Cloud Computing

Automating the Virtualized Data Center

Venkata Josyula
Malcolm Orr
Greg Page

ciscopress.com

Cloud Computing: Automating the Virtualized Data Center

Venkata Josyula
Malcolm Orr
Greg Page

Cisco Press

800 East 96th Street

Indianapolis, IN 46240

driving up asset utilization or for disaster recovery (DR) purposes is now a viable technical use case. The latter case is driving up the bandwidth requirements (today, a recommended minimum of 622 Mbps) required for VMware's VMotion/Site Recovery Manager (SRM) service. Technologies like Cisco virtual Port Channel (vPC) is ideal for Layer 2 deployments over dark fiber. Layer 2 extension technologies need to guarantee the basic operational principles of Ethernet, loop-free forwarding, no packet duplication, and MAC address table stability. In addition, the solution should provide the following:

- Flooding minimization (ARP broadcast, some unknown unicast)
- MAC mobility
- Ease of management and provisioning
- Multicast optimization

In this challenging environment, Layer 3 overlay solutions that enable fast, reliable, high-capacity, and highly scalable DCI are also essential. Such a solution is available with virtual private LAN service (VPLS), a technology that provides Ethernet connectivity over packet-switched WANs. VPLS supports the connection of multiple sites in a single bridged domain over a managed IP or IP and MPLS (IP/MPLS) network. VPLS presents an Ethernet interface, simplifying the LAN and WAN boundary for enterprise customers and helping enable rapid and flexible service provisioning. Data centers, each having their own Ethernet LAN, can be united in a VLAN over a WAN by using VPLS.

The Advanced VPLS (A-VPLS) feature introduces the following enhancements to VPLS:

- Capability to load-balance traffic across multiple core interfaces using equal-cost multipathing (ECMP)
- Support for redundant DCI and provider-edge switches

One of the most recent innovations for Layer 2 extension over IP (or MPLS) is Overlay Transport Virtualization (OTV). OTV provides Layer 2 connectivity between remote network sites by using MAC address–based routing and dynamic IP-encapsulated forwarding across a Layer 3 transport network to provide support for applications that require Layer 2 adjacency, such as clusters and virtualization. OTV is deployed on the edge devices in each site. OTV requires no other changes to the sites or the transport network. OTV builds Layer 2 reachability information by communicating between edge devices with the overlay protocol. The overlay protocol forms adjacencies with all edge devices. After each edge device is adjacent with all its peers on the overlay, the edge devices share MAC address reachability information with other edge devices that participate in the same overlay network.

OTV discovers edge devices through dynamic neighbor discovery that can leverage the multicast support of the core. This means efficient multisite Layer 2 extensions, which are ideal for the VM live migration use case. It is important to note that OTV is aimed at private cloud scenarios, as the protocol does not explicitly support per-tenant semantics. In other words, one can dedicate an overlay to a customer (maximum overlays

supported today is three on a Nexus 7000), but it cannot provide per-tenant isolation (for example, VPN). Figure 5-4 illustrates the simplified view of the OTV dynamic encapsulation of Layer 2 frames in a Layer 3 (pseudo-GRE) header for transport across a Layer 3 (IP) network.

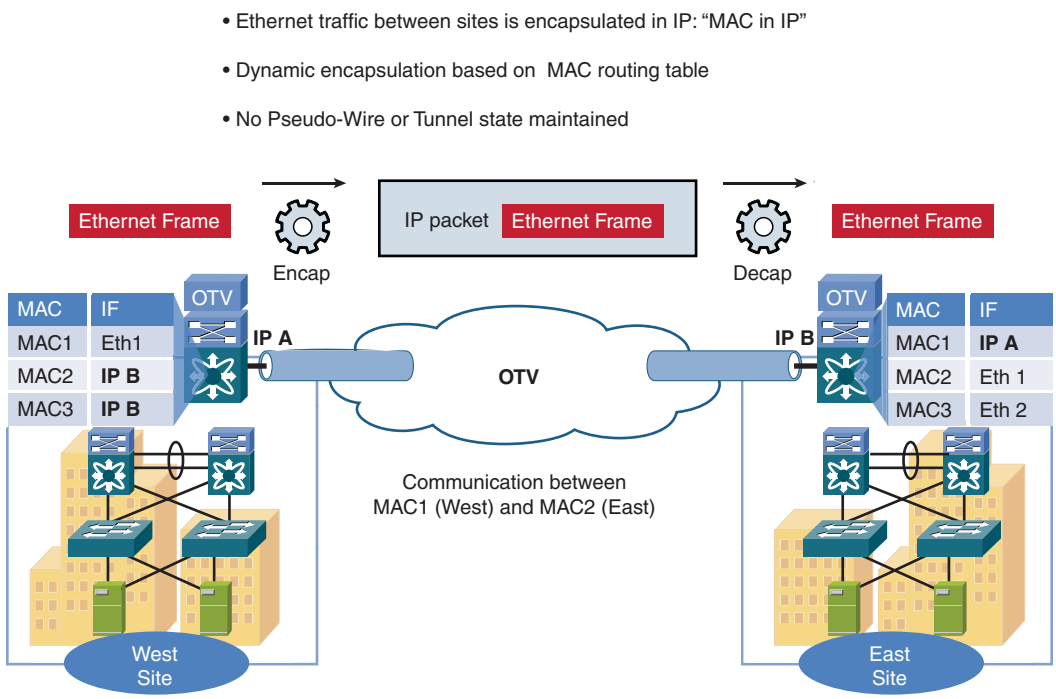

Figure 5-4 *Overlay Transport Virtualization Overview*

Enabling Machine Mobility Across Layer 3 Boundaries

The vision of the cloud for many is the ability to instantaneously "burst" into (and retract from) third-party clouds when additional infrastructure resources are needed. However, there are large parts that need to be adapted for the mechanisms to be ready to support "cold" and ultimately "live" VM migrations.

Today, if an administrator wants to spin up VMs in a cloud, he would use an API to manage the life cycle of that VM. There are plenty of cloud OS/cloud stacks that have been built to support these APIs. Examples of cloud APIs (normally using the RESTful protocol) include but are not limited to the following:

- **OCCI:** www.ogf.org/gf/group_info/view.php?group=occi-wg
- **VMware vCD API:** www.vmware.com/pdf/vcd_10_api_guide.pdf
- **Amazon EC2 (AWS) API:** http://aws.amazon.com/developertools/Amazon-EC2/351
- **ElasticHosts API:** www.elastichosts.com/cloud-hosting/api

- **FlexiScale API:** www.flexiant.com/reference/api
- **GoGrid API:** www.gogrid.com/cloud-hosting/cloud-api.php
- **Sun Cloud API:** http://kenai.com/projects/suncloudapis/pages/Home
- **OpenStack:** Using nova-manage (https://launchpad.net/openstack-dashboard) and Euca2ools (http://open.eucalyptus.com/wiki/Euca2oolsGuide_v1.1) APIs

Use of these APIs is suitable for cold VM migration from one (for example, private) cloud to another (for example, public) cloud managed and operated separately. Because each cloud has its own administrative ambience and methods, this is one of the many challenges today that restricts live VM migration between clouds. Each cloud requires unique machine images (for example, Amazon Machine Image [AMI]). There are companies that specialize in converting machine images, such as CohesiveFT. (AWS also has its own conversion service called VM Import.)

The Distributed Management Task Force (DMTF) is working on something called the Open Virtualization Format (OVF). From www.dmtf.org/standards/ovf:

DMTF's Open Virtualization Format (OVF) is a packaging standard designed to address the portability and deployment of virtual appliances. OVF enables simplified and error-free deployment of virtual appliances across multiple virtualization platforms.

OVF is a common packaging format for independent software vendors (ISV) to package and securely distribute virtual appliances, enabling cross-platform portability. By packaging virtual appliances in OVF, ISVs can create a single, prepackaged appliance that can run on customers' virtualization platforms of choice.

Note that OVF v1.1.0 supports both standard single VM packages (VirtualSystem element) and packages containing complex, multitier services consisting of multiple interdependent VMs (VirtualSystemCollection element). OVF v1.1.0 supports virtual hardware descriptions based on the Common Information Model (CIM) classes to request the infrastructure to support the running of the virtual machine(s) or appliance(s). The XML representation of the CIM model is based on the WS-CIM mapping.

Some cloud service providers offer their own CloudOS as Software as a Service (SaaS) to manage the private cloud on the end customer's premises or expose APIs, as previously mentioned.

From a "live" machine migration point of view, solving this problem at the management plane is only the first step before moving onto another challenge at the data plane (network and storage). Let's briefly discuss the challenge of OSI Layer 3 boundaries related to this goal and how Cisco is innovating with new technology, Locator/Identifier Separation Protocol (LISP), to address, at least in part, the challenge.

LISP is a "map-and-encapsulate" protocol that is currently being developed by the IETF LISP Working Group. The basic idea behind the separation is that the Internet architecture combines two functions, routing locators (where you are attached to the network) and identifiers (who you are) in one number space: the IP address (see Figure 5-5).

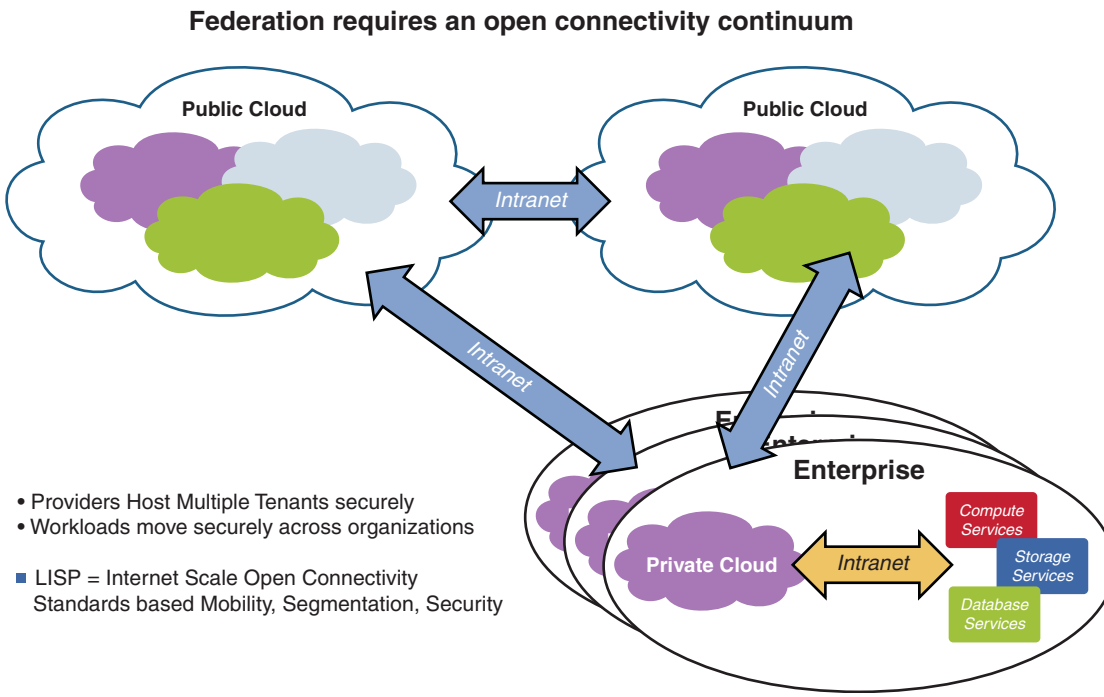

Figure 5-5 *Path from Hybrid Cloud to Federation with LISP*

Layer 3 routing has been developed over time to incorporate interadministrative domain interface points. Exterior gateway protocols like Border Gateway Protocol (BGP) have been specifically developed for this purpose. For Layer 2 connectivity between administrative domains, this is more problematic and as such is not considered as a viable option. So we need to look at Layer 3 options that can support the live VM migration use case.

A basic observation, made during early network research and development work, is that the use of a single address field for both device identification and routing is problematic. To effectively identify a device as a network session endpoint, an address should not change, even if the device moves, such as from a home to a work location, or if the organization with which the device is associated changes its network connectivity, perhaps from one cloud service provider to another. However, it is not feasible for the routing system to track billions of devices with such flexibly assigned addresses, so a device needs an address that is tightly coupled to its topological location to enable routing to operate efficiently.

To provide improved routing scalability while also facilitating flexible address assignment for multihoming, provider independence, and mobility, LISP was created. LISP describes a change to the Internet architecture in which IP addresses are replaced by routing locators (RLOC) for routing through the global Internet and by endpoint identifiers (EID) for identifying network sessions between devices. Essentially, LISP introduces a new hierarchy

(also known as "jack up") to the forwarding plane, allowing the separation of location and identity.

Note You can find more information about LISP capabilities, use cases, and deployments at www.cisco.com/go/lisp, http://lisp4.cisco.com, and http://lisp6.cisco.com.

Cisco Systems Nexus 7000 now supports LISP VM-Mobility mode (see www.cisco.com/en/US/docs/switches/datacenter/sw/5_x/nx-os/lisp/configuration/guide/NX-OS_LISP_Configuration_Guide_chapter2.html).

LISP VM-Mobility provides adaptable and comprehensive first-hop router functionality to service the IP gateway needs of the roaming devices that relocate in addition to being able to control which VMs can move through dynamic-EID prefix lists. This capability works in both Extended Subnet Mode (ESM), extending Layer 2 connectivity between data centers, and Across Subnet Mode (ASM), which allows a Layer 3 hop between data centers.

Policy Management of the Data Center Network and Services

For some time now, architects have looked at ways to try and automate the activation and change management of Data Center Networks (DCN) and network service in IaaS environments. Thus far, it has proven a difficult challenge because of the frequency and complexity of changes through traditional per-device management using scripting tools and Secure Shell command-line interface (SSH CLI) access (or even XML-based RFC 4741 NETCONF). Per-device knowledge (syntax) adds complexity, and SSH CLI access methods are serial in nature, thus causing the queuing of orchestration tasks in system management tools, which in turn slows change management tasks.

What is needed is a way to abstract from the low-level "concrete" configuration tasks to more policy-based, high-level ("abstract") system change management tasks.

The Cisco Network Hypervisor product has been designed specifically for highly virtualized environments and cloud delivery models and does for the network infrastructure what server virtualization has done for the data center—provide efficiency, elasticity, automation, and control. The virtualization capabilities provided by Network Hypervisor facilitate the transformation of static, rigid networks into a dynamic infrastructure that responds automatically to the demands of virtual and cloud environments based on the rules and business policies defined by administrators.

The network services orchestration capabilities of Network Hypervisor allow physical or virtualized computing/storage resources to be combined with network access and security models into a single service chain—a cloud service—that is fully automated and can be deployed, on demand, to selected end users. Network Hypervisor business policies define and capture the discrete elements of a cloud service and translate those elements into actual device services and configuration syntax that is automatically disseminated to the appropriate devices across the network to initiate the requested service.

From the activation of a business policy that defines a new cloud service, Network Hypervisor automatically initiates the creation of the required VMs. As the VMs are coming online, Network Hypervisor defines and deploys the network access and security models across all required infrastructure devices (routers, switches, and firewalls) as needed to deliver the cloud service to the defined end users. The entire process is completed in seconds and can include the setup and deployment of network routes, Virtual Private Networks (VPN), VLANs, and access control lists (ACL); the deployment of security certificates; and the configuring of firewall rules and DNS entries, all of which are defined through the business policy and deployed automatically without any chance of command-line mistakes.

Cisco Network Hypervisor virtualizes network services by creating or abstracting a logical network in concordance with the physical network that it also manages. It controls the physical network by virtualizing hardware switches and routers to create subnets of network addressing space, typically VPNs, that also enable and orchestrate clouds and VMs.

The logical network is driven by policies that control network access for individuals to resources. The policies specify high-level resource sharing. They can be created externally or using the Network Hypervisor Command Center as documents that can be imported, exported, and edited within the center. At the XML level, they comprise elements that model all the specifications (and more) that can be expressed using a grammar of variable and parameter substitutions that let admins and network configurators easily specify individual and multiple models that Network Hypervisor can express. For this reason, they are called metamodel files.

Figure 5-6 depicts the functional subcomponents of the Network Hypervisor and shows how it logically connects to the northbound ITSM (Information Technology Service Management) tools shown on the left and to the southbound underlying infrastructure shown on the right.

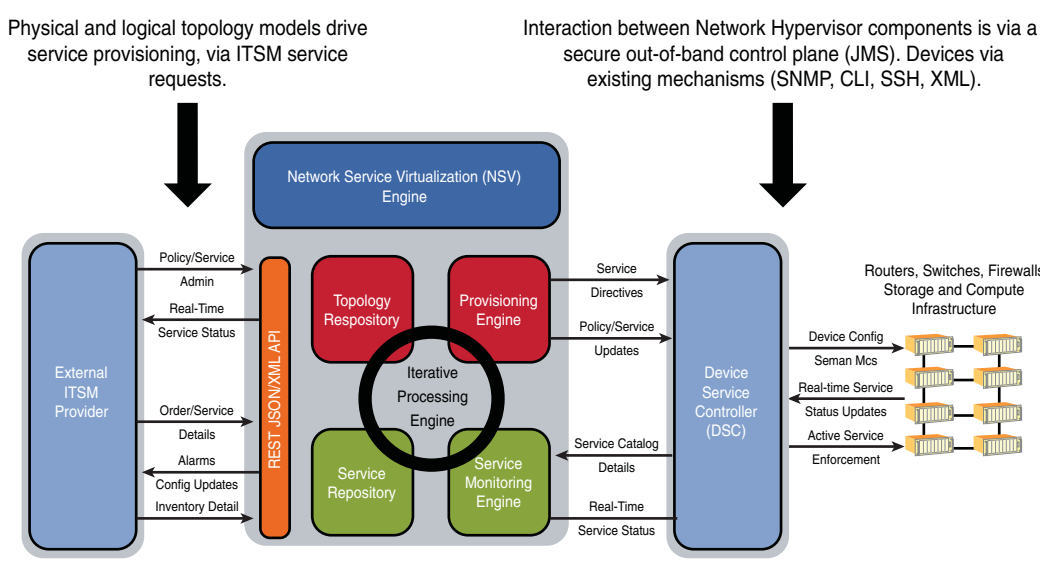

Figure 5-6 *Cisco Network Hypervisor – A Metamodel-Driven Architecture*