Using Contrail with OVSDB in Top-of-Rack Switches

Standards-Based Control and Data Plane Protocols Allow Seamless, Automated Integration between Virtual and Physical Data Center Networks
Table of Contents

Executive Summary .................................................................................................................. 3
Introduction ............................................................................................................................... 3
How Contrail Works .................................................................................................................. 3
Using OVSDB with Contrail ..................................................................................................... 5
  Contrail ToR Service Node ...................................................................................................... 5
Physical Server and VM in the Same Virtual Network .............................................................. 7
  Packet from VM to Physical Server in the Same Virtual Network ......................................... 8
  Packet from Physical Server to VM in the Same Virtual Network ...................................... 9
Physical Server and VM in Different Virtual Networks ........................................................... 10
  Packet from VM to Server in a Different Virtual Network ................................................... 12
  Packet from Server to VM in a Different Virtual Network ................................................... 14
Conclusion ............................................................................................................................... 14
Appendix A: Control Plane Initialization ................................................................................ 15
  Initialization for VM and Physical Server in the Same Virtual Network .............................. 15
  Initialization for VM and Physical Server in Different Networks ........................................ 16
About Juniper Networks .......................................................................................................... 20

List of Figures

Figure 1: Data flows from OpenStack and Contrail with encapsulation examples.................. 4
Figure 2: The ToR Service Node mediates between XMPP and OVSDB, and acts as proxy for broadcast traffic for physical servers ......................................................................................... 5
Figure 3: Control and data planes for virtual machine and physical host in the same virtual network .......................................................... 7
Figure 4: Interaction sequence for first packet from a VM to a physical host in the same virtual network .................................................... 8
Figure 5: Interaction sequence for first packet from a physical host to a VM in the same virtual network ............................................................ 9
Figure 6: Control and data plane when a gateway is employed between virtual and physical networks ................................................................................. 11
Figure 7: Interaction sequence for first packet from a VM to a physical host in a different virtual network ......................................................... 12
Figure 8: Interaction sequence for first packet from a physical host to a VM in a different virtual network .............................................................. 14
Figure 9: Initialization sequence for server and Contrail with VM and physical server in the same network .............................................................. 15
Figure 10: Initialization sequence for a physical server and virtual machine in different networks when a gateway is used ................................................................. 16
Figure 11: Interactions when Server Manager reimages a host .................................................. 18
Figure 12: Initialization sequence for switch and server ........................................................... 19
Executive Summary

Today’s data centers suffer from scalability limitations due to a restriction on the number of VLANs (4,096). Data centers also suffer from operational inflexibility because network devices must be configured a certain way in order to connect to servers running applications. Juniper Networks® Contrail removes these restrictions by creating virtual networks on hosts where applications are running. These virtual networks can be Layer 3 (MPLS over GRE or UDP) or Layer 2 (VXLAN), depending on design and architectural decisions. (Details of how Contrail implements virtual networks using overlay tunnels are described in OpenContrail Architecture Documentation and will be covered only briefly in this document.)

Introduction

Virtual networking will be widely deployed in data centers over the next several years to overcome scalability limitations and a lack of operational efficiency. However, applications running in virtual networks still have to communicate with systems on physical servers (also known as bare-metal servers) which are connected directly to Layer 2 switches. Also, many applications are deployed as virtual machines (VMs) connected to a virtual switch in a hypervisor, which is then connected to a physical Layer 2 switched network. This document will describe how standards-based control and forwarding plane protocols can enable interconnectivity between virtual and physical domains by leveraging standards-based control plane learning.

This white paper describes how control-plane learning using Open vSwitch Database (OVSDB) configured in a top-of-rack (ToR) switch can facilitate direct interconnection between VMs in a virtual network domain, which is created by Contrail and physical servers connected to a switch. Additionally, this document shows how VMs and physical servers in different virtual network domains can be interconnected using a gateway such as Juniper Networks MX Series 3D Universal Edge Routers, a Juniper Networks EX9200 Programmable Ethernet Switch, or a Juniper Networks QFX10000 switch. The gateway performs complex forwarding tasks that cannot be handled efficiently by the merchant silicon used in most of today’s switches.

Since the architecture described in this white paper is based on well-known and well supported industry standards, equipment from a number of vendors can serve as the gateway router or the ToR switch. For the purposes of this document, it is assumed that the gateway router is an MX Series router and the ToR switch is a Juniper Networks QFX5100 Switch. Each of these devices supports the required protocols and is well suited to its respective role.

Detailed control plane and data plane interactions are described for scenarios where a Layer 2-connected server is in the same virtual network domain as the VM it is communicating with, and where the Layer 2-connected server is in a different virtual network domain, which requires the use of a gateway device.

This document also includes an appendix that describes Contrail Server Manager, which allows physical servers to automatically join a predefined Contrail virtual network when OVSDB is used.

How Contrail Works

Contrail implements overlay networking using a function called Contrail vRouter located inside each hypervisor (e.g., KVM with QEMU) or operating system supporting containers (e.g., Linux with Docker). The Contrail vRouters implement user-specified network policies issued by the Contrail Controller, a centralized software system that works hand-in-hand with cloud orchestration systems such as OpenStack. Whenever a new VM is created by the cloud orchestrator, the controller is notified and configures each vRouter to support the network policies that will allow the new VM to communicate with other VMs in the same network, as well as with other networks as specified by the policies defined in the controller. These policies are implemented by installing routes to each destination that must be reachable from any given VM. The routes are installed in virtual routing and forwarding (VRF) instances created for each virtual network, and each VRF contains forwarding entries specific to one virtual network. This architecture allows network policies to be applied at the same time applications are deployed in VMs or within containers, as shown in Figure 1.
When a user requests a new VM, OpenStack submits a VM creation request to the Nova agent on the server where the VM is to reside. The Nova agent, in turn, requests network configuration via the Neutron API in OpenStack; this request is passed to the Contrail Controller via the Contrail plug-in for Neutron. The plug-in translates from the OpenStack object model and API to the object model and API of Contrail. Each API call specifies the host (physical server) on which the VM will be created, and the network to which each VM interface should belong. The Contrail Controller determines whether a new VRF must be created to support a new network on a host, configures each VRF to be attached to specific virtual interfaces of the new VM, and installs routes in each VRF to which a new interface has been attached. The installed routes provide connectivity to other systems as allowed by network policies defined in the Contrail Controller.

When a packet is sent from one VM to another, it is sent via an encapsulation tunnel, which can be MPLS over generic routing encapsulation (MPLS over GRE), MPLS over UDP, or Virtual Extensible LAN (VXLAN). Figure 1 shows the structure of packets arriving at host H1, where virtual machine VM1 is running for two of the protocols. The packets are being sent from virtual machine VMn, running on host H2 (not shown). The destination address in the outer IP header is for H1. The vRouter removes this header and examines the MPLS label or VXLAN network identifier (VNI), depending on the protocol.

In the case of MPLS over GRE or MPLS over UDP, labels are generated by the vRouter when each VM is instantiated; these labels are advertised to the Contrail Controller, which in turn includes the labels in route updates to other vRouters and physical network devices. The vRouter uses the value of the label in an incoming packet to identify which interface the stripped, original packet should be sent to without requiring a lookup of the destination IP address in the inner header. In the case of VXLAN, the VRFs in each vRouter are VXLAN Tunnel Endpoints (VTEPs), and the VNI value is the same for all of the VTEPs in a VXLAN virtual network. The VNI value in the VXLAN header of an incoming packet is used to identify which VRF should be used to look up the destination media access control (MAC) address contained in the inner header, adding an extra (small) step.

In addition to providing forwarding lookup, the vRouter acts as a proxy for several network services, including:

- **DHCP**: vRouter traps Dynamic Host Configuration Protocol (DHCP) requests and asks for IP addresses from the Contrail Controller using Extensible Messaging and Presence Protocol (XMPP). The controller then requests IP addresses from the OpenStack IP address management subsystem.
- **ARP**: vRouter responds to Address Resolution Protocol (ARP) requests from connected VMs. It will respond immediately with the target’s MAC address if it is in the receiving virtual routing and forwarding (VRF) table; otherwise, it will broadcast an ARP request to any corresponding VXLAN tunnel endpoints in its forwarding table and forward any response to the requesting VM. If an ARP response is received, it is forwarded to the requesting VM.
- **Default gateway**: The vRouter is the default gateway for all connected VMs. Any traffic not in the VM’s subnet will be sent in an Ethernet frame to the vRouter’s MAC address. The vRouter will then pass the traffic to an actual default gateway.
- **DNS**: Contrail can be configured to support Domain Name System (DNS) when an external DNS server is not present.
Using OVSDB with Contrail

This section describes how Contrail virtual networks can be extended to include physical servers connected via ToR switches that run OVSDB. The routes to MAC addresses are stored in OVSDB tables in each physical switch; OVSDB has a built-in replication capability that is used to copy routes between switches via a new Contrail component called the ToR Service Node (TSN). Each TSN mediates between Contrail’s XMPP-based protocol and the OVSDB protocol on switches so that MAC routes can be exchanged between Contrail virtual networks and physical networks configured in switches. This is described in detail in the following section.

OVSDB also supports the creation of VTEPs on switches and the assignment of interfaces to them. Contrail provides a feature that allows users to specify assignment of switch interfaces to virtual networks and, in combination with Contrail Server Manager, can provision a physical server with a specified image and connect it to a specified virtual network. This is described in detail in Appendix B: Contrail Server Manager.

Contrail ToR Service Node

The Contrail TSN provides mediation for route exchanges between OVSDB (used on network devices) and XMPP (used by Contrail). In addition, the destination for broadcast frames in the VTEPs in each switch is set to the IP address of a TSN, which provides proxy services for ARP, DHCP, and DNS. Such packets are sent to the TSN in VXLAN encapsulation. Since TSNs only deal with control plane interactions and not data plane traffic, they scale well. A single TSN can support OVSDB sessions with up to 128 switches. The scalability of the TSN solution contrasts with software gateway implementations that pass user traffic.

Figure 2 shows the major components of TSN and how they interact with OVSDB on switches and with Contrail.

![Diagram](image_url)
A TSN contains four types of software components:

- **OVSDB Client**: Each client maintains a session with the OVSDB server on a switch. Route updates and configuration changes are sent over the OVSDB session.

- **ToR Agent**: The ToR agent maintains an XMPP session with the Contrail Controller and mediates between the Contrail XMPP messages and OVSDB.

- **vRouter Forwarder**: The forwarding component of a vRouter traps and responds to broadcast packets that VTEPs in switches direct towards the IP address of a TSN inside VXLAN encapsulation.

- **ToR Control Agent**: The ToR control agent is a vRouter that provides proxy services (DHCP, DNS, and ARP) for broadcast traffic arriving over VXLAN from servers attached to switches. Response data is provided by either the ToR Control Agent itself, or by the Contrail Controller via an XMPP session.

Each TSN instance is implemented on an OpenStack compute node with Contrail Networking installed, but with the Nova compute service disabled, preventing VMs from being launched on these servers.

When a switch learns a new MAC address on an interface that is configured in a VTEP, it creates a bridge table entry for the MAC on that interface. A corresponding entry in the OVSDB table is created, which causes a MAC route to be sent via OVSDB protocol to the TSN. The route specifies a VXLAN encapsulation tunnel where the next hop is the IP address of the switch, and the VNI will be that of the VTEP on the switch to which the server is connected. When a route arrives at the TSN, it is converted to an XMPP message that is sent to the Contrail Controller, which sends the route to vRouters that have VRFs with matching VNI. The TSN also sends the routes to other switches that are running OVSDB and have VTEPs with the same VNI.

Similarly, when VMs are created using OpenStack, routes to the new VMs are sent by the Contrail Controller via the TSN to each switch with a VTEP with matching VNI. The routes specify VXLAN tunnels with the next hop being the IP address of the vRouter where the destination VM is running, and where the VNI value of the Contrail virtual network is being used. When VXLAN traffic arrives at a vRouter, the VNI is used to identify which VRF should be used for MAC lookup in order to find the virtual interface to which the inner Ethernet frame should be sent.

The routes for VMs created in OpenStack, and for gateways configured in Contrail, are distributed immediately to vRouters connected to the Contrail Controller using XMPP and OVSDB (via TSN) to switches with matching VXLANs.

In contrast, switches and routers in physical networks only learn a route to a server when the server first sends an ARP request or some other traffic. Most servers will send out some packets during booting and during initialization of applications running on them. When this happens, a switch can learn which port a server is connected to and advertise a route to that server using OVSDB. However, if a server does not send any further traffic, the routes will be removed from the bridging table after a timeout and then be withdrawn via OVSDB. Thus, when a VM first sends a packet to a server that should be reachable, according to network policy, there may not be a complete path to that server. The path will be completed as a result of ARP requests that are sent out by networking elements (servers or switches) when an L2 route is not present for the destination.

When bare metal servers are connected to virtual networks, the Contrail controller does not have the MAC-IP associations that would enable a vRouter to respond to an ARP request from a VM by proxy. When a VM makes an ARP request for the MAC address corresponding to an IP address that is on a bare metal server, the ARP must be sent to all VTEPs in the virtual network containing the IP address; those VTEPs then flood VLANs and attached servers with the request. Contrail implements edge replication to ensure that all endpoints are reached, which avoids the need to impose a high load at the originating vRouter. The controller calculates a replication tree using multiple vRouters, where each vRouter replicates the broadcast packet to other vRouters (the default is four). Some of the vRouters act as ToR service nodes (TSNs); a vRouter receiving a broadcast packet will have more destinations to send it to until all vRouters are reached. When a TSN receives a broadcast packet from a vRouter, it replicates the packet to each VTEP in the same network on switches with which it has OVSDB sessions. The replication trees are calculated whenever VMs or TSNs are added or removed, so they are already installed when a broadcast packet needs to be sent.

The interactions between the various TSN components, the Contrail Controller, and OVSDB running on switches are described in detailed packet walkthroughs in the following sections. In each case, it is assumed that the physical server has been “quiet” with respect to the network, so the bridge table in the switch does not contain an entry for the server when the first packet is sent. If the interaction flow is modified when a packet is sent before the bridge table has timed out, this is noted after the interaction list.
Physical Server and VM in the Same Virtual Network

This section describes a scenario where a physical server is connected to a switch running OVSDB for VXLAN management.

A virtual network has been configured in Contrail and a route target (RT) is associated with it. For the purposes of this example, this RT will be called “Red,” although its actual value will be in the form of target:xxx:yyy, where xxx and yyy are numeric values. A virtual machine has been launched and configured with its interface in the Red network. The VNI of the VTEP on the switch to which the server is connected matches the VNI associated with the Contrail virtual network (Red), so the physical server is in the same virtual network as the VM and they should be able to communicate.

Figure 3 shows how routes are exchanged via the TSN, which mediates between the XMPP messages used by the Contrail Controller and OVSDB used on the switch. As a convention in this document, a letter in parenthesis, e.g., (R), indicates the route target or VNI attached to a route update.

The switch configuration contains a routing instance which is of the type virtual-switch and whose interfaces are configured to be managed using OVSDB. The routing instance contains bridge domains (VLANs) that, in turn, are linked to the logical interfaces that connect to the servers. Each bridge domain also specifies the VNI that its VLAN should map to for VXLAN encapsulation, so each bridge domain becomes a VTEP. Note that, as indicated in the inset in Figure 3, the architecture is essentially identical if a VM is connected to the VTEP via a VLAN from the virtual switch in the hypervisor, since in each case, a logical interface is associated with the VTEP and control and data plane flows are identical. Although this document uses examples showing a physical server connected directly to a switch, all described interactions are also valid for VMs and physical servers connected to an OVSDB-enabled ToR using VLANs and an intermediate switched network.

In Figure 3, server S1 is running a hypervisor, with the vRouter providing network support. A VM has been created by OpenStack on server S1, and its interface was specified as being in the Red network. Contrail was notified of the VM to be instantiated, so it created a VRF and attached the VM interface to it. When the Contrail Controller receives route updates with a route target matching that of the VRF in the vRouter (RT=Red), it sends forwarding entries to the vRouter for installation in the Red VRF. For convenience, it is assumed that there is a single imported RT with value Red, although multiple RTs are in fact used by Contrail. Additionally, the VRF is associated with a VNI value that matches that of the VTEP on the switch (VNI value of Red).

After control plane initialization, the forwarding tables in the network are summarized as shown in Table 1.

<table>
<thead>
<tr>
<th>System</th>
<th>Table</th>
<th>Destination</th>
<th>Next hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>vRouter on server S1</td>
<td>Red VRF</td>
<td>IP-VM1, MAC-VM1</td>
<td>Local, IF-VM1</td>
</tr>
<tr>
<td>Switch</td>
<td>Red VTEP</td>
<td>MAC-VM1</td>
<td>S1, Encap VXLAN, VNI=Red</td>
</tr>
</tbody>
</table>

Table 1: Network Forwarding Tables
The vRouter on server S1 has routes in the Red VRF for both the IP address and MAC address of the VM via a local interface (IF-VM1). Switch contains a MAC route in its Red VTEP to VM1, with a next hop being the IP address of server S1, and using VXLAN encapsulation with a VNI value of Red. There are no routes to S2, since it is assumed that it is not sending out traffic and its bridge table entry in S2 timed out and any routes to S2 were withdrawn.

The details of control plane initialization are described in the “OVSDB: Initialization for VM and Physical Server in the Same Virtual Network” section in Appendix A.

### Packet from VM to Physical Server in the Same Virtual Network

This section describes what happens in Figure 3 when an IP packet is sent from virtual machine VM1 to physical server S2 when both are in the same virtual network and the OVSDB control plane is used. Since the packet being sent from VM1 to S2 is the first one, the network stack in VM1 does not have a MAC address corresponding to the IP address of server S2. An ARP request from VM1 for S2’s MAC address is flooded by the vRouter in VXLAN tunnels to Red VTEPs. The request reaches the network interface card (NIC) in S2, which replies with an ARP response so VM1 now has an ARP table entry linking the IP address of S2 to its MAC address. Switch SW creates a bridge table entry for S2 when it sends the ARP response, and sends a MAC route update via OVSDB to the TSN. The route update is sent via XMPP to the Contrail Controller, which sends it to the vRouter where VM1 is running. When a TCP timeout occurs in the network stack of VM1, an Ethernet frame is sent over the now complete path from VM1 to S2.

Details are shown in Figure 4, while the following list offers a step-by-step description.

1. A process in VM1 needs to send a packet to server S2. The IP packet is sent into the VM1 operating system’s network stack.
2. There is no entry for S2 in the ARP cache of the VM network stack.
3. The network stack in VM1 drops the IP packet (see footnote 1) and sends an ARP request for S2 to the vRouter on the VM network interface.
4. The vRouter hosting the VM receives the ARP request and looks up S2 in the ARP cache for the Red virtual network, but it does not have an entry for S2.
5. The vRouter uses edge replication to forward the ARP request to all vRouters with a Red VRF (including TSNs). The ARP requests are sent in VXLAN encapsulation with Red VNI.
6. Each TSN floods the ARP request to switches with which it has an OVSDP session, and which have a Red VTEP.
7. Switch SW receives the ARP request and floods the request into the VLAN corresponding to the VNI in the VTEP receiving the request.
8. ARP reply
   - SenderIP=VM1, SenderMAC=VM1, TargetMAC=S2, TargetIP=S2
9. ARP cache store
   - MAC-VM1 with IP-VM1
10. Bridge table store
    - MAC-S2 – VLAN/interface in Red VTEP
11. TCP timeout
   - Again, ARP request
12. ARP broadcast in VLAN
   - SenderIP=VM1, SenderMAC=VM1, TargetMAC=?, TargetIP=S2
13. OVSDB update
    - MAC-S2 – NH SW, Encap VXLAN, VNI=Red
14. XMPP update
    - MAC-S2 – NH SW, Encap VXLAN, VNI=Red
15. VXLAN packet
    - SrcIP=S1, DestIP=S2, Encap VXLAN, VNI=Red [SenderIP=VM1, SenderMAC=VM1, TargetMAC=S2, TargetIP=S2]
16. VXLAN packet
    - SrcIP=S1, DestIP=S2, Encap VXLAN, VNI=Red [SenderIP=VM1, SenderMAC=VM1, TargetMAC=S2, TargetIP=S2]
17. Ethernet frame
    - SrcMAC=VM1, DestMAC=S2, SrcIP=VM1, DestIP=S2, ...
18. VXLAN packet
    - SrcIP=S1, DestIP=SW, Encap VXLAN, VNI=Red [SenderIP=VM1, SenderMAC=VM1, TargetMAC=S2, TargetIP=S2]
19. Ethernet frame
    - SrcMAC=VM1, DestMAC=S2, SrcIP=VM1, DestIP=S2, ...
20. IP packet
    - SrcIP=VM1, DestIP=S2, ...

1 In this document, it is assumed that a packet is dropped in Layer 2 of the network stack if an ARP request to the network is needed, and TCP takes care of resending. It is possible that some implementations store the packet as part of the Layer 2 processing and forward it when an ARP response is received and the TCP timeout may not occur.
8. Server S2 NIC receives the ARP request, matches it against its own IP address, and sends an ARP reply.
9. Server S2 NIC places an entry for the MAC and IP addresses of virtual machine VM1 into its ARP cache.
10. The switch sends the ARP reply via unicast VXLAN to the vRouter on S1.
11. The switch caches server S2’s MAC address in the Red VTEP bridging table against the interface that server S2 is connected to.
12. The vRouter forwards the ARP reply to VM1. It does not update its own ARP table at this point.
13. The switch sends an OVSDB update to the TSN containing a route to S2 using VXLAN encapsulation with VNI value of Red.
14. TSN translates the OVSDB message to an XMPP message, which it sends to the Contrail Controller.
15. Contrail Controller sends an XMPP route update to the vRouter on S1.
16. A TCP timeout will occur in the network stack in VM1.
17. The IP packet will be resent inside the VM’s networking stack and the Ethernet frame is sent out.
18. The vRouter finds a route to S2’s MAC address. It encapsulates the frame with VXLAN using VNI value of Red.
19. The switch receives the VXLAN packet in its Red VTEP, decapsulates it, and forwards the frame to server S2’s NIC.
20. The server NIC removes the VXLAN header and forwards the IP packet into the server OS.

**Packet from Physical Server to VM in the Same Virtual Network**

In this case, a process on server S2 sends a packet to the IP address of virtual machine VM1. The server NIC will issue an ARP request with VM1’s IP address. At the switch, the frame is identified as a broadcast frame and sent over a Red VXLAN tunnel to the TSN that has been configured as the destination for such frames. The vRouter forwarding function located on the TSN acts as a proxy and responds with VM1’s MAC address, since the controller previously installed MAC and L3 routes in the TSN to all VMs so that the TSN has the MAC-to-IP address associations. A route to the VM1 MAC address was also previously installed in the Red VTEP in switch SW via OVSDB from the TSN when VM1 was created, so the switch can forward an Ethernet frame with VM1’s destination MAC address. Figure 5 shows the interactions for the first packet from S2 to VM1, while the following list offers a detailed description of the process.

Figure 5: Interaction sequence for first packet from a physical host to a VM in the same virtual network.
1. An IP packet with a destination IP address of VM1 is sent from a process running in server S2’s operating system.
2. An ARP cache miss occurs in the server NIC for IP address VM1.
3. The IP packet is dropped and server S2’s NIC sends an ARP request from its interface.
4. Switch SW receives the ARP request in the Red VTEP, identifies the frame as a broadcast frame, and sends it to the TSN (the designated service node for broadcast frames) with Red VXLAN encapsulation.
5. Switch SW creates an entry in the Red VTEP bridging table for server S2’s MAC address using the interface on which the ARP request arrived. This causes a MAC route to be advertised using OVSDB, but this is not used in this scenario.
6. The vRouter in the TSN proxies the ARP request and sends back the VMI MAC address in an ARP response, encapsulated in a VXLAN with VNI value of Red.
7. The switch receives the ARP reply in the Red VTEP, decapsulates the VXLAN wrapper, and forwards it to server S2.
8. Server S2’s NIC creates an ARP cache entry for VM1.
9. A TCP timeout occurs in server S2’s NIC.
10. The TCP/IP packet is resent within the NIC; this time an entry in the ARP cache is found for VM1’s IP address.
11. Server S2’s NIC sends an Ethernet frame to switch SW with the destination MAC address of VM1.
12. The switch looks up the MAC address of VM1 in the Red VTEP forwarding table and finds the route that was sent by the Contrail Controller when VM1 was instantiated. It forwards the frame using VXLAN to server S1, using a VTEP value of Red.
13. The vRouter decapsulates the VXLAN packet, looks up the inner MAC address in the Red VRF to find which interface to use, and forwards the frame to the VM1 network stack.

**Physical Server and VM in Different Virtual Networks**

This section describes what happens when a physical server is connected to a switch that is running OVSDB for managing VXLANs, and the server is connected to a VTEP with a different VNI (in this case, Green) than that of the VM it needs to communicate with.

Due to limitations in today’s merchant silicon, an additional gateway router is normally required to enable packets to traverse the Red and Green networks. The gateway router is configured to receive routes from both the Red and Green networks. The gateway decapsulates each packet arriving from one network, performs a lookup (MAC or IP), and then encapsulates the packet before sending it to the tunnel destination. The merchant silicon chipsets used in switches today are limited in the types of lookup they can perform on packets; this is why a router or switch containing more powerful ASICs—such as the MX Series routers, EX9200 line of Ethernet switches, or the QFX10000 line of Ethernet switches—is required as part of this solution.

Contrail Version 2.2 introduces a feature that enables it to dynamically configure MX Series routers to send traffic between network policies are created. This is done by sending commands to the router using the Network Configuration Protocol (NETCONF). The commands can create VRFs and configure them with route targets that cause the VRF of one network to import routes to destinations in another network according to the network policies configured in the controller.

Details of the control plane initialization, including VRF creation on the gateway router, are described in the section “OVSDB: Initialization for VM and Physical Server in Different Networks” in Appendix A.

Figure 6 shows how routes are exchanged via the TSN, which mediates between the XMPP messages used by the Contrail Controller and OVSDB used on the switch. The gateway router uses BGP to exchange routes with the Contrail Controller, and the VRF is configured to import routes to both Red and Green destinations.
The MX Series gateway router is configured to use Ethernet VPN (EVPN) as the control plane, and the IP addresses of control nodes in the Contrail Controller as peers. The dynamic tunnels feature is enabled for the address space of the servers hosting VMs. The dynamic tunnels feature enables the use of MPLS over GRE encapsulation for L3 VPN routes when an MPLS label-switched path (LSP) is not configured (as in the case of data centers).

When the Green network is created in Contrail, the administrator specifies that the network should be extended to physical routers and the MX Series gateway is selected. This causes Contrail to create a pair of routing instances on the gateway. One routing instance is a VRF (for L3 VPN routes) and the other is a virtual switch (for L2 EVVPN routes). The routing instances import and export routes with the Green route target. The VRF is configured to contain an integrated routing and bridging (IRB) interface whose IP address is the same as the default gateway address for the Green network, which was specified when each network was defined in Contrail, and is of the form of x.y.z.1 for a /24 network, by default. The virtual switch routing instance is configured as a VTEP with the Green VNI, and to use the IRB in the VRF for routing. The interface that the service S2 is attached through is associated with the Green VTEP.

The VTEP on the switch is configured to use a VNI value of Green. The switch has OVSDB configured and a session is maintained with a ToR Service Agent (TSA) in the TSN. The IP address of the TSN is specified as the destination for L2 broadcast traffic received by the switch. The switch advertises VXLAN routes learned in the Green VTEP using OVSDB to the TSN, and from it to the Contrail Controller using XMPP. The Contrail Controller will send the VXLAN routes via BGP (EVPN) to the gateway router using the Green RT, and the VXLAN routes will be installed in the Green virtual switch routing instance. For the purposes of this example, it is assumed that physical server S1 is quiescent and that its bridge table entry in the switch timed out and was removed. This, in turn, would cause the OVSDB entry to be removed and for the route to be withdrawn from the Green spot in the gateway router.

The routing table entries for the Figure 6 example are shown in Table 2.

<table>
<thead>
<tr>
<th>System</th>
<th>Table</th>
<th>Destination</th>
<th>Next hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>vRouter on server S1</td>
<td>Red VRF</td>
<td>IP-VM1</td>
<td>Local, IF-VM1</td>
</tr>
<tr>
<td>Gateway</td>
<td>Green VRF</td>
<td>Prefix-Green</td>
<td>Local, IF-IRB</td>
</tr>
<tr>
<td></td>
<td>Green VTEP</td>
<td>MAC-IRB</td>
<td>Local, IF-IRB</td>
</tr>
<tr>
<td>Switch</td>
<td>Green VTEP</td>
<td>MAC-IRB</td>
<td>GW, Encap VXLAN, VNI=Green</td>
</tr>
</tbody>
</table>

When the administrator creates a network policy that allows traffic to flow between the Red and Green networks, it sends routes for Green destinations to vRouters with Red VRFs. In this example, a route to the Green network prefix is installed in the Red VRF on S1. The controller also configures additional import and export in the gateway to cause the Green routing instances to import routes with the route targets of the Red network in addition to the Green routes. This results in routes to Red destinations appearing in Green routing instances on the gateway. In the example network shown in Figure 6, an IP VPN route to VM1 is installed in the Green VRF, and an L2 VPN route is installed in the Green virtual switch. The L3 route will use MPLS over GRE, and the L2 route will use VXLAN. Finally, the controller sends a MAC route for VM1 to the switch using OVSDB from the TSN. The route will have a next hop of the gateway and use VXLAN with Green VNI.
After control plane initialization, the relevant entries in the network forwarding tables are shown in Table 3.

Table 3: Network Forwarding Table Entries Following Control Plane Initialization

<table>
<thead>
<tr>
<th>System</th>
<th>Table</th>
<th>Destination</th>
<th>Next hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>vRouter on server S1</td>
<td>Red VRF</td>
<td>IP-VM1</td>
<td>Local, IF-VM1 GW, Encap GRE, Lbl=GW-G</td>
</tr>
<tr>
<td>Gateway</td>
<td>Green VRF</td>
<td>Prefix-Green</td>
<td>Local, IF-IRB S1, Encap VXLAN, VNI=Green</td>
</tr>
<tr>
<td>Switch</td>
<td>Green VTEP</td>
<td>MAC-IRB</td>
<td>GW, Encap VXLAN, VNI=Green</td>
</tr>
</tbody>
</table>

At this point, paths in both directions between VM1 and S2 are completed once packets start flowing between them.

Note that if there were other VMs in the Red and Green networks, the Controller would send routes to their vRouters with direct routes to each of the other VMs, and the gateway would not be involved for traffic between VMs in different networks. The gateway is only needed for traffic flowing between a VM and a physical server that is attached to a physical switch and which is in a different network from the VM.

Packet from VM to Server in a Different Virtual Network

When a packet is sent from VM1 to server S2, and S2 is not in the same subnet as VM1, it will be sent to the default gateway for VM1, which is the IP address in the Red virtual network of the vRouter to which it is attached. The same address is used by all vRouters in the same virtual network (e.g., x.y.z.1 for a /24 network, by default). The vRouter forwards the packet to the gateway router, since there is a route to the Green network via an MPLS over GRE tunnel.

The label in the encapsulation identifies the IRB interface on the gateway as the destination, and the packet arrives in a bridge domain which is configured in the IRB as a VXLAN endpoint with a VNI value of Green. An ARP request is sent to the other VTEPs in the Green network, including switch SW. The switch decapsulates the request, translates the VNI to a VLAN ID, and floods the VLAN with the ARP request. Server S2 receives the ARP request and sends a reply to the gateway; this sequence allows packets from VM1 to be routed first to the gateway through an MPLS over GRE tunnel, then sent in a Layer 2 VXLAN tunnel to the switch, and then via Ethernet to server S2.

Figure 7 shows the detailed sequence of interactions that allows a packet to get from VM1 to server S2, while the following list offers a step-by-step description.

1. **Prepare IP packet**
   - Send first packet to S2

2. **ARP for default gateway**
   - SenderIP=VM1, SenderMAC=VM1, TargetMAC=?, TargetIP=VR

3. **ARP reply**
   - SenderIP=VM1, SenderMAC=VM1, TargetMAC=VR, TargetIP=VR

4. **TCP timeout**

5. **Ethernet frame**
   - SrcMAC=VM1, DestMAC=VR, SrcIP=VM1, DestIP=S2, ...

6. **MPLSoGRE packet**
   - SrcIP=S1, DestIP=GW Encap GRE, Lbl=GW-G
   - [SrcIP=VM1, DestIP=S2, ...

7. **ARP cache miss for IP=S2 in Green IRB**

8. **ARP to Green VTEPs – VXLAN**
   - DestIP=S2
   - Encap VXLAN, VNI=Green
   - [SenderIP=IRB, SenderMAC=IRB, TargetMAC=S2, TargetIP=S2]

9. **ARP reply – VXLAN**
   - SenderIP=IRB, SenderMAC=IRB, TargetMAC=S2, TargetIP=S2

10. **OVSDB update**
    - MAC-S2 – NH SW, Encap VXLAN, VNI=Green

11. **VXLAN packet**
    - SrcIP=IRB, DestIP=S2
    - Encap VXLAN, VNI=Green
    - [SrcMAC=IRB, DestMAC=S2, ...

12. **Ethernet frame**
    - SrcMAC=IRB, DestMAC=S2, SrcIP=VM1, DestIP=S2, ...

13. **IP packet**
    - SrcIP=VM1, DestIP=S2, ...

Figure 7: Interaction sequence for first packet from a VM to a physical host in a different virtual network.
1. A process in VM1 needs to send a packet to server S2. The IP packet is sent to the VM1’s operating system’s network stack.

2. The packet has a destination in a different subnet. The VM1 network stack was configured during DHCP so that the vRouter is the default gateway. If the ARP cache in VM1 has flushed the vRouter’s MAC address, an ARP request will be sent by VM1.

3. The vRouter sends an ARP reply containing its own MAC address.

4. A TCP timeout occurs in the VM1 network stack.

5. A TCP retry in the VM1 network stack finds the MAC address for the default gateway and an Ethernet frame is sent out.

6. The vRouter removes the frame’s Ethernet header and looks up S2’s IP address in the Red forwarding table. The destination matches the Green prefix, and a forwarding entry is found with a next hop of the MX Series gateway, GRE encapsulation, and a label of GW-G. The packet is encapsulated and sent to the MX Series device.

7. The MX Series router receives the encapsulated IP packet and uses the label to identify that the packet should be sent to the IRB configured in the Green VRF. The IRB contains a bridge domain which is configured as a VTEP in the Green network. The ARP cache in the IRB does not have an entry for S2’s MAC address. See the note at the end of this list for a case where an ARP cache entry does exist.

8. An ARP request for S2 with the IRB MAC address as the sender MAC is sent to the other VTEPs in the Green network.

9. The ARP request is received by switch SW in the Green VTEP, which forwards it to interfaces that are in the VLAN configured in the Green VTEP.

10. Server S2’s NIC receives the ARP request, matches it against its own IP address, and sends an ARP reply.

11. Server S2’s NIC places an entry for the MAC and IP addresses of virtual machine VM1 into its ARP cache.

12. The switch receives the ARP reply and caches server S2’s MAC address in the Green VTEP bridging table against the interface to which server S2 is connected.

13. An ARP reply is sent via unicast VXLAN to the MX Series gateway router.

14. The ARP reply is received by the bridge domain configured as a VTEP in the IRB in the Green VRF on the MX Series gateway router. An ARP cache entry for S2 is created in the IRB.

15. The switch sends an OVSDB update containing a MAC route to S2 with the next hop of its own loopback address and using VXLAN encapsulation with a VNI value of Green.

16. The TSN receives the OVSDB update, converts it to an XMPP message, and sends that to the Contrail Controller, which then converts the message to a BGP update that is sent to the gateway router using the Green RT.

17. A TCP timeout will occur in the VM1 network stack.

18. The Ethernet frame will be resent by the networking stack.

19. The vRouter encapsulates the packet and sends it to the MX Series gateway.

20. The incoming encapsulated packet at the MX Series gateway is decapsulated and the label is used to determine that the packet should be sent to the IRB configured in the Green VRF. This time the local ARP table has an entry for S2, and its MAC address is used to look up a route. The route has a next hop of switch SW, VXLAN encapsulation, and a VNI value of Green. The packet is encapsulated and sent to the switch SW.

21. The packet is decapsulated in the switch and a lookup in the Green VTEP bridging table identifies which interface should be used to send the inner Ethernet frame.

22. The Ethernet frame is received at the server S2 NIC, which removes the Ethernet header to form an IP packet and sends it on to the operating system.

Note: Traffic could be sent from VM1 to S2 while the bridge table in the switch contains an entry for S2, if the server has been sending network traffic. An active L2 route will exist in the vRouter on S1 as a result of OVSDB, BGP, and XMPP updates. In this case, steps 13 and 14 will have occurred prior to the first packet being sent from S2 to VM1.
Packet from Server to VM in a Different Virtual Network

The first packet sent from S2 with the VMI as its destination will have an ARP cache miss in the server NIC. An ARP request for VMI will be replied to by the gateway, with the MAC address of VMI as the target since it was given the association between the IP and MAC addresses for VMI by the controller. Once that MAC address of VMI is known by the S2 NIC, the forwarding tables create a continuous path from S2 to VMI via Ethernet, VXLAN, and MPLS GRE.

Figure 8: Interaction sequence for first packet from a physical host to a VM in a different virtual network.

1. An IP packet is sent from a process in server S2’s operating system with VMI’s IP address as its destination.
2. The packet’s destination is in a different subnet. S2’s network stack was configured during DHCP to use an address that is shared by the vRouters in the Green network and the IRB on the router. If the ARP cache has flushed the MAC address, an ARP request will be sent by S2.
3. Switch SW receives the ARP request and sends it to the TSN, which is the destination configured for broadcast Ethernet frames.
4. The TSN is programmed not to reply to ARP requests that come from the network. Instead, it floods the request to Green VTEPs.
5. The gateway receives the ARP request in the bridge domain contained in the IRB, which is in the Green VRF. The gateway sends a reply to switch SW with the IRB’s MAC address as the target (MAC-GW) using unicast VXLAN.
6. The switch forwards the ARP reply to server S2.
7. Server S2’s NIC receives the ARP reply and creates an ARP cache entry for VMI using the MAC of the IRB on the gateway.
8. A TCP timeout for the original packet occurs in server S2’s NIC.
9. A TCP/IP packet is resent within the NIC; this time an entry for the VMI’s IP address is found in the ARP cache.
10. The NIC sends an Ethernet frame to switch SW, with destination MAC of VMI.
11. The switch receives the Ethernet frame and looks up the MAC address of VMI in the Green VTEP forwarding table. A route is returned with a next hop of the gateway with VXLAN encapsulation and a VNI value of Green.
12. The gateway receives the VXLAN packet, decapsulates it, and looks up the MAC address of VMI in the Green virtual switch forwarding table. It finds a route with next hop of server S1 and a VNI value of Red.
13. The switch encapsulates the packet and sends it to server S1.
14. The encapsulated packet is received by the vRouter on S1, decapsulated, and the VNI value is used to identify which VRF should perform forwarding lookup. The MAC address of VMI is looked up, and a route is found via the interface that VMI is connected to. The packet is forwarded to VMI.

Conclusion

Today’s data centers suffer from scalability limitations and operational inflexibility. Juniper Networks Contrail removes these restrictions by creating virtual networks on hosts where applications are running. Using Conrail with OVSDB in ToR switches allows seamless, automated integration between virtual and physical data center networks. Control plane and data plane flows can satisfy a number of use cases where Contrail virtual networks must communicate with physical servers or virtual machines that are directly connected to switches or indirectly connected on VLANs. OVSDB can be used as a control plane, and MX Series routers, when used as a gateway, can facilitate traffic flows between VMs and physical devices on different virtual networks.

The TSN received the IP-MAC association of the gateway via EVPN and XMPP. However, it does not currently use this information to perform ARP proxy (Contrail V3.0), so it has to flood the request and let the gateway respond.

©2016, Juniper Networks, Inc.
Appendix A: Control Plane Initialization

Control plane interactions that occur before the first packet is sent from a VM or physical server are described in the following sections for each of the network scenarios described in the body of this document:

- OVSDB where the VM and physical server reside in the same virtual network
- OVSDB where the VM and physical server reside in different virtual networks

Initialization for VM and Physical Server in the Same Virtual Network

Figure 9 shows the control plane initialization, while the following list provides a step-by-step description of the process.

1. Server S2 is connected to switch SW and is turned on.
2. Server NIC is initialized.
3. NIC sends out a DHCP request to get its IP address.
4. Switch forwards the DHCP request using VXLAN to the ToR service node which is specified as the destination for broadcast Ethernet frames in the Green VTEP.
5. Switch creates an entry in the Red VTEP’s bridge table for the MAC address of S2 against the interface that the DHCP packet arrived on.
6. ToR service node responds over VXLAN with the IP address associated with the interface that the server is connected to (allocated when the logical interface was added to the Red virtual network in Contrail).
7. The switch decapsulates the VXLAN packet and forwards the DHCP response to server S2.
8. The NIC updates the OS with the IP address, gateway address, and other received data such as the DNS server address.
9. During the boot sequence, the server will send traffic into the network as a result of setting up service such as Network Time Protocol (NTP) and Network File System (NFS).
10. Once the boot sequence is complete, the server will become quiescent on the network, the bridge table entry for S2 in the Red VTEP will time out, and the entry will be deleted.
11. The process for creating a VM is initiated in OpenStack.
12. OpenStack selects a server that the new VM will run on (host S1) and makes an API call to the Nova agent running on that compute node to initiate creation of the new VM.
13. OpenStack makes an API call via the Neutron API to the Contrail Controller. The API call specifies which server the VM will be created on (S1), and that the VM should be placed in the Red network.
14. The Contrail Controller sends an XMPP update that informs the vRouter on server S1 that a virtual machine (VM1) will be created with an interface whose IP address is in the Red virtual network.

15. The new VM boots and sends a DHCP request for an IP address.

16. The vRouter sends a request in XMPP to the Contrail Controller.

17. The Contrail Controller gets an IP address from the designated IP address manager for the Red network and sends it to the vRouter.

18. The vRouter traps the DHCP request and sends back a DHCP response to VM1 using the IP address that was sent by the Contrail Controller during the instantiation of the Red VRF.

19. The vRouter learns the MAC address of VM1 during the DHCP exchange and sends an XMPP message to the controller that contains the MAC and IP addresses of the new VM, together with a label for use in L3 VPN routes.

20. The Contrail Controller creates a MAC route for VM1 using VXLAN, and sends it as an XMPP update to the TSN.

21. TSN converts the XMPP update to an OVSDB update, which is then sent to the switch. This update is encapsulated as VXLAN with VNI of Red and contains a route to VM1, whose next hop is server S1; it uses VXLAN encapsulation with VNI of Red.

After the initialization process, there are local IP and MAC routes to VM1 in the vRouter, and a MAC route via VXLAN to VM1 in the switch. However, due to the bridge table timeout, there are no routes to server S2 in the switch or the vRouter. The initial forwarding tables are summarized in the following table.

Table 4: Initial Forwarding Tables

<table>
<thead>
<tr>
<th>System</th>
<th>Table</th>
<th>Destination</th>
<th>Next hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>vRouter on server S1</td>
<td>Red VRF</td>
<td>IP-VM1, MAC-VM1</td>
<td>Local, IF-VM1</td>
</tr>
<tr>
<td>Switch</td>
<td>Red VTEP</td>
<td>MAC-VM1</td>
<td>S1, Encap VXLAN, VNI=Red</td>
</tr>
</tbody>
</table>

Initialization for VM and Physical Server in Different Networks

Figure 10 shows the control plane flow as systems are installed and configured, while the following list provides a step-by-step description of the process.

Figure 10: Initialization sequence for a physical server and virtual machine in different networks when a gateway is used.
The sequence for booting the server S2 is the same as for the previous use case and is not described here. (Server 2 is connected to switch SW, turned on, and receives an IP address via DHCP.)

1. The gateway router is configured to support BGP peering with the Contrail Controller.
2. Contrail is configured to peer with the gateway router, and the gateway is configured in Contrail as a physical router. Red and Green networks are created, and the Green network is extended to the gateway router. A network policy is created and applied that allows traffic to flow between the Red and Green networks.
3. A BGP session is established between the Contrail Controller and the gateway router, and the NETCONF credentials for the gateway router are entered into the Contrail Controller.
4. Contrail configures a VRF and virtual switch on the router that exports routes with the Red route target and imports routes with both Red and Green route targets. An IRB interface is created with the address of the default gateway in the Green network. This is the same address used by VMs running on vRouters.
5. The gateway router sends BGP updates to the Contrail Controller containing L3 VPN routes to the Green network.
6. The process for creating a VM is initiated in OpenStack.
7. OpenStack selects a server that the new VM will run on (host S1) and makes an API call to the Nova agent running on that compute node to initiate creation of the new VM.
8. OpenStack makes an API call via the Neutron API to the Contrail Controller. The API call specifies which server the VM will be created on (S1), and that the VM should be placed in the Red network.
9. The controller sends an XMPP update that informs the vRouter on server S1 that a virtual machine (VM1) will be created with an interface whose IP address is in the Red virtual network.
10. The new VM boots and sends a DHCP request for an IP address.
11. The vRouter traps the DHCP request and sends back a DHCP response to VM1 using the IP address for the interface that was sent by the Contrail Controller when the Red VRF was being configured.
12. The vRouter learns the MAC address of VM1 during the DHCP exchange and sends an XMPP message to the Contrail Controller containing the MAC and IP addresses of the new VM, together with a label for use in L3 VPN routes.
13. The Contrail Controller sends a BGP update to the gateway router containing L3 and MAC VPN routes to VM1.

After equipment installation and control plane initialization, the forwarding tables in each system are shown in Table 5.

<table>
<thead>
<tr>
<th>System</th>
<th>Table</th>
<th>Destination</th>
<th>Next hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>vRouter on server S1</td>
<td>Red VRF</td>
<td>IP-VM1</td>
<td>Local, IF-VM1</td>
</tr>
<tr>
<td></td>
<td>Red VRF</td>
<td>Green prefix</td>
<td>GW, Encap GRE, Lbl=GW-G</td>
</tr>
<tr>
<td>Gateway</td>
<td>Green VRF</td>
<td>IP-VM1</td>
<td>S1, Encap GRE, Lbl=S1-R</td>
</tr>
<tr>
<td></td>
<td>Green VRF</td>
<td>Green prefix</td>
<td>Local, IRB</td>
</tr>
<tr>
<td>Switch</td>
<td>Green VTEP</td>
<td>IP-VM1</td>
<td>GW, Encap VXLAN, VNI=Green</td>
</tr>
</tbody>
</table>

Appendix B: Contrail Server Manager

Contrail Server Manager provides a flexible, easy-to-use method for managing the images loaded onto physical servers, and for automatically attaching physical servers into Contrail virtual networks when OVSDB is used in ToR switches. Server Manager provides a convenient high-level object model and API that wraps functionality provided by the underlying open source software packages: Cobbler (for provisioning images onto servers) and Puppet (for loading a base package onto a provisioned server).

The Cobbler component of Server Manager allows users to provision physical servers with software images using the industry standard Preboot Execution Environment (PXE); this is the component of Server Manager used here.

A user can use the Contrail GUI or API to select which image, stored in Server Manager, should be associated with the MAC address of an interface on a server. The user also configures the virtual networks into which the switch interfaces connected to the server should be placed. OVSDB is used to associate interfaces with VTEPs in the switch.

When a server is booted, with PXE enabled in the server BIOS, each interface performs a DHCP request to get an IP address, with the option to request the IP address of a PXE server. When the DHCP request is received in the switch that the server is connected to, the frame is sent to the TSN (since it is configured as the broadcast destination). This
is done using unicast VXLAN with a VNI of the virtual network that was specified for the interface to which the server is connected. As described in the “Using OVSDB with Contrail” section, OVSDB also supports the creation of VTEPs on switches and the assignment of interfaces to them. Contrail also includes a feature that allows users to assign switch interfaces to virtual networks and, in concert with Contrail Server Manager, provision a physical server with a specific image and connect it to a specific virtual network.

As described in the “Contrail ToR Service Node” section, each TSN contains a vRouter forwarding instance that contains VRFs for each VXLAN present on connected switches. The VRF for the VXLAN that the booting server is connected to will receive the DHCP request, trap it, and make a proxy request to the Contrail Controller. It will respond with an IP address in the associated virtual network for the interface MAC, and with the IP address of Server Manager as the PXE server. Both responses are sent by the TSN to the requesting server. The server then sends a request to the PXE server for the location of a boot image, which is then downloaded via Trivial File Transfer Protocol (TFTP); the server then boots from it. Note that the IP address of Server Manager is usually in a management network, and it is accessed using a different interface than that used for data traffic and the one that received the DHCP/PXE response.

The high-level flow for reimaging a server with one interface in a virtual network is shown in Figure 11.

The process for reimaging a server is as follows:

1. Image files are loaded into Server Manager using scripts, API, or GUI.
2. User makes a provisioning request via GUI or API for a Red VTEP to be created on the switch and for the interface on the switch to which the server is connected to be placed in the Red VTEP.
3. A configuration request for the VTEP is sent using XMPP from the Contrail Controller to the TSN.
4. The TSN converts the XMPP request to OVSDB and sends it to the switch. The Red VTEP is created and configured to contain the interface to which the server is connected.
5. User makes a provisioning request via GUI or API for a server to be provisioned with a specified image.
6. The association of MAC address to a stored image is sent via REST API to Server Manager.
7. Server Manager automatically issues a restart command via Intelligent Platform Management Interface (IPMI) to the server.
8. The server restarts and tries PXE boot ahead of booting from disk. The server sends a DHCP request that includes a PXE boot option to the switch where it is received in the Red VTEP. The VTEP is configured to send broadcast frames to the TSN.
9. The TSN contains a vRouter with a VRF associated with the Red VXLAN, so it receives the DHCP request. The vRouter traps the request and sends a proxy request to the Contrail Controller via XMPP, which responds with an IP address for the server and the address of the boot server (Server Manager).

10. The TSN forwards the DHCP and PXE responses via VXLAN to the switch, which decapsulates them and sends them to the server.

11. The server requests a boot image from Server Manager.

12. Server Manager sends the requested image to the server, which then loads it and boots from it.

After this process is complete, the server is active with the specified image running, and its interfaces have addresses in the specified virtual networks.

Figure 12 shows the detailed information flow when a switch is configured to run OVSDB and where a server is reimaged by Server Manager.

**Figure 12: Initialization sequence for switch and server.**

1. Switch is installed and configured to run OVSDB.

2. User identifies the switch in Contrail as running OVSDB.

3. The Contrail Controller launches a ToR Service Agent (TSA) in the TSN.

4. TSN establishes an OVSDB session with the switch.

5. User adds a VXLAN into Contrail by specifying a VNI for a virtual network, and specifies which switch interfaces are to be attached to it. The MAC addresses of servers attached to these interfaces are also supplied.

6. The Contrail Controller uses the REST API to update TSN with the new switch and interface information for the VXLAN.

7. TSN sends the VXLAN updates to the switch via OVSDB.

8. The Contrail Controller user configures server S2 for reimaging; its MAC address is associated with an image stored in Server Manager, and its interfaces are assigned to virtual networks.

9. The Contrail Controller uses the REST API to update Server Manager with the new server image information.

10. Server Manager uses IPMI to force a server restart.

11. During boot, the interface of the server sends a DHCP request for an IP address. The DHCP request includes PXE-specific options.

12. The switch identifies the incoming packet as a broadcast and matches this against an entry in the OVSDB database that specifies that such traffic should be sent in a VXLAN packet to the TSN.

13. The DHCP packet arrives at the TSN and its vRouter forwarding component passes it into the VRF with matching...
VNI. The vRouter proxies the DHCP request and sends an XMPP message containing the DHCP request information to the Contrail Controller.

14. The Contrail Controller responds with an IP address from the designated IP address manager for the Red virtual network, and with the IP address of Server Manager as the PXE boot server.

15. TSN sends the DHCP response back to the switch inside a VXLAN packet with VNI value of Red.

16. The switch decapsulates the DHCP response and sends it to the server.

17. The server contacts Server Manager as its PXE boot server and requests a boot image.

18. Server Manager looks up the MAC address of the server and identifies which image the server should boot from; it then sends it the name of the image file.

19. The server loads the image and reboots.

About Juniper Networks

Juniper Networks is in the business of network innovation. From devices to data centers, from consumers to cloud providers, Juniper Networks delivers the software, silicon and systems that transform the experience and economics of networking. The company serves customers and partners worldwide. Additional information can be found at www.juniper.net.