Virtual Switches

Yao-Min Chen

Outline

- Virtual Networks and Virtual Switches
- Xen Networking and Bridging
- XenServer Networking
- Open vSwitch

Virtual Network

- A virtual network (VN) is a network of VMs running on a single physical machine
- The VMs are connected logically to each other so that they can send data to and receive data from each other.
- VMs can be connected to the VNs in the procedure to “add a network.”
- Each VN is serviced by a single ***virtual switch***.

Virtual Network (Cont'd)

- A VN can be connected to a physical network by associating one or more network adapters (***uplink adapters***) with the VN's virtual switch.
  - If no uplink adapter is associated, all traffic on the VN is confined within the host machine.
  - Otherwise, VMs connected to that VN are also able to access the physical networks connected to the uplink adapters.

Virtual Switch (vSwitch)

- A vSwitch works much like a physical Ethernet switch.
- It detects which VMs are logically connected to each of its virtual ports and uses that information to forward traffic to the correct virtual machines.
- A vSwitch can be connected to physical switches using physical Ethernet adapters, also referred to as uplink adapters, to join virtual networks with physical networks.

History of vSwitch in Xen

- Xen networking relies on dom0, not hypervisor
- Bridging was used first, which was based on Linux bridging
- Just like Ethernet bridging, Xen bridging suffers scalability issue
- Community has moved towards vSwitch (also b/c pressure from vSphere)

Xen Networking and Bridging

MAC Addresses

- By default, randomly assigned by xend
- Static assigned by mac= option to the vif configuration directive (e.g. vif = ['mac=00:16:3e:aa:00:11'])
- Select “locally administered”, “unicast” addresses *xy:xx:xx:xx:xx:xx* where y=2, 6, A, or E (see next slide)
- Reserved range for Xen 00:16:3e:xx:xx:xx

Xen Networkng (1)

- Eight pairs of connected virtual Ethernet interfaces for use by dom0
- veth0 is connected to vif0.0, veth1 is connected to vif0.1, etc, up to veth7 -> vif0.7.

DomU Virtual Interfaces

- Each VM has a domain ID 1, 2, 3, … according to the sequence the VMs are created
- For each new VM, Xen creates a pair of "connected virtual Ethernet interfaces", with one end in the VM and the other in dom0.
- For example, eth0 (resp., eth1) of domU 1 is the virtual Ethernet interface connected to dom0 vif1.0 (resp., vif1.1).

Xen Bridging Packet Flow

- Packet arrives at NIC 0, is handled by dom0 Ether driver
- Packet appears on peth0.
- Interface peth0 is bound to the bridge, so the packet is passed to the bridge
- A number of vifX.Y interfaces are connected to the bridge. The bridge decides where to put the packet based on the receiver's MAC.

Xen Bridging Packet Flow (Cont'd)

- The vif interface receiving the packet puts the packet into hypervisor.
- Hypervisor puts the packe to the domain the vif leads to
  - E.g., vif1.0 receives the packet then forwards to Domain 1.

Admin Commands (1)

- "brctl"
  - set up, maintain, and inspect the Ethernet bridge configuration in the Linux kernel
  - e.g., "brctl show"
- "arp"
  - manipulates system's ARP cache
  - e.g., "arp –a"
- "ifconfig"
  - e.g., "ifconfig xenbr0"

XenServer Networking

XenServer Networking

- Default
  - Linux Bridging
- 5.6 FP1 (currently in Beta)
  - Option to use Open vSwitch (OVS)
  - “xe-switch-network-backend openvswitch” to enable open vswitch

Terminology Defined

- VIF – virutal interface, i.e., virtual NIC
- PIF – physical interface, i.e., real NIC
- Virtual switch – virtual Ethernet segment

Virtual Switching

- *Virtual Switches* - XenServer networking is accomplished by connecting the VIFs and optionally one PIF to a virtual switch or bridge.

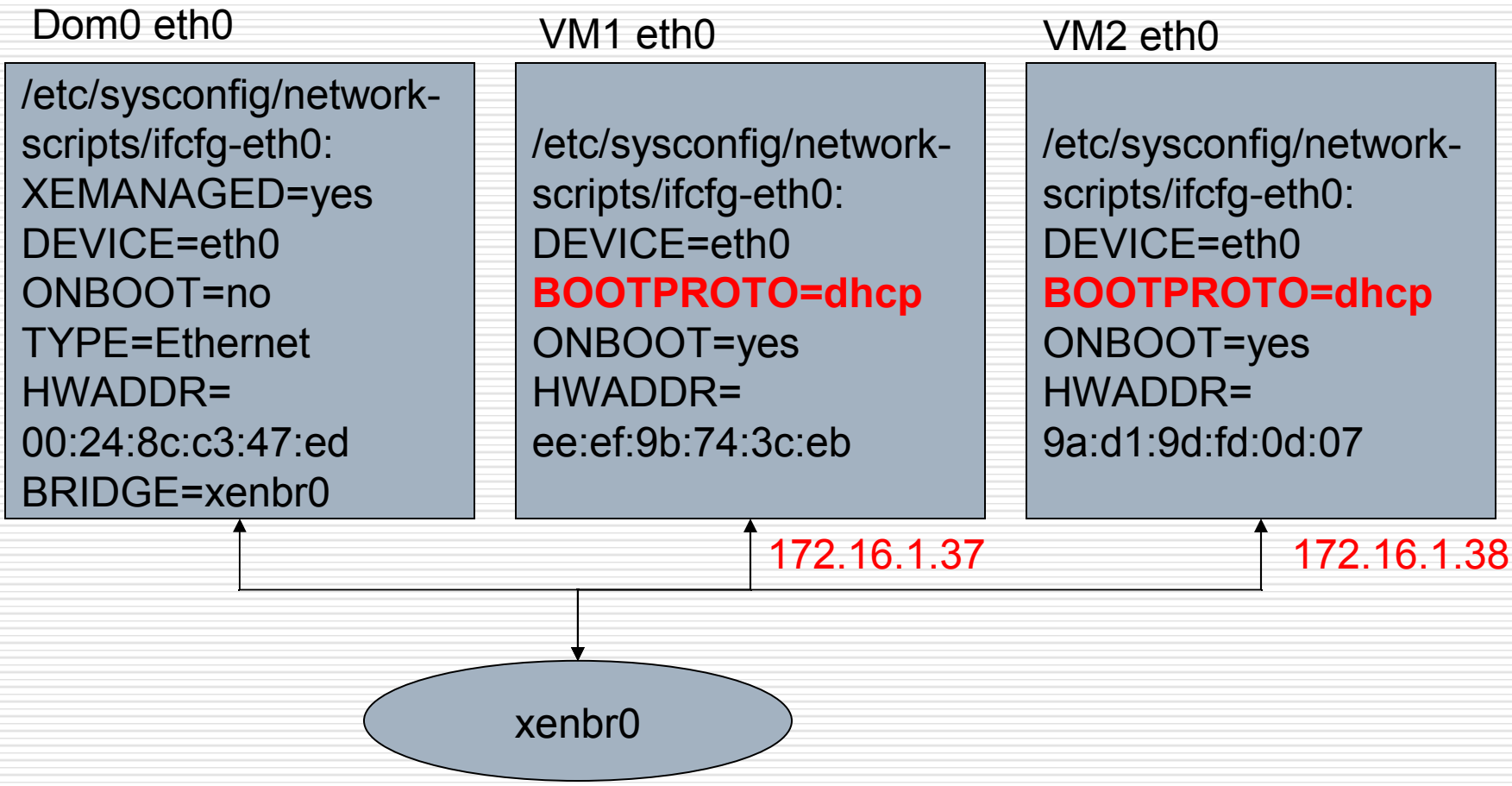
Multiple Virtual Networks

- Multiple networks can be created
- Note that each network has either one or zero PIF
- Only way for two networks to talk each other is for a VM to have VIFs on both networks (i.e., the VM routes traffic between the two networks)
- I.e., the VM has two VIFs connecting to the two virtual switches, resp.

vSwitch Naming Convention

- Virtual switch <id> is named xenbr<id> if it is used join an external network
- It is named xapi<id> if joining an internal network

Using DHCP with VIFs



Using DHCP with the Bridge

- /etc/sysconfig/network-scripts/ifcfg-xenbr0

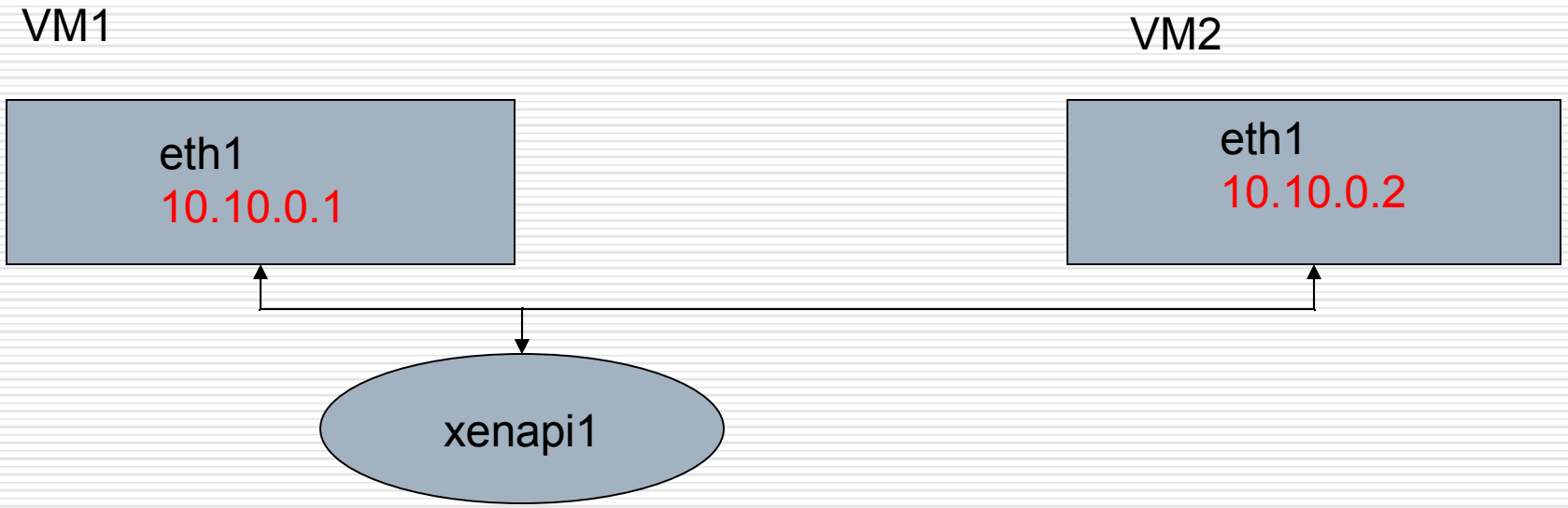
```
XEMANAGED=yes
DEVICE=xenbr0
ONBOOT=no
TYPE=Bridge
DELAY=0
STP=off
PIFDEV=eth0
BOOTPROTO=dhcp
PERSISTENT_DHCLIENT=yes
MTU=1500
```

Using Static IP with the Bridge

- /etc/sysconfig/…/ifcfg-xenbr0

```
XEMANAGED=yes
DEVICE=xenbr0
ONBOOT=no
TYPE=Bridge
DELAY=0
STP=off
PIFDEV=eth0
BOOTPROTO=none
NETMASK=255.255.0.0
IPADDR=172.16.0.102
GATEWAY=172.16.0.1
MTU=1500
DNS1=68.94.156.1
DNS2=68.94.157.1
```

Virtual Internal Network




```
[root@VM2]#brctl show
bridge name  bridge id          STP enabled     Interfaces
xapi1        8000.feffffffffff      no              vif1.1
                                                    vif2.1
xenbr0       8000.00248cc347ed   no              eth0
                                                    vif1.0
                                                    vif2.0
```

XenServer MAC Addressing

- Internal MAC (FE:FF:FF:FF:FF:FF) and external MAC
- The PIF is in promiscuous mode so it won't drop frames not destined to its own MAC

XenServer Network IP

- The IP address is with the virtual switch (bridge, e.g., xenbr0), not with the PIF (e.g., eth0).
- Any packet destined for the IP address will be processed by the bridge, and then the host OSI layers, since the MAC destination on that frame will be that of the bridge.
  - Ssh to the bridge device takes you to dom0
- The PIF won't process the IP packet, since it doesn't have an IP address assigned.

Dom0 Network Bridge Startup

- Actions taken by network-bridge startup script
  - Creates a new bridge named xenbr0
  - “Real" Ethernet interface eth0 is brought down
  - IP and MAC addresses of eth0 are copied to virtual network interface veth0
  - “Real” interface eth0 is renamed peth0
  - Virtual interface veth0 is renamed eth0
  - Attach peth0 and vif0.0 to bridge xenbr0
  - Bridge xenbr0, “real” interface peth0, “virtual” interface eth0 and vif0.0 are brought up

DomU Network Startup

- When a domU <id#> starts up, xend (running in dom0) runs the vif-bridge script, which:
  - attaches vif<id#>.0 to xenbr0
  - vif<id#>.0 is brought up

Custom Bridge (Xen only)

- You can change the bridge name from xenbr0 to *mybridge* using:
- Config xend-conf.sxp
  - (network-script 'network-bridge bridge=*mybridge*')
- Configure the bridge to attach to in the domU's config file using:
  - vif=['bridge=*mybridge*'] or something like:
  - vif=['mac=00:16:3e:01:01:01,bridge=*mybridge*']
- Reboot or restart xend

VIF Promiscuous Mode

- http://docs.vmd.citrix.com/XenServer/5.6.0
- *xe vif-param-set uuid=<VIF UUID> other-config:promiscuous="on"*
- Or, *xe vif-param-set uuid=<VIF UUID> other-config:promiscuous="true"*
- Or, set time out for address from the Forwarding DataBase (fdb) to 0
  - *brctl setageing xenbr0 0*
- Then set promisc mode on VM's interface
  - *ifconfig eth0 promisc*

Open vSwitch

Open vSwitch

- Openvswitch.org
- Started out a derivative project from Stanford's OpenFlow project (www.openflow.org)
- Incorporated into XenServer 5.6 FP1 as a strong open-source competitor to vSwitch in vSphere (VMware)

Open vSwitch Architecture

- Run in XenServer dom0
- Upon start, 6 daemon processes
  - **ovsdb-server** (1 monitor + 1 worker)
  - **ovs-vswitchd** (1 monitor + 1 worker)
  - **ovs-xenserverd** (1 monitor + 1 worker)
- Monitor will launches a new worker if previous worker dies

Open vSwitch Daemons

- **ovs-vswitchd**
  - Daemon that manages and controls any number of OVS switches on the host.
- **ovs-xenserverd**
  - Open vSwitch daemon for XenServer-specific functionality
- **ovsdb-server**
  - usage: ovsdb-server [OPTIONS] DATABASE
  - where DATABASE is a database file in ovsdb format (JSON)

Optional OVS Daemons

- ovs−brcompatd
  - Bridge compatibility front-end for ovs−vswitchd
- ovs−controller
  - Simple OpenFlow controller reference implementation
- ovs−discover
  - OpenFlow controller discovery utility
- ovs−openflowd
  - Implements an OpenFlow switch using a flow-based datapath

Open vSwitch Utilities

- **Ovs-vsctl -** utility for querying and configuring **ovs–vswitchd**
  - "ovs-vsctr list-br" to see a list of Xen bridges (xenbr1, xenbr2, xenbr3, xenbr4, xapi3)
  - "ovs-vsctl br-to-vlan" to see bridge to vlan mapping, e.g., 'ovs-vsctl br-to-vlan xapi3' shows vlan 5 mapped to xapi3
  - "Ovs-vsctl list-ports" to see ports associated with a bridge, e.g., 'ovs-vsctl list-ports xapi3' shows vif1.2 and vif2.2 are associate with xapi3

OVS Utilities (Cont'd)

- **ovs−appctl** - control OVS daemons: ovs−vswitchd, ovs−openflowd, ovs−controller, ovs−brcompat, ovs−discover
  - E.g., 'ovs-appctl vlog/list' lists logging levels
  - E.g., 'ovs-appctl fdb/show xenbr0' lists each MAC address/VLAN pair learned by bridge xenbr0

Centralized Management

- Open vSwitch exposes a number of interfaces that enable centralized mgmt
- The centralized mgmt requires a centralized controller that could be built on top of an OpenFlow controller such as NOX (noxrepo.org)
- In XenServer 5.6 FP1, Distributed Virtual Switch Controller (a virtual appliance) takes the role of controller

DDK VM

- Driver Development Kit (DDK) VM is used to compile vSwitch RPM package
- Once the RPM package is built in DDK VM, scp it to Dom0 and run rpm –i to install

Welcome Further Discussions

- Email: yaominchen@gmail.com
- Skype: yaominchen
- LinkedIn: Yao-Min Chen