Understanding XenServer Networking – The Linux Perspective

Overview

This document describes, in some detail, how Citrix XenServer networks work and how they interact with your regular networks. The first section is a reminder of the standard terminology in the workings of an Ethernet network, and the second section builds on that to explain XenServer networking.

Because XenServer networking is facilitated using standard Linux tools, there is also some discussion on the simplest way to use Linux commands to "see" what's going on.

Finally, there is a table at the end that provides a summary of XenServer "xe" networking commands and their uses.

Target Audience

This document has been written for information technology (IT) infrastructure specialists who are responsible for planning and designing a XenServer infrastructure for server virtualization. These specialists include consultants, internal IT architects, and others who are concerned with design decisions related to virtualization.

CITRIX®

Table of Contents

Quick Revision

ISO Does OSI

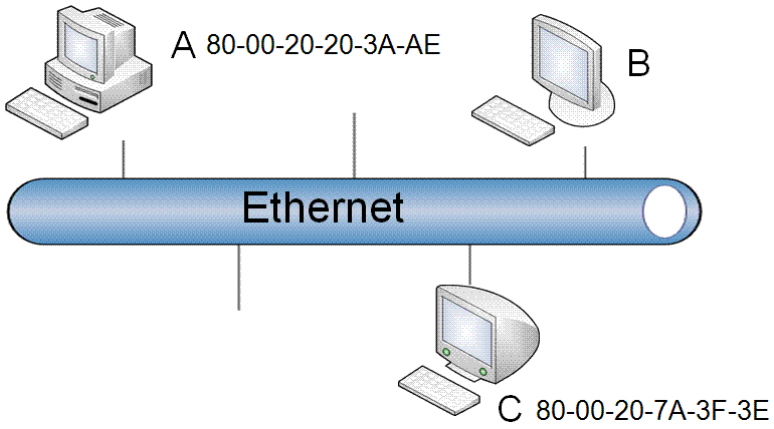
It's important to understand that XenServer networking operates at Layer 2 of the OSI.

Layer	Description
7	Application layer
6	Presentation layer
5	Session layer
4	Transport layer
3	Network layer
2	Data link layer • LLC sublayer • MAC sublayer
1	Physical layer

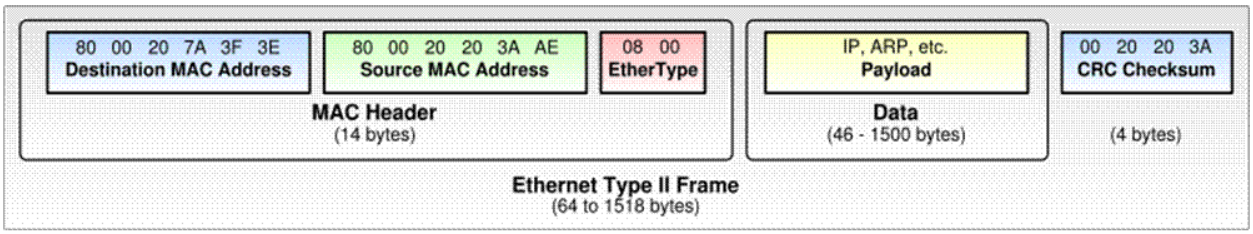
This means it's independent of any L3 addressing, such as IP. As we'll see, XenServer acts as an L2 virtual switch.

Simple Ethernet Segment

In this simple Ethernet segment, how do nodes A and C talk to each other?



Ethernet defines a frame (not a packet) which is the carrier of the data payload from the upper layers.

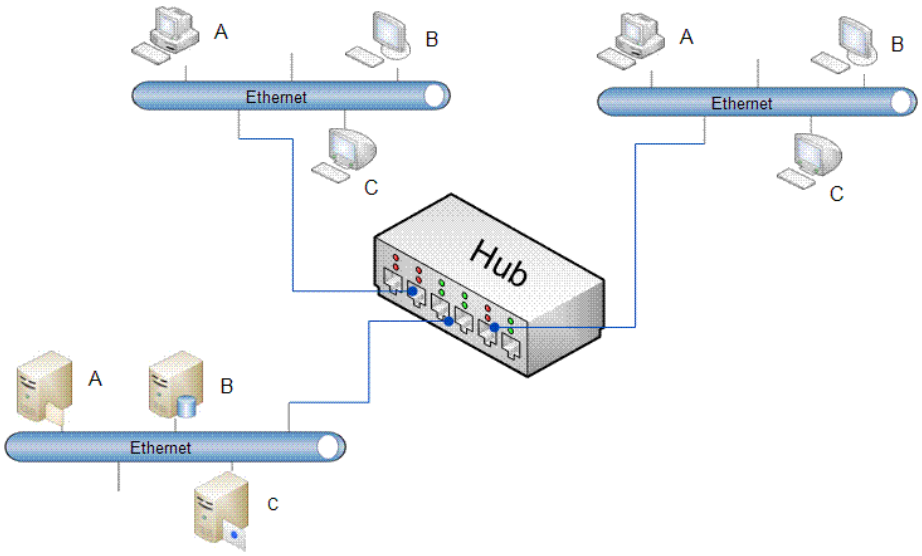


This frame is placed on the wire by layer 1 of node A, and is picked up by node C. If we looked at layer 2 of node C, we would see the same frame that node A transmitted. So:

- Each Network Interface Card or NIC for each node has a unique address, usually burnt in at the factory. This is known as the Media Access Control address, or simply the MAC address.
- When preparing a frame for transmission the destination MAC address must be known. Getting the target address to layer 2 is taken care of by layer 3, for example ARP when using IP addressing.
- Each frame has a source and destination address. The frame is seen by all nodes on the segment, but discarded by all but the NIC that has this destination address.
- In our example, node C knows who sent the frame, because of the built-in source MAC address.

Ethernet Segments

Several Ethernet segments can be joined together with a device known as a hub. A hub is also known as a repeater hub, because it repeats any frame it receives on one port out of all its other ports. This in effect turns the connected segments, into one big segment. This has scalability problems due to the way frames are transmitted.



Collisions

Nodes A & B can place a frame on the network at any time, in hopes that the circuit is not busy. If the circuit is busy, perhaps because both nodes are trying to transmit at the same time, then both nodes will back off and retry after a random period of time.

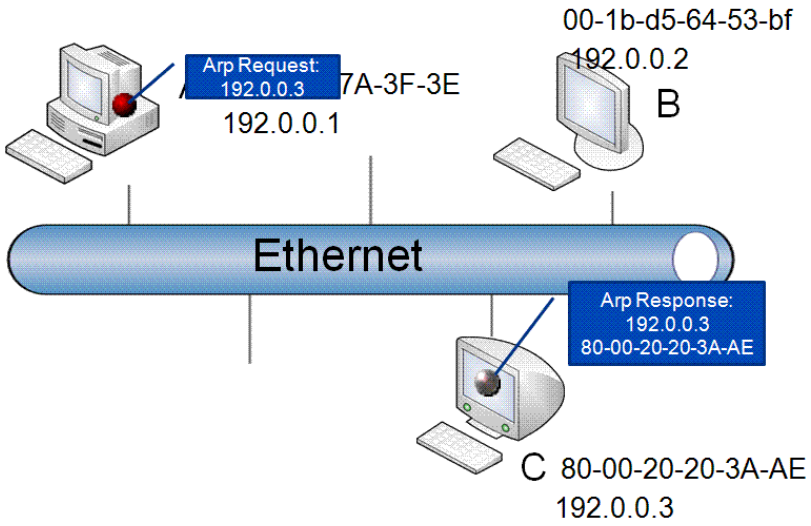
As the number of nodes increase, though, the chance of two frames from two different nodes colliding becomes a significant limiting performance factor for the segment.

Collision Domains

Similar to a hub, a switch, which is sometimes called a switched-hub, connects multiple segments. The difference is that it learns which port, on the switch, a MAC address belongs to by looking at each frame's source MAC address. When it needs to send a frame to that MAC address in the future, it doesn't repeat the frame on all ports, it sends it to the port down which the frame belongs. For the duration of transmitting the frame, this effectively creates a segment between the two ports – the source and destination – so that the two segments can talk at wire-speed. This keeps the collisions confined to each segment.

Addressing

Even though node A addresses node C by its MAC address, this still leaves the problem of how A knows the MAC address of C in the first place.



Layer 3 of the 7-layer OSI defines a network addressing scheme, most commonly the Internet Protocol (IPv4). Here both nodes A and C have been given unique IP addresses. Although not obvious here, we haven't just replaced one addressing scheme with another. IP can route packets from one node in one Ethernet segment to another node in another segment across the world.

As previously stated: an Ethernet frame has a destination address, and only the NIC with *that* address will accept the frame. There is one of two exceptions to this rule, and it's called a broadcast frame. A broadcast frame is one whose destination MAC address has all the bits set, and every NIC on the segment will process the frame.

There is a Layer 3 to Layer 2 mapping protocol called the Address Resolution Protocol (ARP) that takes advantage of broadcast frames. ARP is used to translate L3 addresses like an IP address into an L2 address like a MAC address.

In the above example network, node A has an IP address of 192.0.0.1, and node C has an address of 192.0.0.3. With TCP/IP, node A would want to communicate with node C by IP address and not MAC address, but Layer 2 still needs to address that payload to node C by MAC address.

In this case, node A at Layer 3 uses ARP to map node C's IP address to a MAC address, so that L3 can tell L2 how to address the frame. But first how does ARP know the MAC address of node C?

At first it doesn't, so ARP sends out an *ARP request* – a broadcast frame – with a payload of the IP address for node C. All nodes in the segment will process the ARP request, but only one will send an *ARP response* if they have that IP address – which in this case should only be node C. The ARP response will contain the MAC address.

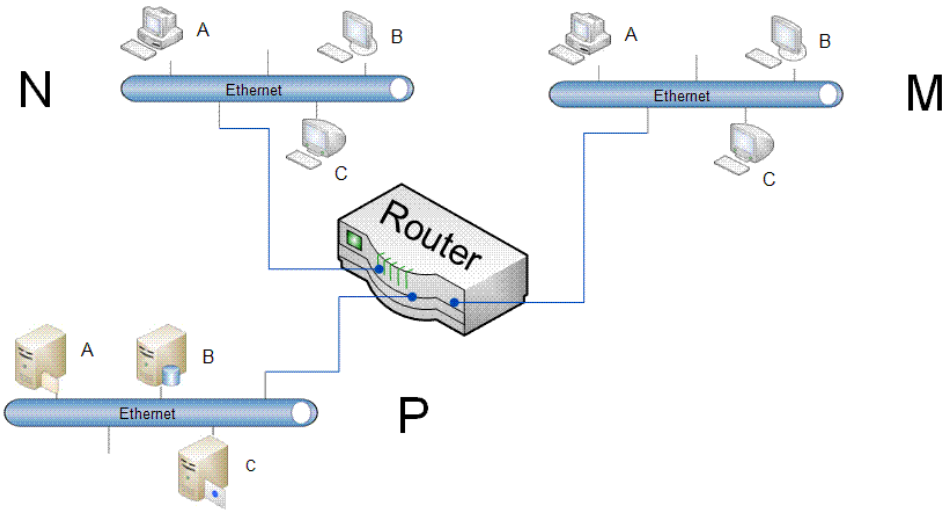
But this would be inefficient if we had to preempt every real frame A wanted to send to C with a broadcast. For this reason, each node in the segment will keep a map of IP address to MAC address translations that ARP will look at first. This is known as the "ARP cache" or "ARP tables".

In both a Windows and Unix shell, you can interrogate that machine's ARP cache with the "arp –a" command.

Routing

A router connects networks at layer 3, usually IP. The router – or gateway – also has an IP address. In fact a router is a device that can have several IP addresses, and his job is to route packets from one TCP/IP network to another. Each interface – or port – on the router will have a MAC address.

Usually, at a minimum, each node on a network has two IP addresses it knows about: its own address and the address of the router (usually called the default gateway). Users usually use TCP/IP addresses that are passed down from L7, for example when browsing web sites.



In this example, if node NA wants to send a TCP/IP packet to node PB, what needs to happen?

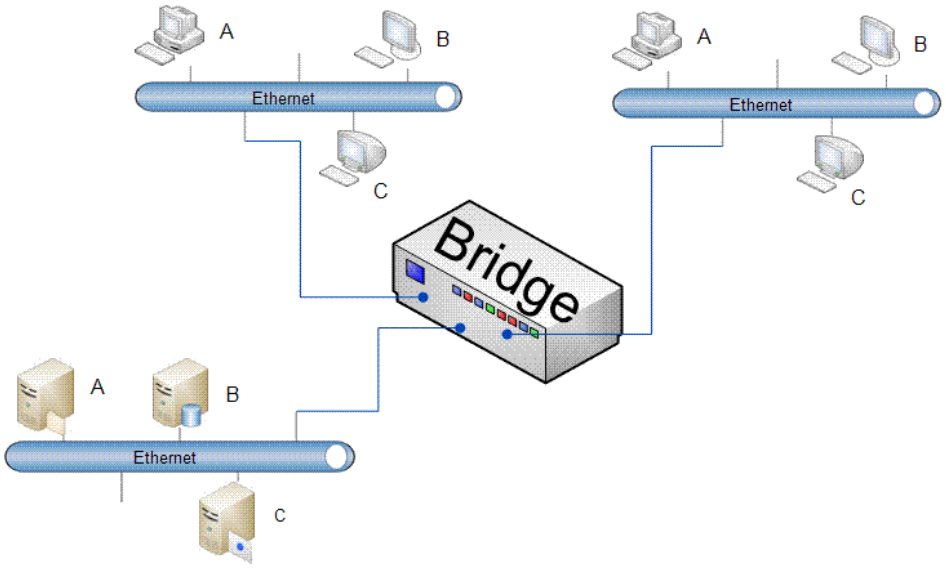
With routing, node NA will know if the destination IP address is contained on his subnet, or not. In this case it's not, so node NA at L3 knows he needs to send his TCP/IP packet through the router to get to node PB. In this case, though, the destination MAC address in the Ethernet frame will actually contain the address of the port on the router to which network N is connected. Node NA discovers the MAC address for that router port by using ARP in the usual way – the ARP request will actually contain the IP address of that router port to which network N is connected.

The router will take that IP addressed packet and place it on the appropriate port for the destination IP address. In this example the port on the router that is attached to network P will now have to use ARP in the usual way to find the MAC address of node PB.

This is a very simple example, and it can get a lot more complicated by adding switches and more routers. These basics are the same in all cases, though.

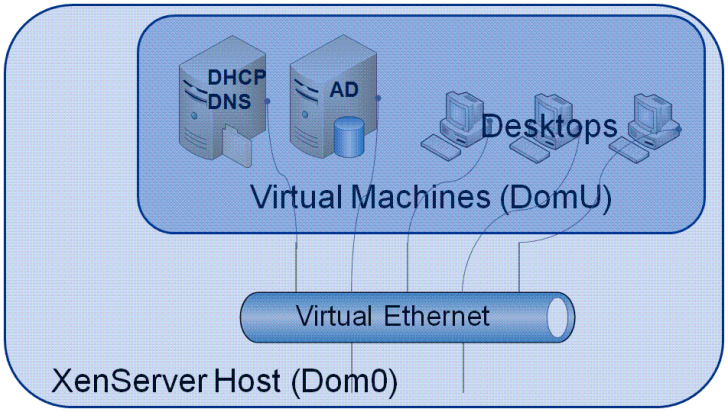
Bridging

A bridge, in the case of XenServer networking, is the same as a switch, except it's implemented in software on the XenServer host. The bridging software XenServer uses is the standard Linux implementation, with no special code from Citrix. There is therefore plenty of documentation available online.



XenServer Networking

XenServer Internal Network

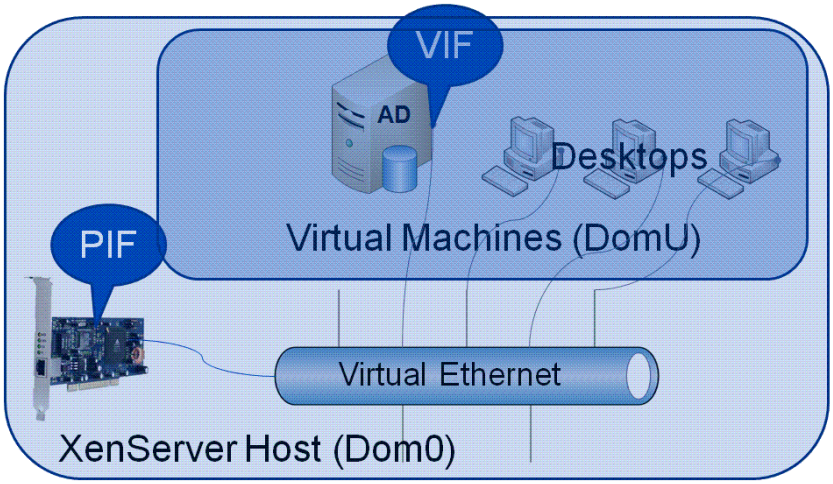


This is a simple picture of how XenServer networking works. Each virtual machine has a virtual NIC, which is connected to a virtual network, which is controlled by the XenServer host: Domain zero.

The network shown acts just like a physical Ethernet segment, and works at layer 2 of the OSI – there is no TCP/IP configuration needed here. Just like a real network, the way the network is utilized is completely up to the way the virtual machines are configured. If they need to communicate using TCP/IP then the individual OSes on those VMs need to be setup, just like with real machines on a network.

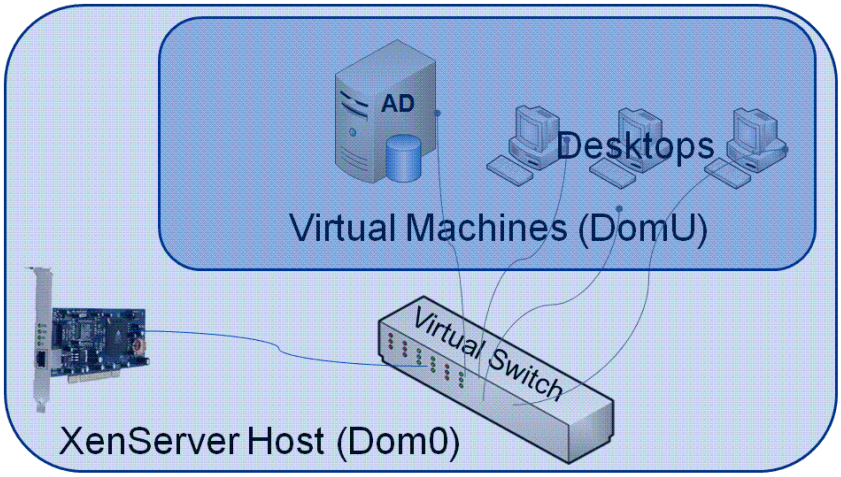
Configuration doesn't just apply to TCP/IP. The virtual machines will almost certainly need services such as DNS and DHCP. The above picture depicts, what is known in XenServer, as an *Internal Network*: It has no connection to the outside world.

The only difference between a XenServer *Internal Network*, and a XenServer *External Network* is that we can connect the virtual Ethernet to a real NIC. This means that the virtual machines can now take advantage of real services provided by other servers outside of the XenServer host.



In the Unix world, the NICs, real or virtual, are called interfaces. XenServer calls a real interface a Physical Interface or PIF and each virtual interface in a virtual machine is a VIF.

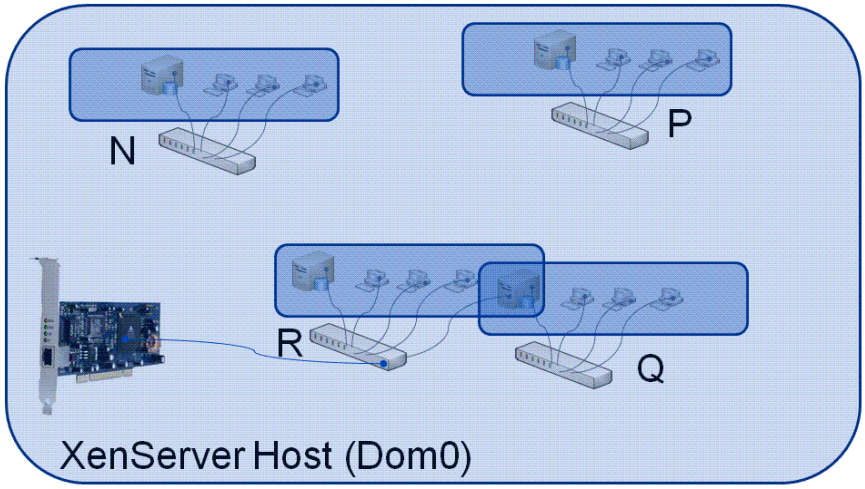
In the OS running on a VM, the VIF looks and operates like a locally installed PIF. In Windows, the device driver name may be different, depending on if the paravirtualized tools have been installed or not.



In reality, XenServer networking is accomplished by connecting the VIFs and optionally 1-PIF to a virtual switch or bridge. If you remember, a switch learns the MAC addresses of the nodes connected to each of its ports, thereby reducing the amount of traffic on its other ports. It's the same with a bridge.

In the real world a switch is used to reduce the number of collision domains on attached segments. In XenServer networking, a bridge is used to connect virtual machines together not networks. It's also used to connect those virtual machines to the outside world.

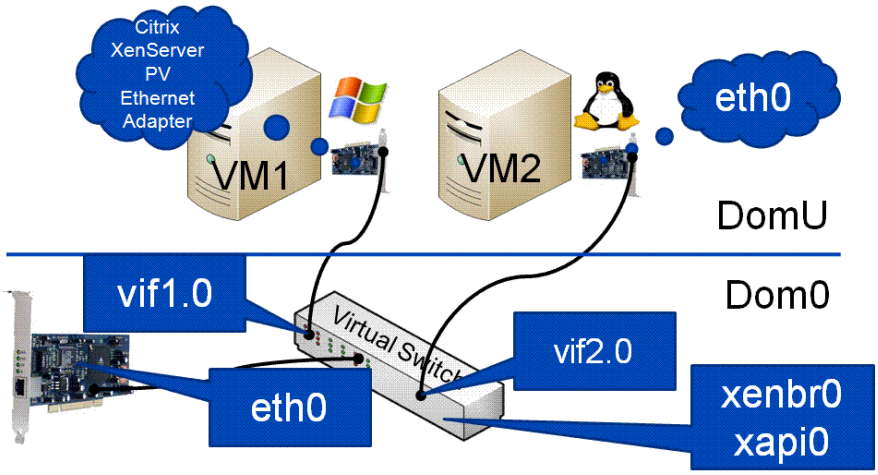
You can create several internal networks, but (in the following network) the only way to route between network Q and network R, is to have a machine that has 2 VIFs, one in each network.



Each VIF can belong to only one network, but a virtual machine can have many VIFs.

A virtual machine on internal network Q can gain access to the external network R, by routing through one of the virtual machines that has one VIF on each network. Internal networks N and P will not be able to communicate with each other, or the PIF.

So how does this transpire in reality?



Each virtual machine has a unique ID, and in the above example the Windows VM has an ID of 1, and the Linux VM has an ID of 2.

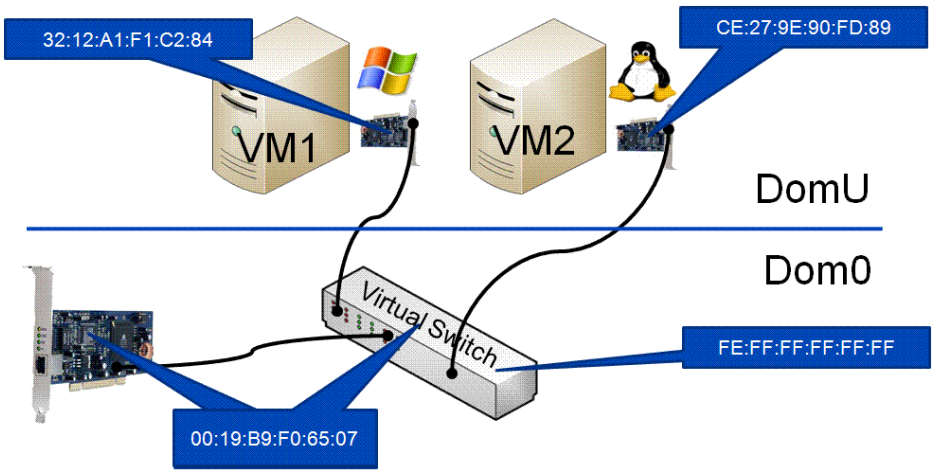
The virtual machines, assuming that the PV Tools have been installed, will see an Ethernet NIC in terms of their own environment. This mean that the Windows VM will see the “Citrix XenServer PV Ethernet Adapter” in the device manager, and the Linux VM will see “eth0”.

On the host-side, what is actually the other side of the virtual Ethernet cable joining these VMs to the bridge; the Ethernet interfaces that XenServer sees will have a naming convention. Each interface on VM1 – or guest1 – will begin with V-I-F-1-dot, followed by a sequence number that represents the interface number on that VM. So in this case, the first adapter on VM1 will have an interface name of VIF1.0 inside the XenServer host. Whatever frames are transmitted by VM1 on his virtual NIC, will be received by VIF1.0. Any frames transmitted by VIF1.0 will be received by the VM1 virtual NIC.

The real NIC on the XenServer host will be “eth0” and will be plugged into the bridge as well. Virtually speaking, this would not be from his RJ-45 interface, but from his bus interface. The RJ-45 interface is a real interface used to connect the XenServer host to the outside world.

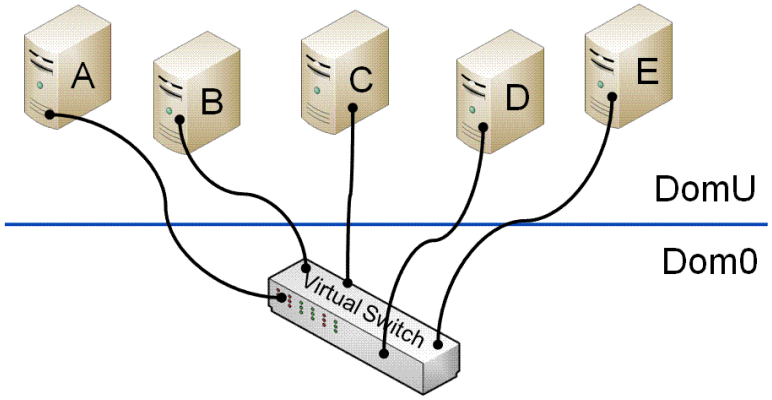
In this setup, the bridge is also seen as an interface. If it’s joining members of an external network together, its name will begin with “xenbr”, the “br” meaning bridge. If it’s joining members of an internal network together, its name will being with “xapi”.

Xen MAC Addressing



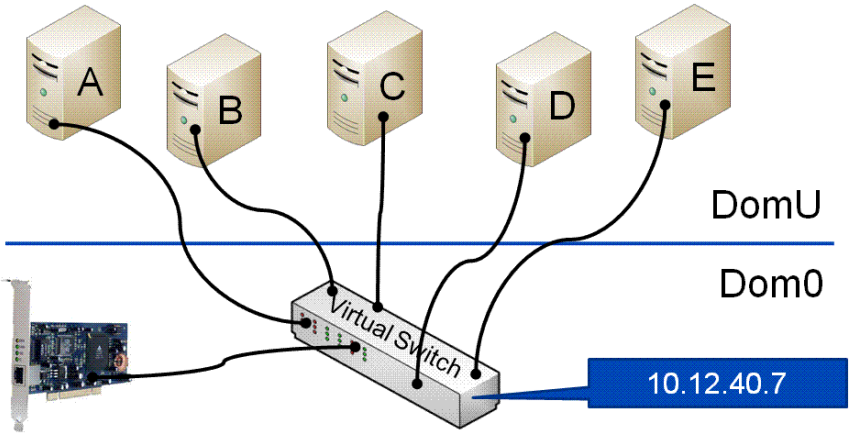
As far as MAC addressing is concerned, the virtual NICs in the VMs will be allocated random link-local MACs, or the administrator can allocate them himself. On an *Internal Network*, the bridge will be allocated a MAC address of FE:FF:FF:FF:FF:FF. But when it’s on an *External Network* it shares the MAC address of the PIF, in this case 00:19:B9:F0:65:07.

On an internal network, all the NICs on all VMs should have unique unicast MAC addresses already assigned. The bridge will have the unicast address FE:FF:FF:FF:FF:FF.



- The first frame transmitted on the network (after booting the host), from A to E, would pass through the bridge, and would be passed to all other nodes, since the switch doesn't yet know the MAC address of E. In this way it acts like a repeater.
- Node E will be the only one to pick up the frame, and nodes B thru D will ignore it.
- The switch will now learn that A is on port 1, and will associate A's MAC address with that port.
- When E responds to A, the bridge learns which port E is on, but can also send the frame directly back to A, and only A on port 1.

There will be no frames with a source or target address of the bridge.



Adding access to the physical NIC on the host turns our *Internal Network* into an *External Network*. The virtual machines communicate as previously described, but now we have a special case – the real ethernet NIC. What happens when we need to transmit a frame from node A, for example, to a node outside of the host?

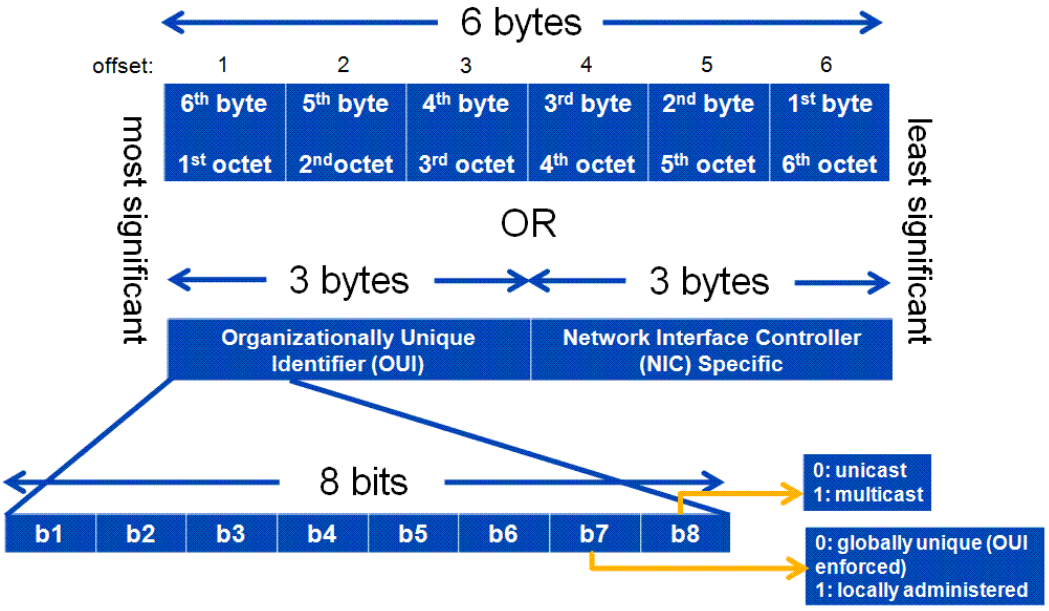
In a normal network, any frame transmitted by the NIC would have a source MAC of the NIC itself. In this case, however, the source MAC will be that of node A.

What about the other way round? What happens when the real NIC receives a frame with a destination MAC of node A? In a normal network, the NIC would drop the frame since that destination MAC address is not his. In this case, however,

part of the NICs configuration is to be put into *promiscuous mode*. When a NIC is in promiscuous mode, it will process **every** frame on that network segment.

- The virtual cables that are used to connect these interfaces, virtual or otherwise, are implemented at the interface driver level directly. In this way there is no conflict that the NIC and bridge both share the same MAC address.
- If this PIF is also being used as the management port, an IP address needs to be assigned. In this case it's actually the bridge that has the IP address, not the PIF.
- Any packet that's destined for the management 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. The PIF won't process the IP packet, since it doesn't have an IP address assigned. It's really a quirk of Linux that the IP address needs to be assigned to the bridge.

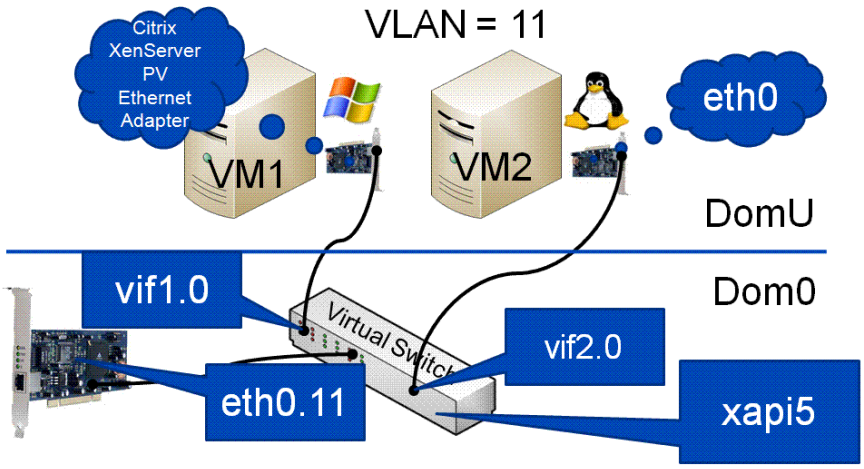
Assigning MAC Addresses



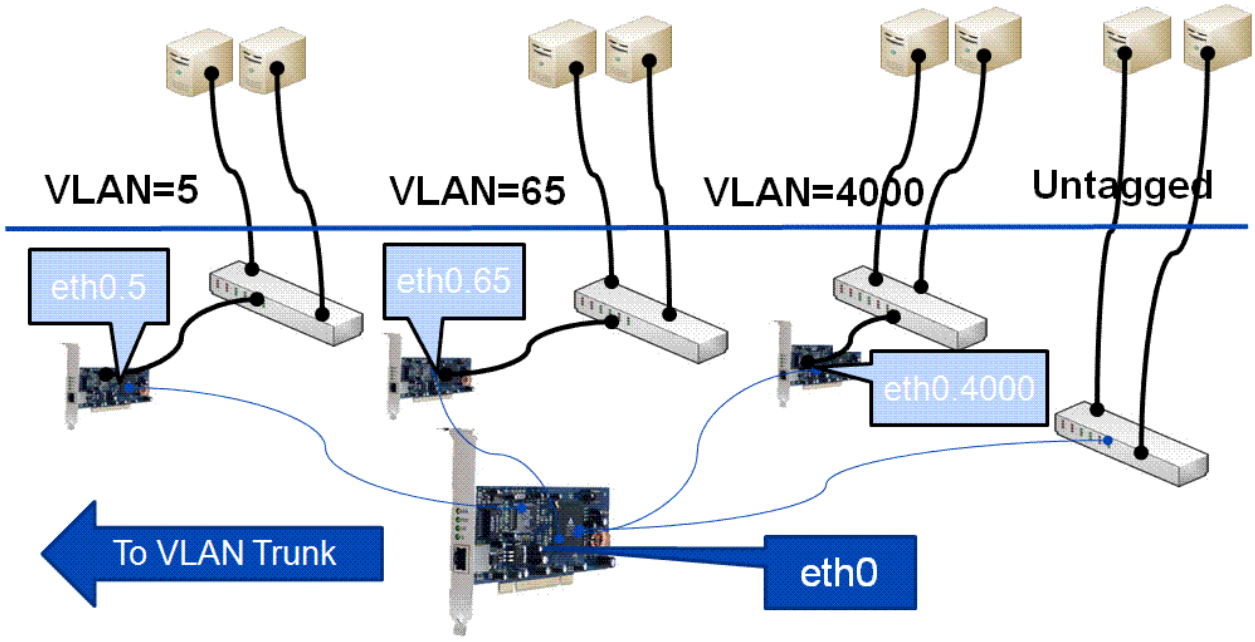
Two important bits really matter when assigning a MAC address. They are the first and second least significant bits of the second leftmost byte in a MAC address, bit B7 and B8, shown.

- When b7 is set to 0, this MAC address is a manufacturer-assigned MAC address.
- If b8 is set to 1, it is a multicast MAC address. For example, FF:FF:FF:FF:FF:FF is a broadcast MAC address that will be received by all the machines in the Ethernet while FE:FF:FF:FF:FF:FF is a local unicast MAC address, which is used in the Xen network. So a user-specified MAC address should at least be a unicast MAC address, and probably locally administered. Basically the 2nd hex digit should be one of: 2, 6, A or E.

XenServer Networking: VLANs



In the case of creating a VLAN, every distinct VLAN will get its own bridge. Also, the (pseudo) PIF will have a dot separated name to include the vlan tag number and, when on the real network, the bridge name will start with “xapi”. Apart from that, everything else will be the same as normal external network. It’s not possible to create an internal VLAN network.



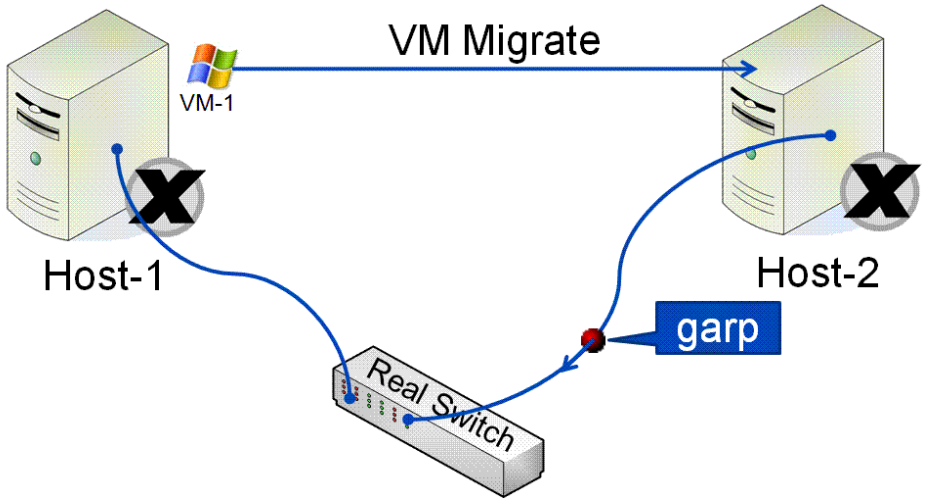
In this example there are 8 virtual machines as guests on a single XenServer host. Also on this host are 3 VLANs, with tags 5, 65 and 4000. There are also a couple of machines that have a regular non-vlan (untagged) external network, so their bridge is wired directly to the PIF.

- The interfaces (eth0.5, eth0.65, eth0.400, etc) for the VLANs are actually created in Linux as virtual interfaces. These interfaces simply tag each frame with the appropriate VLAN number before they are moved on to the PIF.
- All frames emanating from a guest VLAN will leave the PIF, and hence will be seen by the outside world, as tagged. This means when having multiple VLAN guest networks, a frame may be tagged differently depending on the source VLAN. For this reason, you are required to (eventually) connect the PIF of the XenServer to a VLAN trunk port that supports 802.1Q encapsulation, when VLANs are in-play.

When using VLANs the XenServer Host handles all interpretation of the VLAN tags. Any frames sent to guests that are part of a VLAN will, even so, remain untagged.

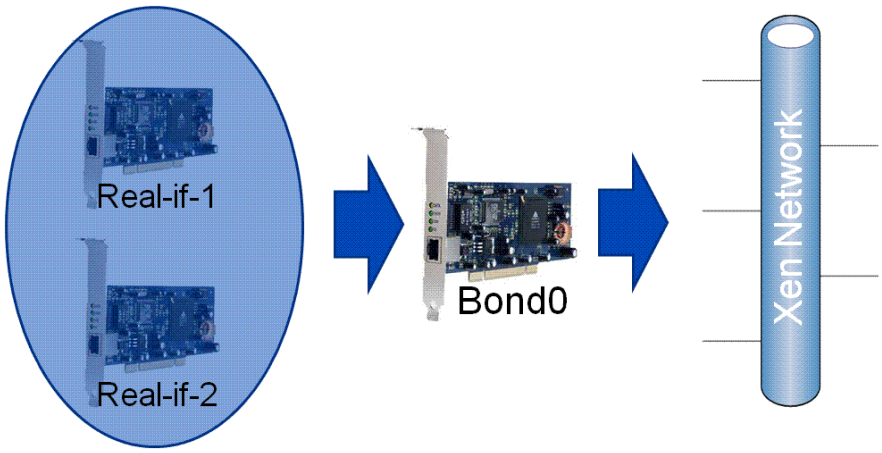
XenMotion

During a migration of a VM from one host to another, any memory pages that change on the source host are copied to the destination host. This process is repeated until the number of pages to copy is minimal, and the VM can be started on the destination host.



But in this example setup, the external real switch device is expecting the MAC address of the VM to be on one port, while it's actually just migrated to another port. In this case, the last step of the migration is for the destination host to update any external devices with a gratuitous ARP packet. A "garp", is nothing more than a regular ARP request, but with the MAC and IP address already filled-in. This serves as an update to any external devices' arp cache, and there would be no response.

NIC “Teaming” or “Bonding”



NIC bonds can improve XenServer Host resiliency by using two physical NICs as if they were one. If one NIC within the bond fails the host's network traffic will automatically be routed over the second NIC. In XenServer 4.1, NIC bonds work in an active/passive mode, with only one physical NIC ever in use.

This virtual interface (in the above case Bond0) will then look like a regular PIF to any XenServer External Network.

- XenServer NIC bonds completely subsume the underlying physical devices (PIFs). In order to activate a bond the underlying PIFs must not be in use, either as the management interface for the host or by running VMs with VIFs attached to the networks associated with the PIFs.
- XenServer NIC bonds are represented by additional PIFs. The bond PIF can then be connected to a XenServer network to allow VM traffic and host management functions to occur over the bonded NIC.
- The bond interface itself can be made to have its own MAC address; otherwise it inherits the MAC address of the first listed PIF when created.

Example Admin Commands

Bridge Control: brctl

```
[root@xenserver-owithoff00 ~]# brctl show
bridge name     bridge id               STP enabled     interfaces
xapi1           8000.000000000000       no
xapi5           8000.0019b9f06509       no              vif14.0
                                                        eth1.11
xenbr0          8000.0019b9f06507       no              eth0
xenbr1          8000.0019b9f06509       no              vif14.2
                                                        eth1
```

brctl is used to set up, maintain, and inspect the Ethernet bridge configuration in the Linux kernel. Bridging and the associated brctl command can be found on most later, non-XenServer Linux derivatives.

- The *show* option will show all the bridges being supported, and their properties.
- This example shows “xapi1” which is a bridge for an Internal Network. Looking at the interfaces column, you can see that there are no interfaces attached. This tells us that there are no virtual machines alive on this network.
- xapi5 is VLAN bridge because we can see there is one vif called vif14.0, and a virtual Ethernet adapter that deals with VLAN 11, eth1.11.
- xenbr0 is a real, untagged, external network, which lets guests get to the outside world through the PIF eth0. In this case there are no guests booted on this network.
- xenbr1 is a real, untagged, external network, which links the virtual adapter vif14.2 to the 2nd PIF on the host. We know by the naming that both vif14.0 and vif14.2 are virtual adapters on the same guest. Perhaps some TCP/IP routing is being done through this guest.

arp and ifconfig

```
[root@xenserver-owithoff00 ~]# arp -a
? (10.12.41.7) at 32:12:A1:F1:C2:84 [ether] on xenbr0
? (10.12.41.1) at 00:D0:68:0C:A6:70 [ether] on xenbr0
[root@xenserver-owithoff00 ~]#
```

The arp command manipulates a particular system’s ARP cache. For example in this case, 10.12.41.7 belongs to a virtual machine whose MAC addr is 32:12:A1:F1:C2:84. From what we know about the MAC address format, we can see that this MAC address is a locally administered unicast address.

```
[root@xenserver-owithoff00 ~]# ifconfig xenbr0
xenbr0    Link encap:Ethernet  HWaddr 00:19:B9:F0:65:07
          inet addr:10.12.41.40  Bcast:10.12.41.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:18607986 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12072934 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2039439342 (1.8 GiB)  TX bytes:1692101941 (1.5 GiB)
```

Using the *ifconfig* Linux command, with the bridge for the 1st external network, we can see that the bridge itself has an IP address of 10.12.41.40. This means that this also serves as the administrative interface for XenAPI. For example, the XenCenter application would connect through this IP address.

xe examples

XenServer "xe"	Comment
xe network-list xe pif-list xe vif-list	• List the networks and attributes • List interface attributes. Also good for UUID values
xe pif-forget uuid= xe pif-introduce host-uuid= mac= device=	• Forgetting an interface means the Xen host no longer has control. Configure through Linux. • Introducing an interface means all configuration should go through Xen.
xe pif-reconfigure-ip ip=xx.xx.xx.xx uuid=	Change the IP attributes of management interface.
xe pif-* uuid= xe vif-* uuid=	There are several pif- and vif- commands for network setup.
xe pif-param-set other-config:ethtool-duplex="full" uuid=	Usually used to set adapter specific attributes.
xe host-set-hostname-live host-name=fqdn.com host-uuid=	Change the FQDN of the host.
xe vm-list params=name-label,dom-id	Shows VMs and their dom-id to associate with VIF names

Notice

The information in this publication is subject to change without notice.

THIS PUBLICATION IS PROVIDED “AS IS” WITHOUT WARRANTIES OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT. CITRIX SYSTEMS, INC. (“CITRIX”), SHALL NOT BE LIABLE FOR TECHNICAL OR EDITORIAL ERRORS OR OMISSIONS CONTAINED HEREIN, NOR FOR DIRECT, INCIDENTAL, CONSEQUENTIAL OR ANY OTHER DAMAGES RESULTING FROM THE FURNISHING, PERFORMANCE, OR USE OF THIS PUBLICATION, EVEN IF CITRIX HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES IN ADVANCE.

This publication contains information protected by copyright. Except for internal distribution, no part of this publication may be photocopied or reproduced in any form without prior written consent from Citrix.

The exclusive warranty for Citrix products, if any, is stated in the product documentation accompanying such products. Citrix does not warrant products other than its own.

Product names mentioned herein may be trademarks and/or registered trademarks of their respective companies.

Copyright © 2008 Citrix Systems, Inc., 851 West Cypress Creek Road, Ft. Lauderdale, Florida 33309-2009 U.S.A. All rights reserved.

Version History			
Author	Version	Change Log	Date
Olivier Withoff Principal Technical Readiness Engineer Worldwide Field Readiness	1.0	Initial Document	July 17th, 2008

851 West Cypress Creek Road Fort Lauderdale, FL 33309 954-267-3000 http://www.citrix.com

Copyright © 2008 Citrix Systems, Inc. All rights reserved. Citrix, the Citrix logo, Citrix ICA, Citrix MetaFrame, and other Citrix product names are trademarks of Citrix Systems, Inc. All other product names, company names, marks, logos, and symbols are trademarks of their respective owners.