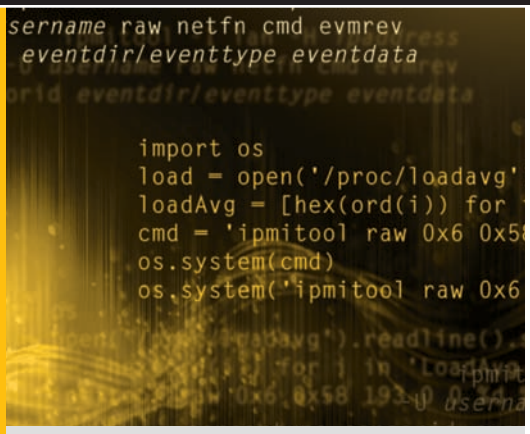

BEST PRACTICES FOR NETWORKING IN A VMWARE ENVIRONMENT

By Alberto Ramos

Dell has used VMware® virtualization extensively to help optimize its hardware utilization, streamline provisioning processes, and increase resilience. By following the networking best practices used by Dell engineers in their own IT environment, organizations can create an efficient, resilient virtualized infrastructure.

Related Categories:

Dell PowerEdge servers

Networking

Virtualization

VMware

Visit DELL.COM/PowerSolutions for the complete category index.

Virtualization has become a key technology in enterprise IT environments, helping increase flexibility and allowing organizations to consolidate their servers to help increase hardware utilization. When working with virtualization, the key factor to keep in mind is that each physical server is no longer one system but many—a single server might run 25 or more virtual machines (VMs). This new paradigm requires administrators to give attention to the resilience of the technical design, regardless of the importance of the VMs being hosted. In terms of networking, this approach provides the VMs with resilient network connections that can tolerate the failure of network interface cards (NICs), cables, and switches.

This article describes best practices developed and used by Dell when implementing VMware virtualization in its own IT environment, and considers some aspects of networking design that organizations should take into account when implementing this type of virtualized infrastructure—including the mapping between physical NICs and virtual NICs (vNICs), a scripted solution for virtual switch creation, the importance of virtual LANs (VLANs), scripts for trunk creation, and an approach for testing the network infrastructure once it is in place.

IDENTIFYING NIC PORTS

The first step toward creating a fault-tolerant design is identifying where the different NIC ports are from the point of view of the OS. At boot time, the VMware ESX software scans the PCI slots for NIC ports and assigns them the names vmnic0, vmnic1, and so on based on the order of discovery. The order of discovery depends on the server model and the location of the NICs in the PCI slots.

To identify the NIC ports, administrators can plug a single network cable into a server and then run the `esxcfg-nics -l` command at the console to see which port is active. (This command is part of the VMware esxcfg tool set, which plays a key role in the best practices described in this article and also includes commands for configuration aspects other than networking.)

Figure 1 shows the output of this command for a Dell™ PowerEdge™ 2950 server with a dual-port NIC in the third PCI slot. After identifying the first NIC port, administrators can then move the cable to a different port and repeat the process until all ports are identified. In the case of this PowerEdge 2950 server, they might decide to use two NIC ports for VM traffic. To help enhance the system's resilience, they would choose one port from the PCI NIC and another port in the on-board NIC—for example, vmnic1 and vmnic2.

UNDERSTANDING PORT GROUPS

To facilitate configuring network settings in VMware ESX, VMware has introduced the concept of the *port*

```
Name      PCI              Driver    Link        Speed   Duplex Description
vmnic0    05:00.00 bnx2    Up        1000Mbps    Full    Broadcom Corporation BCM5708
vmnic1    09:00.00 bnx2    Down      1000Mbps    Full    Broadcom Corporation BCM5708
vmnic2    0a:00.00 e1000   Down      1000Mbps    Full    Intel Corporation 82571EB
vmnic3    0a:00.01 e1000   Down      1000Mbps    Full    Intel Corporation 82571EB
```

Figure 1. *Output of the esxcfg-nics -l command on a Dell PowerEdge 2950 server with a dual-port NIC in the third PCI slot*

group, defined as a set of ports that share a common configuration such as a VLAN, traffic shaping, or security setup. Unlike with physical servers, where administrators choose which port to patch, ESX allows administrators to define a collection of network settings as a port group, then add VMs to that group. Because most environments have multiple network ports that require the same configuration, this approach helps simplify management by allowing administrators to define the configuration for many ports at once.

Administrators can apply similar configurations at the virtual switch level to act as default settings for virtual switches, although port group settings override virtual switch settings. Virtual switches can contain many port groups, but each vNIC in a VM belongs only to a single port group. Because a given port group can be used with only one virtual switch, port group names must be unique within each host.

CONFIGURING EXTERNAL CONNECTIVITY

Communication between VMs attached to the same virtual switch is an internal process handled by the VMware ESX host. However, most solutions also typically require some degree of external connectivity, which can be enabled by attaching one or more vNICs to the virtual switch. Administrators should keep in mind that physical NICs can be part of only one virtual switch.

The physical NICs act as the uplink to the physical network. ESX allows administrators to add up to eight physical NICs to a virtual switch to provide fault tolerance and load balancing, two features that are also configured through port group and virtual switch settings. Administrators can select from three different load-balancing modes:

- **Originating port ID:** In this mode, VMs take on a NIC based on the order they are booted, in a round-robin fashion. This mode is typically the simplest and requires less overhead than the other two, and produces good results in most cases.
- **Source Media Access Control (MAC) address hash:** This method helps ensure that a VM always uses the same NIC for external connectivity. However, depending on the result of the hash calculation, it might cause one NIC to handle considerably more connections than the others in the team.
- **IP address hash:** This mode is the most powerful of the three, taking into account both source and destination IP addresses. By doing so, each TCP connection requested by a VM can potentially use a different NIC. This mode can produce a good load distribution, but requires increased overhead and additional configuration on the external switch to implement the IEEE 802.3ad protocol.

Administrators should keep in mind that these modes only balance outbound traffic.

SCRIPTING VIRTUAL SWITCH CREATION

The VMware graphical user interface for configuring virtual switches is designed to be intuitive. However, setting up a farm with many nodes and many network switches using this interface can be time-consuming. In addition, because port group names are case sensitive and certain features such as VMware VMotion™ technology rely on the spelling of these names being identical across all systems, scripted installation of a virtual network infrastructure can have significant advantages. Some of the tools in the esxcfg tool set enable administrators to perform this type of installation.

Constructing an example script can help demonstrate this process. Figure 2 shows a scriptable virtual switch configuration that follows best practices recommended by VMware. For example, keeping the console separate from the VM networks and using a dedicated network for VMotion help increase security. Because this example network does not require external connectivity, a private IP address range can be allocated. In addition, this configuration uses two physical NIC ports in separate PCI slots (vmnic1 and vmnic2) to provide fault tolerance.

The example script assumes that administrators have installed an ESX host with a VMware Infrastructure 3.0.*x* installation CD and have chosen to create a default network for virtual switches, in

Physical port	Function	Virtual switch	Port group
vmnic0	ESX console	vSwitch0	Service Console
vmnic1	VM port	vSwitch1	VMnet
vmnic2	VM port	vSwitch1	VMnet
vmnic3	VMotion	vSwitch2	VMotion

Figure 2. *Virtual switch configuration created by the example script*

which case vSwitch0 exists and contains the Service Console and VM Network port groups. (A kickstart installation produces the same result.) The script also assumes that administrators have specified the desired IP address for the service console connection.

The first lines of the script focus on the console configuration. Administrators can separate the console and VM network traffic by removing the default VM functionality from vSwitch0 (this functionality is later implemented in vSwitch1):

```
# Remove the default port group for VM traffic
esxcfg-vswitch -D "VM Network" vSwitch0
```

Administrators should keep in mind that because they use the ESX console when connecting remotely to the server, they should avoid changing the console's IP address as part of the script to help ensure they do not lose connectivity. If they do change the example script to modify the console's IP address, they should run it locally at the system screen.

The following lines create vSwitch1 with two physical NICs to help provide resilience for the VMs:

```
# Create a switch for VMs
esxcfg-vswitch -a vSwitch1

# Add two NICs to the virtual switch
esxcfg-vswitch --link=vmnic1 vSwitch1
esxcfg-vswitch --link=vmnic2 vSwitch1

# Create a port group for VM use
esxcfg-vswitch --add-pg=VMnet vSwitch1
```

To help increase security and performance, best practices recommend keeping all VMotion connections together in a private Gigabit Ethernet switch. Because VMotion uses TCP/IP to relocate VMs from host to host, administrators need a unique IP address for each server for this purpose. The VMotion NICs will connect to a private switch, so the example script can use the 192.168.*x.x* range:

```
# Create the VMotion switch
esxcfg-vswitch -a vSwitch2

# Add the last NIC to the VMotion switch
esxcfg-vswitch --link=vmnic3 vSwitch2

# Create a port group for VMotion
esxcfg-vswitch --add-pg=VMotion vSwitch2

# Add the VMkernel interface to the VMotion port
   group
esxcfg-vmknic -i 192.168.1.'hostname -s | awk
   -F- '{ print $2 }'' -n 255.255.0.0 -a VMotion
```

The last command in the preceding script assumes that the server name contains a single hyphen followed by the server number. It then uses that number to create a unique IP address for VMotion. For example, an environment with 20 physical servers would have the server names ESX-1, ESX-2, and so on through ESX-20, with the script generating the IP addresses 192.168.1.1, 192.168.1.2, and so on through 192.168.1.20. Administrators can manipulate the `awk` statement to look for a different separator between the server name and server number.

When using a recent version of ESX, administrators may experience a problem in which one command does not finish execution before the next command starts. If so, they can introduce a delay between each command using the `sleep` command followed by the number of seconds to wait: `sleep 10`, for example, adds a delay of 10 seconds.

The final step is to assemble the pieces of the script in a text file, make it executable, and deploy it to the servers. Administrators can make a script executable using the command `chmod 755 scriptname`. A good approach to deploying the script is to have it copied at installation time. For example, if administrators are performing a kickstart installation from a Network File System (NFS) share, they can add lines such as the following at the `%post` section of the kickstart file to copy the script to the /root folder of the ESX console:

```
%post
mkdir /mnt/post
mount -o nolock  NFSserver:/exported/directory/
   /mnt/post
cp -f  /mnt/post/vnet-config.sh  /root/
umount -v /mnt/post
rmdir -v /mnt/post
```

After the installation, administrators can log on to the ESX server and run the script. This method helps ensure consistency in the virtual network configuration across the farm, and in particular helps ensure consistent naming, which is critical for the proper functioning of the VMotion and Distributed Resource Scheduler (DRS) features. Administrators can also use Altiris® software or a similar tool if available.

Figure 3 shows the resulting network layout in VMware VirtualCenter after the script has run, which corresponds to the configuration specified in Figure 2.

SUPPORTING VIRTUAL LANs

VLANs provide a way to segment a physical network to help increase security or performance. This segmentation is achieved by creating a tag in every network packet that identifies which VLAN that packet belongs to; systems connected to a given VLAN can receive only packets on that VLAN. A port that belongs to more than one VLAN is called a trunk.

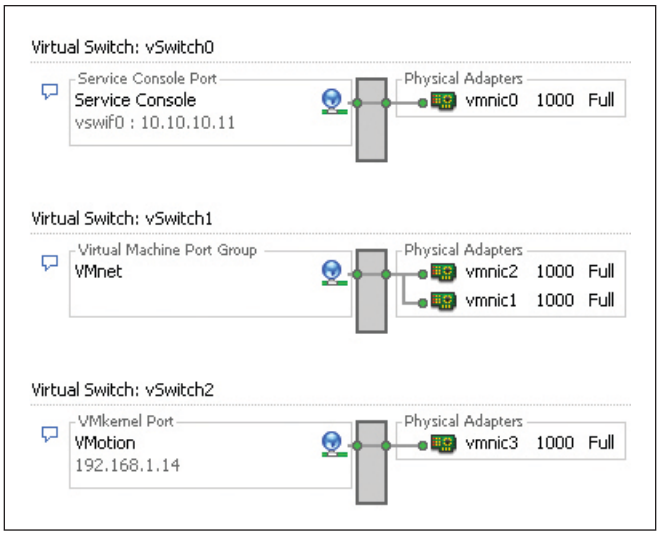

Figure 3. *Network layout in VMware VirtualCenter after running the example script*

VMware ESX enables administrators to configure VLANs at each of the three levels that make up a virtualized network configuration: the VM with its vNICs, the virtual switch that connects upstream through physical NICs, and the physical switch, a part of a traditional network infrastructure.

Virtual machine VLANs

Administrators sometimes must connect a VM directly to multiple networks, which they can do in one of two ways. The first method is to configure multiple vNICs for the VM. However, ESX limits each VM to four vNICs. In most cases this number is sufficient, but if administrators require more than four vNICs, they can use a second method: installing a VLAN-capable driver for the vNIC so that the VM can communicate with multiple VLANs, turning the vNIC into a trunk. Administrators then must instruct the virtual switch to preserve all VLAN tagging performed by the VM and pass the tagged packets to the physical switch, which they can do by configuring VLAN 4095 on the port group the VM is a member of (see Figure 4). In addition, they must configure the port in the physical switch as a trunk as described in the "Physical switch VLANs" section in this article. Administrators should keep in mind that installing VLAN- and trunk-capable drivers inside a VM is not always possible, and may not be the most advisable solution in their environment.

Virtual switch VLANs

Using virtual switch VLANs is the preferred method when sharing physical cables is not a problem and the environment requires connectivity to a large number of networks, making it unfeasible to assign specific NICs to each. Administrators must create one port group per VLAN, then attach each VM to the corresponding port group (see Figure 5).

Configurations such as the one shown in Figure 5 transmit multiple VLANs through the same physical NIC, which makes it a trunk. The tagging and trunk functionality is handled transparently by ESX for the physical NIC only. However, administrators must explicitly configure the port on the physical switch as a trunk. (For more information about setting up trunks on Dell PowerConnect™ and Cisco switches, see the "Configuring trunks on physical switches" section in this article.)

If administrators want to modify the example script from the "Scripting virtual switch creation" section in this article to enable the VMnet port group to tag the packets for VLAN 101, they can do so by adding the following lines:

```
# Enable VMs to use VLAN 101
esxcfg-vswitch --add-pg=VMnet vSwitch1
esxcfg-vswitch -p VMnet -v 101 vSwitch1
```

If they require multiple VLANs, they can repeat these lines to create multiple port groups, each with a different VLAN.

Physical switch VLANs

Using physical switch VLANs is convenient when administrators must connect a farm to several networks that need to be kept separate. For example, an organization might have a production network where tight security policies are in place and a test network where security is more relaxed. The natural solution is to set up two different farms. The problem is that, depending on demand, the organization could end up saturating one farm and underutilizing the other. Setting up a large consolidated

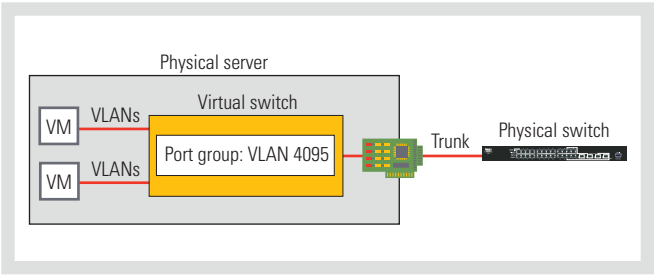

Figure 4. *Example configuration for a virtual machine VLAN in a VMware ESX environment*

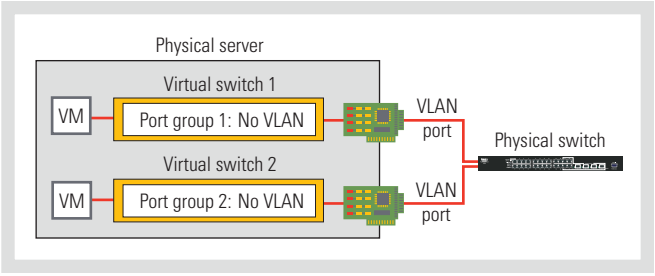

Figure 5. *Example configuration for a virtual switch VLAN in a VMware ESX environment*

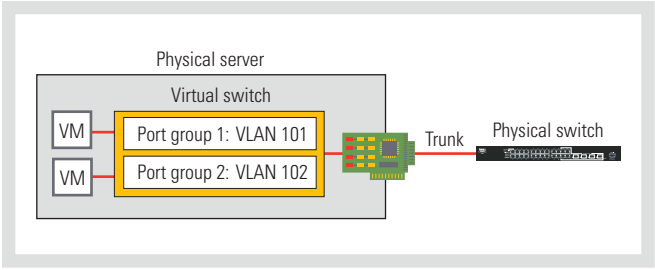

Figure 6. *Example configuration for a physical switch VLAN in a VMware ESX environment*

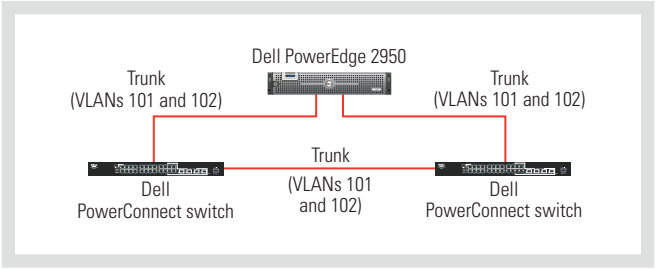

Figure 7. *Implementation of a fully redundant network in a VMware ESX environment*

farm with two separate virtual and physical network infrastructures can help maximize the investment in the hardware for virtualization.

One drawback of this approach is that it requires at least as many NICs as VLANs, and in practice typically requires significantly more NICs than VLANs to support console connections, VMotion, Internet SCSI (iSCSI) storage, and perhaps multiple NICs for certain VLANs to help increase resilience. From a configuration point of view, this approach does not require any special configuration on the ESX host because the work is performed by the physical switch (see Figure 6). Alternatively, administrators can specify VLAN ID 0 in the port group, which in ESX has the special meaning of "not tagged."

CONFIGURING TRUNKS ON PHYSICAL SWITCHES

Example commands can help demonstrate the process of configuring trunks on physical switches. The commands in this section assume an environment consisting of a Dell PowerEdge 2950 server configured with two VLANs—VLAN 101 and VLAN 102—in vSwitch1 by means of two port groups, with both VLANs already in the network and available through vmnic1 and vmnic2. In this case, the goal is to configure two switch ports as trunks. These example commands show how to convert one of these switch ports as a trunk for Dell PowerConnect and Cisco switches; administrators can configure the second port by repeating the commands for that port.

In this example environment, administrators can configure the Dell PowerConnect switch trunk using the following commands:

```
interface ethernet g7
switchport mode trunk
switchport trunk allowed vlan add 101
switchport trunk allowed vlan add 102
exit
```

They should use VLAN 1 as the native VLAN of the trunk, which they can configure using the following command:

```
switchport trunk native vlan 1
```

Next, they can configure the Cisco CatOS trunk as follows:

```
CatOS-switch> (enable) set vlan 1 2/7
CatOS-switch> (enable) set trunk 2/7 on dot1q
CatOS-switch> (enable) clear trunk 2/7 1-1005
CatOS-switch> (enable) set trunk 2/7 on dot1q
   101,102
```

The first line of this group of commands sets the native VLAN of the trunk to VLAN 1, while the third line removes VLANs that are added to the trunk by default. Finally, they can configure the Cisco IOS trunk as follows:

```
IOS-sw (enable)# configure terminal
IOS-sw (config)# interface gigabitethernet2/7
IOS-sw (config-if)# switchport mode trunk
IOS-sw (config-if)# switchport trunk allowed
   vlan remove 1-4094
IOS-sw (config-if)# switchport trunk allowed
   vlan add 101,102
IOS-sw (config-if)# switchport trunk native vlan 1
```

One feature of VMware ESX servers that may surprise network administrators is the ability to provide multiple links between physical and virtual networks without causing spanning tree loops. The configuration created by the preceding commands, for example, has created two uplinks from vSwitch1 to the physical switch. In a non-virtualized environment, this would cause a Layer 2 loop, requiring a protocol like Spanning Tree Protocol (STP) to handle the configuration. However, because ESX does not create loops, administrators have an additional degree of flexibility.

For example, rather than connecting both links to one physical switch, administrators can spread the uplinks across multiple physical switches. The only requirement is that the physical ports be in the same VLAN (or in the same broadcast domain, to be precise). Figure 7 shows a practical implementation for a Dell PowerEdge 2950 server with two NIC ports and a trunk with VLANs 101 and 102. The link between both physical switches is also a trunk that carries those two VLANs.

```
C:\>PCATTCP.exe -r
PCAUSA Test TCP Utility V2.01.01.08
TCP Receive Test
  Local Host  : Srv-15
**************
  Listening...: On port 5001
```

Figure 8. *Command and output to start the TTCP utility in a virtual machine as a receiver*

TESTING NETWORK RESILIENCE

The design and provisioning of a solution often receives the majority of energy and focus in enterprise environments. However, systematic and comprehensive testing procedures can be critical to helping ensure that the final result behaves as expected. These testing procedures are even more important for virtualization solutions than for traditional physical servers because the high consolidation ratios are likely to magnify even very small problems.

From a networking perspective, failover is a key test. A Dell PowerEdge 2950 server with two NICs for VM connectivity, for example, can boot two VMs in the same VMware ESX server. The default load-balancing mechanism makes each VM use a different NIC. Administrators can test this mechanism by setting up a continuous ping to each VM from an external system, then disconnecting one cable at a time to verify failover and failback. Typically, recovery may take up to 30 seconds.

To help administrators make failover decisions, ESX supports monitoring link status, a method that works well in most scenarios. One exception, however, is environments configured as in Figure 7, where ESX would not detect a failure in the link between the two switches. For this type of configuration, administrators could instead use the "beaconing probe" detection method, which uses probe packets at regular intervals. They should use this feature carefully, however; because the probes are broadcast packets, an excessive number of them can degrade network performance.

TESTING NETWORK THROUGHPUT

Typically, the best approach to testing network throughput is to do so from the VMs themselves, which helps provide an accurate gauge of actual performance. Administrators can perform these tests by setting up two test VMs with a network testing tool such as Test TCP (TTCP) for the Microsoft® Windows® OS or NetPIPE (NPtcp) for the Linux® OS.[1]

The first step is to place each VM on a different host, by relocating them using VMotion if necessary. Second, because the environment might have multiple NICs allocated to the VMs, administrators must identify which one they are testing, which they can do by unplugging all the NICs in the team except the one they want to test. They can then perform the test and repeat the procedure for all the NICs in the team.

Figure 8 shows the command to start the TTCP utility in one of the VMs as a receiver as well as the output that the transmitter is ready to start the test. Administrators can then test the performance by specifying the IP address of the receiver in the second VM as well as a sufficiently large number of packets, such as 100,000, using the command shown in Figure 9. The result in Figure 9 translates to 243.6 Mbps. If the result is below 100 Mbps, administrators should revisit the physical and virtual network configuration and try to locate the reason for the reduced performance, which might include speed and duplex settings.

CREATING RESILIENT NETWORKING IN VMWARE ENVIRONMENTS

Networking resilience is a critical part of a virtualized infrastructure, and one that enterprise IT administrators should design and test carefully to provide appropriate functionality and sufficient levels of performance. By following the best practices used by Dell in its own IT environment, organizations can help ensure that they have deployed an efficient, resilient virtualized infrastructure.

Alberto Ramos is a senior solutions architect at the Dell Solutions Innovation Centre in Limerick. Alberto has a degree in Physics from the University of Bilbao as well as VMware Certified Professional (VCP), Microsoft Certified Trainer (MCT), Red Hat Certified Engineer® (RHCE®), EMC® CLARiiON® Technical Architect (EMCTAe), and Cisco Certified Network Associate (CCNA) certifications.

```
C:\>PCATTCP.exe -t -n100000 192.168.0.15
PCAUSA Test TCP Utility V2.01.01.08
TCP Transmit Test
  Transmit    : TCP -> 192.168.0.15:5001
  Buffer Size : 8192; Alignment: 16384/0
  TCP_NODELAY : DISABLED (0)
  Connect     : Connected to 192.168.0.15:5001
  Send Mode   : Send Pattern; Number of Buffers: 100000
  Statistics  : TCP -> 192.168.0.15:5001
819200000 bytes in 25.66 real seconds = 31181.79 KB/sec +++
numCalls: 100000; msec/call: 0.26; calls/sec: 3897.72
```

Figure 9. *Command and output to use the TTCP utility to test network performance between virtual machines*

[1] To download TTCP, visit www.pcausa.com/Utilities/ttcpdown1.htm; to download NPtcp, visit www.rpmfind.net/linux/RPM.

