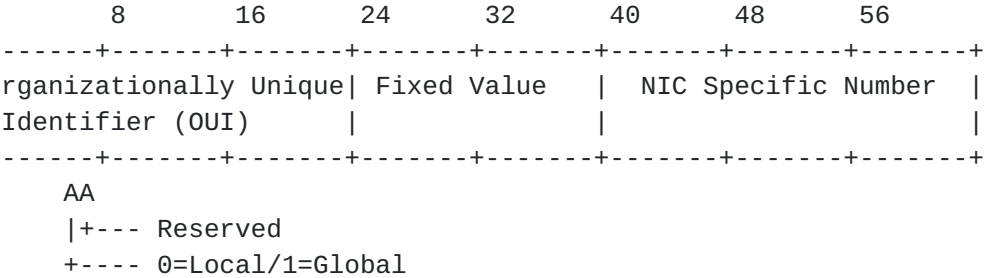

Figure 8: RFC 4291 IPv6 Address

OpenStack today specifies a local IEEE 802.3 address (bit 6 is one). IPv6 addresses are either installed using DHCP or derived from the MAC address via SLAAC.

One could imagine the MAC address used in an OpenStack environment including both a tenant identifier and a system number on a LAN. If the tenant identifier is 24 bits (it could be longer or shorter, but for this document it is treated as 24 bits), as in common in VxLAN and QinQ implementations, that would allow for a 22 bit system number, plus two magic bits specifying a locally defined unicast address, as shown in Figure 9. Alternatively, the first byte could be some specified values such as 0xFA (as is common with current OpenStack implementations), followed by a 16 bit system number within the subnet.

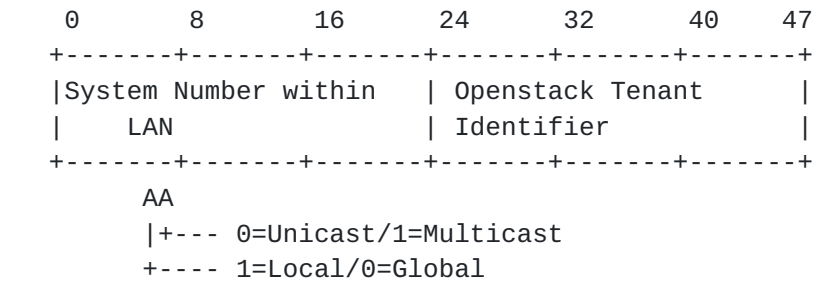

Figure 9: Ethernet MAC Address as installed by OpenStack

After being passed through SLAAC, that results in an IID that contains the Tenant ID in bits 48..63, has bit 6 zero as a locally-specified unicast address, and a 22 bit system number, as in Figure 10.

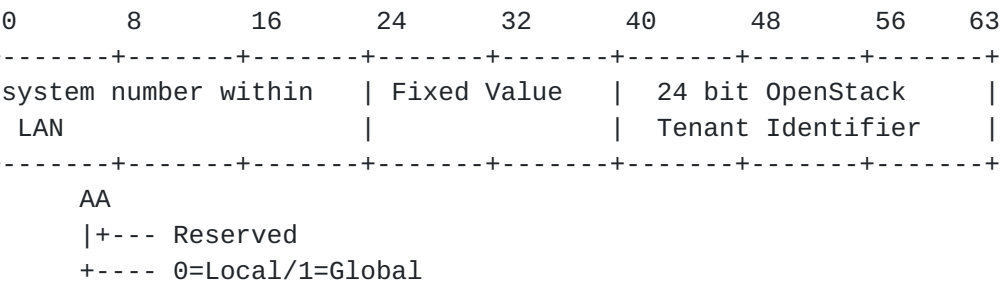

Figure 10: RFC 4291 IPv6 IID Derived from OpenStack MAC Address

More generally, if SLAAC is not in use and addresses are conveyed using DHCPv6 or another technology, the IID would be as described in Figure 11.

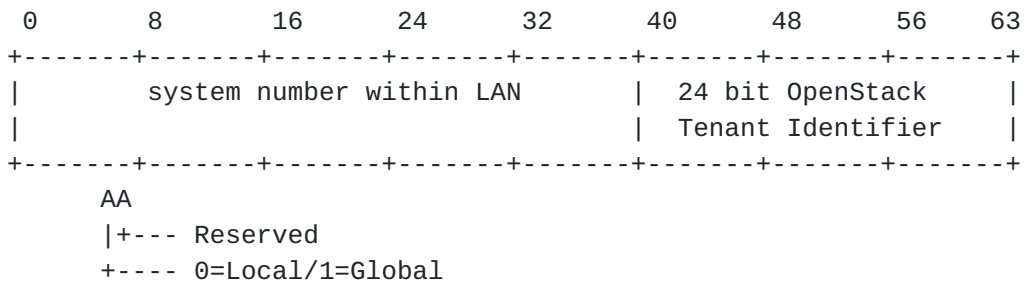

Figure 11: Generalized Tenant IID

As noted, the Tenant Identifier might be longer or shorter in a given implementation. Specifically in Figure 11, a 32 bit Tenant ID would occupy bit positions 32..63, and a 16 bit Tenant ID would occupy positions 48..63.

Following the same model, IPv6 Multicast Addresses can be associated with a tenant identifier by placing the tenant identifier in the same set of bits and using the remaining bits of the Multicast Group ID as the ID within the tenant as show in Figure 12. Flags and scope are as specified in [RFC4291] section 2.7. To avoid clashes with multicast addresses specified in ibid 2.7.1 and future allocations, The Tenant Group ID MUST NOT be zero.

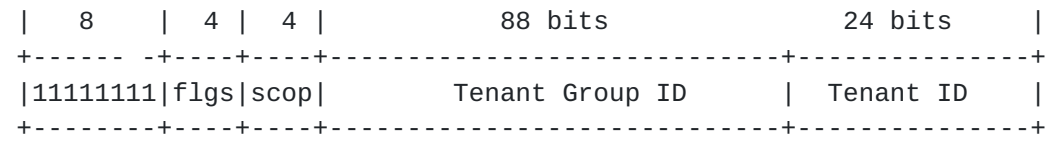

Figure 12: RFC 4291 Multicast Address with Tenant ID

B.5.2. Metaconsiderations

B.5.2.1. Service offered

The fundamental seervice offered in this model is that the key policy parameter, the Tenant ID, is encoded in every datagram for both sender and receiver, and can therefore be tested by sender, receiver, or any other party. It is secure in the sense that it cannot be directly spoofed; in the sender vSwitch, the vSwitch prevents the sender from sending another address as discussed in Section 4, and if the recipient address is a randomly chosen address, even if it meets inter-tenant communication policy, there is unlikely to be a matching destination to deliver it to. Where this breaks down is if a valid and acceptable destination is discovered and used; that is the argument for further protection via TLS.

B.5.2.2. Pros and Cons

B.5.2.2.1. The case in favor of this approach

The case in favor of this approach consists of several observations:

- Many Cisco customers are using and prefer SLAAC for IPv6 clients in OpenStack environments.
- It is easy to configure this IID from the Neutron Controller without modifying the VM, or it even knowing it happened.
- From a security perspective, the tenant ID cannot be spoofed per se; the sender of a message is not permitted to send a message from the wrong address or to an unauthorized address.
- Since it requires no extension headers or other encapsulations, it has no impact on Path MTU.
- Filters can be applied anywhere, and notably at the sender and the receiver of a message.

B.5.2.2.2. The case against

In [I-D.ietf-6man-default-iids], the IETF is moving toward deprecating [RFC4291]'s Modified EUI-64 IID.

B.5.2.3. Filtering considerations

In this model, the vSwitch needs, for each VM it manages, two access control lists:

- Zero or more {IPv6 prefix, tenant ID} pairs; these may be read as 'if the IPv6 prefix matches {prefix}, the IID contains a tenant ID, and it may be {tenant ID}.'
- Zero or more IPv6 prefixes that the VM is authorized to communicate without regard to tenant ID.

Generally speaking, one would expect at least one of those three lists to contain an entry - at minimum, the VM would be authorized to communicate with ::/0, which is to say 'anyone'.

Among the generic IPv6 prefixes that may be communicated with, there may be zero or more IPv4-embedded IPv6 prefix [RFC6052] prefixes that the VM is permitted to communicate with. For example, if the [I-D.ietf-v6ops-siit-dc] translation prefix is 2001:db8:0:1::/96, and

the enterprise is using 192.0.2.0/24 as its IPv4 address, the filter would contain the prefix 2001:db8:0:1:0:0:c000:0200/120.

Note that the lists may also include multicast prefixes as specified in Appendix B.5.1, such as locally-scoped multicast ff01::/104 or locally-scoped multicast within the tenant {ff01::/16, tenant id} . While these access lists are applied in both directions (as a sender, what prefixes may the destination address contain, and as a receiver, what prefixes may the source address contain), only the destination address may contain multicast addresses. For multicast, therefore, the vSwitch should filter Multicast Listener Discovery [RFC2710] using the multicast subset, permitting the VM to only join relevant multicast groups.

B.6. Modified IID using modified Privacy Extension

This variant reflects and builds on the IPv6 Addressing Architecture [RFC4291] Stateless Address Autoconfiguration [RFC4862], and the associated Privacy Extensions [RFC4941]. The address format is identical to that of Appendix B.5, with the exception that The 'System Number within the LAN' is a random number determined by the host rather than being specified by the controller.

It does have implications, however. It will require

- a modification to [RFC4941] to have the host include the Tenant ID in the IID,
- a means to inform the host of the Tenant ID,
- a means to inform the vSwitch of the Tenant ID,
- a the vSwitch to follow FCFS SAVI [RFC6620] to learn the addresses being used by the host, and
- a filter to prevent FCFS SAVI from learning addresses that have the wrong Tenant ID.

B.6.1. Metaconsiderations

B.6.1.1. Service offered

To Be Supplied

B.6.1.2. Pros and Cons

B.6.1.2.1. The case in favor of this approach

To Be Supplied

B.6.1.2.2. The case against

To Be Supplied

B.6.1.3. Filtering considerations

To Be Supplied

Authors' Addresses

Fred Baker (editor)
Cisco Systems
Santa Barbara, California 93117
USA

Email: fred@cisco.com

Chris Marino
Cisco Systems
San Jose, California 95134
USA

Email: chrmarin@cisco.com

Ian Wells
Cisco Systems
San Jose, California 95134
USA

Email: iawells@cisco.com

Rohit Agarwalla
Cisco Systems
San Jose, California 95134
USA

Email: roagarwa@cisco.com

Sebastian Jeuk
Cisco Systems
San Jose, California 95134
USA

Email: sjeuk@cisco.com

Gonzalo Salgueiro
Cisco Systems
Research Triangle Park, NC 27709
US

Email: gsalguei@cisco.com