VMware® Infrastructure 3

Advanced Technical Design Guide

~and~

Advanced Operations Guide

Two books in one!

Ron Oglesby
Scott Herold
Mike Laverick

Chapter 2: Networking

What are vSwitches?

The VMkernel is a very sophisticated kernel. Not only is it adept at running VMs, it can also emulate a virtual switch, or "vSwitch." Virtual Machine's virtual NICs are "plugged in" into virtual switches. These virtual switches are then mapped to physical NICs on the ESX host. This allows many virtual machines to have networking from a single ESX host with relatively few network cards. These vSwitches are not fully-fledged HP ProCurve or Cisco Catalyst switches, so don't expect all the features of a physical switch. Nonetheless, they are VLAN aware and can control outbound traffic using a technology VMware calls "traffic shaping."

There are three types of vSwitch:

1. vSwitch with no NICs mapped to it
2. vSwitch with one physical NIC mapped to it
3. vSwitch with two or more NICs mapped to it

The first type is often referred to as an "internal" vSwitch, as it only allows communication *within* the ESX host. The internal switch could possibly be used as a staging area where you build a virtual machine before "patching" it to production network.

Alternatively, you can use them to emulate a sophisticated network environment. As one vSwitch cannot "auto-magically" communicate with another vSwitch, you could use them to create what VMware calls a "Firewall-in-the-box," linking the various switches together to VMs with more than one NIC containing Network Address Translation (NAT) software with firewall products, such as Microsoft ISA Server or Checkpoint. The only drawback of internal switches is that you cannot carry out VMotion events without first disconnecting users from the virtual machine. In a way this is understandable - they are "locked" within the ESX. They are "internal" to the ESX host, and

therefore we could not 100% guarantee that users would get a continuous connection to the VM during the VMotion event.

The second type gives you basic connectivity to the outside world. This might be suitable for the VMotion network or one that doesn't require fault-tolerance such as a "test" network for test and development of virtual machines. It could be used to allow connectivity to IP based storage such as NAS, if that storage location is not offering anything "mission critical" such as access to an ISO file or VM templates.

The third type gives you fault-tolerance and load-balancing. This is ideal for VMs and IP storage where greater redundancy might be required.

You can have up to 20 physical NICs in an ESX host of any link speed. There are now 56 ports by default on vSwitch and this is configurable for up to 1016 ports. In the past, ESX 2.x users were limited to 8 gigabit or 16 100mps NICs on an ESX host and just 32 ports per vSwitch.

When two virtual machines communicate to each other on the *same* vSwitch, no physical network traffic is generated. The VMkernel moves the data in memory seamlessly from one VM to the other without ever hitting a physical network card. In this respect, you're not limited by the Ethernet access method "carrier sense multiple access with collision detection" (CSMA/CD). If you have two network intensive VMs, locate them on the *same* ESX host on the *same v*Switch. However, watch out for the other core resources as well as CPU, Memory and Disk performance.

Improvements in Networking open a new door of possibilities in ESX, and make certain network tasks much simpler than they have been in the past. For example, enabling IP based load-balancing on a vSwitch with multiple NIC's used to involve the hand editing of text files which can now be easily configured with a couple of clicks of the mouse.

What are Port Groups?

vSwitches can be sub-divided into smaller units called "port groups." While this technology has been in ESX 2.x onwards, it has now been expanded. In the past, ESX 2.x port groups were just a method of allowing virtual machines to interact with VLAN's. They still have this functionality but their purpose includes so much more. We have three types of port groups, each relating to a different type of traffic – with the most important probably being virtual machine traffic:

1. Virtual Machine port group
2. Service Console port group
3. VMkernel (for VMotion/IP Storage) port group

Theoretically, one vSwitch can have many NICs with many port groups, each configured for the Service Console, VMkernel IP Storage, VMotion, and VMs. Although this is certainly possible in ESX, we are still likely to want to maintain physically separate network traffic by NIC card.

GOTCHA:

Names of port groups are very important. They must be consistently named from one ESX host to another; if they are not consistently named you will have problems during VMotion and Cold Migration. Additionally, although the VI client is a Windows application, the definition of a virtual switch is stored on the ESX host in a text file (/etc/vmware/esx.conf). As such, they are also *case-sensitive.* Many people prefer scripted installations to make sure they achieve consistency.

GOTCHA:

One way of resolving this would be to rename port groups which are not a very simple procedure. However, it is not without consequences. When you rename a port group (even simply changing case, for example) virtual machines become "orphaned" from the switch because the name of the switch the VM is attached to is held in the virtual machine's configuration file (.vmx). If I have 32 virtual machines on one ESX host they would each have to be told the new vSwitch

port group name. Some enterprising VMware Community forum members have written looping scripts to handle this for you, but it's perhaps best to avoid this situation in the first place.

Lastly, as we saw in the first chapter of this operations guide, ESX 3.x now has an IP firewall which is configurable through the GUI or command-line. The reasoning for this is twofold- there is now a full IP-stack on ESX which makes it fully NAS and iSCSI aware and because VMware's customers demanded one.

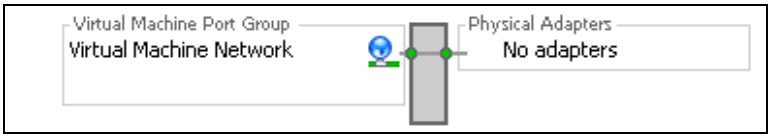
Creating an Internal Only Switch

1. Select your ESX host.
2. Select the Configuration Tab.
3. In the Hardware Pane, select Networking.
4. Click the Add Networking… link.
5. Choose Virtual Machine and Click Next.
6. Remove any tick next to any network adapters.

   Note:

   In Figure 2.1, when you do this, the preview window updates and states under "physical adapters" the words "No Adapters."

Figure 2.1



7. Click Next.
8. In the Port Groups Properties dialog, type a friendly name for this connection, such as internal0.
9. Click Finish.

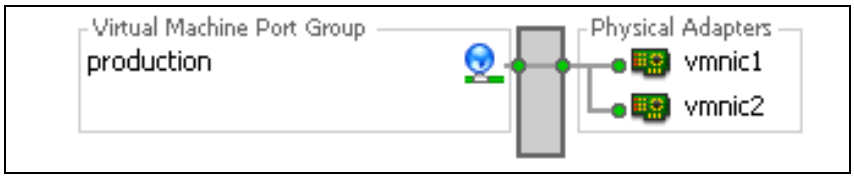
Creating an NIC Team vSwitch

Creating a vSwitch with multiple NICs is as simple as adding an additional tick in a box. By doing this, you will grant every virtual machine on that switch the attributes of fault-tolerance and load-balancing. VMkernel configures this feature for you now, alleviating the use of proprietary wizards like Windows for Intel or Broadcom cards.

1. Select your ESX Host.
2. Select the Configuration Tab.
3. In the Hardware Pane, select Networking.
4. Click the Add Networking… link.
5. Choose Virtual Machine and Click Next.
6. Select two or more NICs.
7. Click Next
8. In the Port Groups Properties dialog, type a friendly name for this connection, such as production.

Figure 2.2 shows a vSwitch with a port group called production configured to use two NICs

Figure 2.2



Creating a vSwitch with VLAN Support

As in ESX 2.x, port groups can only also be used for adding VLAN support. In this case, multiple VMs connect to a single switch containing multiple port groups, each representing the different VLANs available. Actually, ESX support three different methods of enabling access to VLANs.

One method is to simply plug-in the relevant NICs to the relevant VLANs and set the VMs IP settings for that network. However, this method does consume a lot of network cards. For each VLAN you have you would need at least one NIC. If you also factor in the requirement for fault-tolerance, that would mean you need at least one vSwitch per VLAN with two NICs for each vSwitch. Once you get beyond a couple of VLANs you quickly begin to run out of network cards. However, if you only have a small number of VLANs, this "physical" approach has some benefit; it removes the need to speak to the physical switch administrator and is therefore seamless to the networking team. Unfortunately, sometimes avoiding politics and "change management" requests might be your over-riding criteria above and beyond choosing the best technological method.

The port group method requires the fewest network cards and the smallest amount of administration. The downside of this method is that you must persuade your switch administrators to enable IEEE 802.1Q VLAN Tagging on their physical switches.

In 802.1q VLAN Tagging, the network interfaces are plugged into what are called "trunk ports" on the physical switch. Trunk ports allow *many* VLAN packets to traverse them. Even with just one or two network cards, ESX can allow multiple VMs access to many VLANs. There is a very slight CPU burden on the VMkernel, but the overhead is so tiny that they are insignificant.

As each VM communicates with its port group and is about to leave the physical server, the VMkernel adds 4-bytes to the pack. This includes the flag indicating this packet is a VLAN packet and its VLAN number. This packet traverses the trunk port and is intercepted by the physical switch. The physical switch then directs the packet to the appropriate VLAN Broadcast Domain.

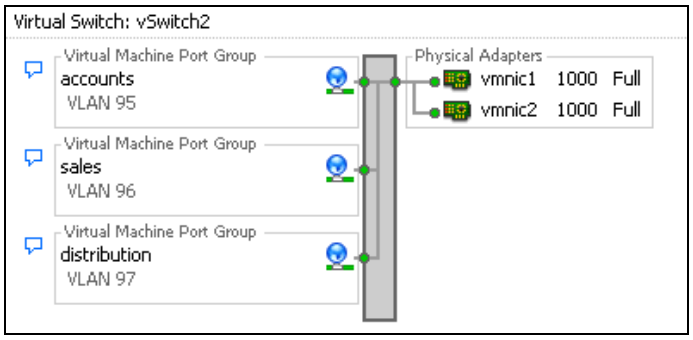
As with NIC teaming, no special work is required of your given guest OS (Windows, Linux, Novell, Solaris); the clever stuff is done by VMkernel, leaving the VM totally unaware it's on a VLAN.

In this example, as I am running out of network ports, I am going to remove the "production" port group to create new port groups with the VLAN information in-place.

1. Select the Configuration Tab.
2. In the Hardware Pane, select Networking.
3. click Properties… of vSwitch2 your “production” Port Group.
4. In the vSwitch2 Properties dialog, click the Production Port Group, and click the Remove button.
5. Then click Add, and in the Add Network Wizard, choose Virtual Machine.
6. In the Port Group Properties dialog, type a friendly name like accounts or vlan95.
7. In the VLAN ID (Optional) field type: 95.
8. Click Next and Finish, repeating this for other VLAN IDs by adding additional port groups.

Figure 2.3 shows this configuration with an additional two port groups called sales and distribution, representing VLAN 96 and 97, respectively.

Figure 2.3



Note:

Personally, I like to name my port groups after the VLAN ID. Even though a description displays in this interface (of VLAN 95 for example), when you plug a VM into the port group all you will see is its friendly name, such as Accounts, Sales, or Distribution.

Creating a VMkernel Switch for VMotion

A very popular configuration is enabling VMkernel port group for use with Vmotion, which is moving a VM from one ESX host to another in real time without powering off the VM. VMotion is essentially a network event – cloning a VM memory from one ESX host to another until they are in the exact state. Once both VMs are identical, the original one can be switched off. This requires a VMkernel port to be configured and gigabit networking between the hosts in question.

When you create a VMkernel port group on vSwitch you will be asked to enter an IP Address, Subnet Mask, and, optionally, a Default Gateway; you may wonder why you're doing this again, since you already set these values during the ESX install. Here, we are setting up an IP configuration for the VMkernel, whereas in the installation we were configuring the IP address of the Service Console merely for management purposes.

1. Select the Configuration Tab.
2. In the Hardware Pane, select Networking.
3. Click the Add Networking… link.
4. Choose VMkernel and Click Next.

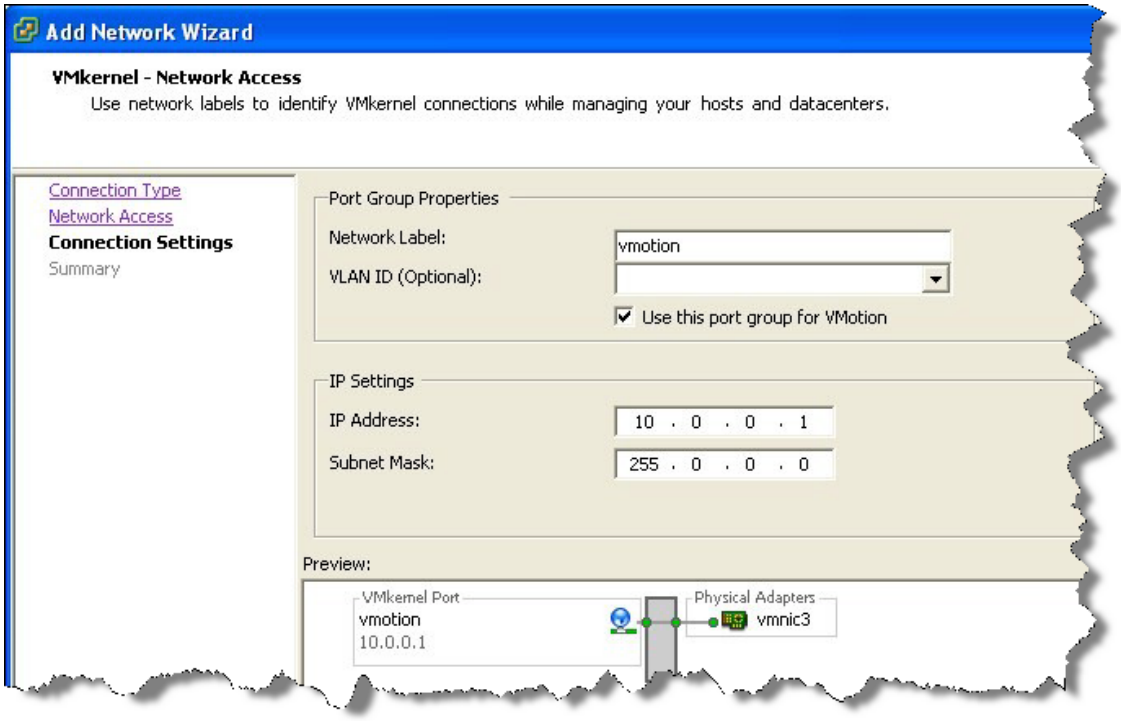
   Note:

   In this case 3 of my 4 NICs are assigned (1 to Service Console, 2 to Production, 0 to Internal). The dialog gives me the option to take NICs assigned to other switches if I wish.

5. Click Next.
6. In the Port Groups Properties dialog, type a friendly name for this connection, such as vmotion.
7. Enable X Use this port group for VMotion.
8. Set an IP Address and Subnet mask for VMotion.

Figure 2.4 shows setting the port group name, and enabling vmotion with the relevant IP settings. I am going to use 10.0.0.1/255.0.0.0 for esx1.vi3book.com and 10.0.0.2/255.0.0.0 for esx2.vi3book.com – and so on.

Figure 2.4



9. You will then receive this message:

   "*There is no default gateway set. You must set a default gateway before you can use this port group. Do you want to configure it now*"

   Note: Does "must" mean must?

   This message can be a little misleading because of the word "must." At this stage, we have been told VMware has no intention of allowing VMotion across routers. So really the dialog box should say "may need." You would only need a default gateway entry if there was a router between the ESX host and the resource being accessed.

   One example of this is when a VMkernel port group is used to access iSCSI and NAS/NFS devices (more about this in the Storage Chapter). In that case you may well need to set the default gateway to cross the router.

This pop-up dialog can be an annoyance because it reappears every time you create VMkernel port group. Sometimes I set up a bogus address just to stop it from appearing.

Broken Network Links

Network links can be broken for many reasons:

- Broken NIC
- Broken Cable
- Switch /Port Failure

Just like many popular operating systems, the VI client will show you if you have network failure by flagging an interface with a red exclamation mark in the network section. Figure 2.5 shows this.

Figure 2.5

Popular vSwitch and Port Group Configurations

Increasing the Number of Ports on vSwitch

In ESX 2.x, we were only allowed 32 virtual machines per vSwitch. You can increase the number of fixed numbers from 24, 56, 120, 248,504 and 1016. What might seem odd about these numbers is they are exactly 8 digits less than what you might expect (32, 64, 128, 256, 512 and 1032). So what happened to the other 8 ports? Well, those 8 ports are there but they are in use by the VMkernel for background monitoring processes.

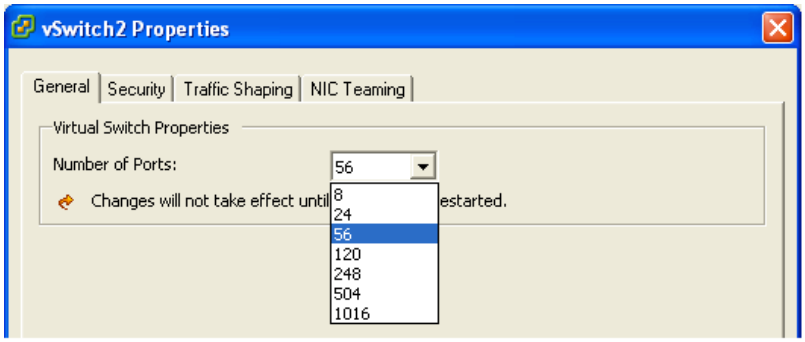
Caution:

Changing this value requires a reboot of ESX for it to take effect.

1. Select your ESX Host.
2. Select the Configuration Tab.
3. In the Hardware Pane, select Networking.
4. Click Properties… allowing you to switch which one contains the port group labeled production (in my case this is vSwitch2).
5. In the dialog box click the Edit… button.
6. Under the General Tab, click the pull down list for the number of ports:

Figure 2.6 shows the pull down list for the number of ports. The message behind reads, "Changes will not take effect until the system is restarted."

Figure 2.6



7. Click OK…

Setting Speed & Duplex on Physical NICs

The IEEE recommends setting the gigabit-to-gigabit to auto-negotiate. In fact, if you want to fix the speed and duplex in gigabit environments, this is *usually* done at the switch, not at the NIC. Even if you fix the speed and duplex settings of gigabit card, it still auto-negotiates anyway. So these settings are really

only relevant perhaps in the Service Console's vSwitch (vSwitch0) where you may perhaps still be using 100mps.

1. Select your ESX Host.
2. Select the Configuration Tab.
3. In the Hardware Pane, select Networking.
4. Click Properties… of vSwitch*N*.
5. Click the Network Adapters Tab.
6. Select the NIC adapter and Click the Edit button.
7. Select the Speed & Duplex and click OK.

   Note:

   For ESX 2.x people - you might be interested to know you can now do this with the Service Console NIC without the need to edit /etc/modules.conf file.

vSwitch and Port group Policies

If you look at the properties of vSwitch *and the* properties of a port group, you will see very similar tabs. These are:

- Security
- Traffic Shaping
- NIC Teaming

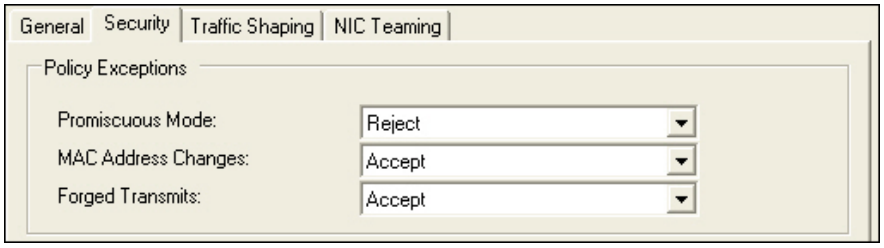
You might ask why they appear twice. Well, the settings on the vSwitch form a "global policy," or rule, if you prefer, whereas the settings on the port group act as exceptions to that global policy. Put another way, you could have settings on vSwitch where "one size fits all," or have per port group settings to give your configuration more flexibility. The settings themselves don't differ; they merely adjust the scope of your changes. As with all security settings, before you decide to close the door you must first be sure that none of the traffic you are disabling is legitimate; the net result being you break one of your applications or services. It's worth saying that these settings are, to some degree, unique to the virtualization world, usually not available elsewhere in the physical world. Let's deal with

the meaning and purpose of each of these dialog boxes in turn, starting with security.

The Security Settings Tab

Prior to the release of ESX 3.x, VMware had an audit of security which formed the basis of these new settings. Previously, these settings weren't configurable at all and if they were you had to manually edit a VMs configuration files (.vmx) to do so. VMware default settings are a compromise (as are all security settings) between security and usability. This is why you don't see *all* security options marked with the reject setting. Figure 2.7 shows the default settings on vSwitch for security.

Figure 2.7



Promiscuous Mode, Default: Reject

As you might know from your "networking essentials" days, promiscuous mode describes a NIC that can collect all network packets, even ones not intended for it. It is generally used in network-sniffing applications that perhaps you use against a firewall to troubleshoot networking and such. Usually, you will find the promiscuous mode in Windows or Linux by looking at the setting of the network card. In VMware's networking architecture this behavior is not allowed. VMware wants to stop the compromised VM from being used as a tool by a potential hacker to attack the rest of the system. Of course, you could always be carrying out packet capturing for legitimate purposes. In that case, you could configure a special port group called "network-analysis" or something equally descriptive, and put a single VM on it.

MAC Address Changes; Default: Accept

Under normal operations, the MAC address of VM does not change. Of course, one thing a hacker may try to do is spoof the MAC address, as they do with an IP Address, to make other systems assume they are sending legitimate traffic.

In a VM, its virtual MAC address is automatically created by algorithm to ensure its uniqueness (the algorithm uses a combination of UUID and unique location of the VM's configuration file). This allows changing the MAC from within the guest operating system, even if the MAC address stored in the VMX file of the VM is different. I've had this situation arise when using Fedora Core 5. Once, I created a brand new VMX file for existing virtual disk. As the Fedora Core 5 operating system loaded, the operating system complained that the MAC of address of my virtual NIC was different from the one stored in the OS. The MAC address of the Fedora Core 5 is also stored in /etc/sysconfig/network-scripts/ifcfg-eth0. Fortunately, I had not altered the default security settings and communication did work, despite the warnings from the guest operating system. If I had changed the setting to 'reject' all *inbound frames* would have been dropped by the ESX host. Later, in the lifetime of this VM, I made the MAC address of the VMX file the same as the MAC address held in ifcfg-eth0. This resolved the warning messages in my VM.

There are some special cases where the MAC address of a VM does change. Microsoft Clustering and VERITAS Clustering are two good examples of services running inside a VM.

Forged Transmits; Default Accept

These are very similar to MAC Address changes, except the restriction is not on the MAC address changing, but rather on a VM being allowed to send traffic under a MAC address which is different from the virtual machines. Again, some "advanced" networking technologies, such as Microsoft's Network Load-balancing, could break if this were configured. If it is set to reject, all outbound frames generated by the VM are dropped by the ESX host.

The Moral of the Story

For the most part, the default settings are good ones – and if in doubt leave them alone. Make sure you thoroughly test your VMs if you do change the default settings.

The Traffic Shaping Tab

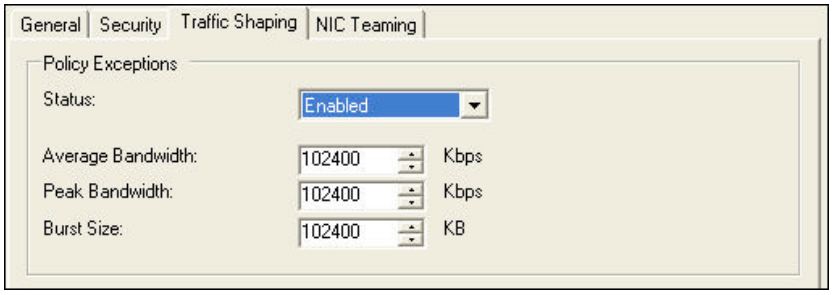
Traffic shaping is the ESX method of controlling outbound traffic generated by VMs heading out of the server on to your wider physical network. You might ask, "Why doesn't VMware have a method to control traffic inbound to the ESX host?" The answer is simple; for the VMkernel to analyze the traffic, it has to arrive at the network interfaces which would cause a performance hit. By then the "damage is already done," so to speak, and limiting can only happen outbound (where the VMkernel has control).

Outbound shaping limits aside, this is an interesting feature. Firstly, it allows us to whittle network performance down to incredibly small amounts – to kps even! Secondly, it acts as a "cap" to network performance. Even if the bandwidth is there, traffic shaped by virtual machines continues to just the amount you have given it. This has some interesting possibilities; the ability to emulate WAN speeds between two ESX hosts, for instance. However, this wouldn't truly reflect the latency and retransmits that typify WAN communications, but it could be used in a pinch. Traffic shaping also comes with some down sides; it's not dynamic - it can't react to changes of circumstances. Once you set the configuration you have some burst rules allowed, but limiting bandwidth based on logic is not available.

Another interesting consideration would be to ask the following question; "What traffic management do we currently do on our LAN?" The answer to this question is nearly none. Most LANs operate on a first-come, first-served basis – and we don't do any traffic management at all. Far from "throttling" our network we try give our VMs as much bandwidth as we possibly can, within reason. Another worry might be that during the hours of 9am-5pm we might wish to throttle network bandwidth used by VM – but what if we are still running backup agents within the VM and running network backups between the hours of 6pm-6am? There isn't any built-in way to indicate when the traffic shaper module is active and non-active.

So, with these caveats in place, how does this feature work? You set three values – an average, a peak, and the burst size. Figure 2.8 shows the standard settings for traffic shaping.

Figure 2.8



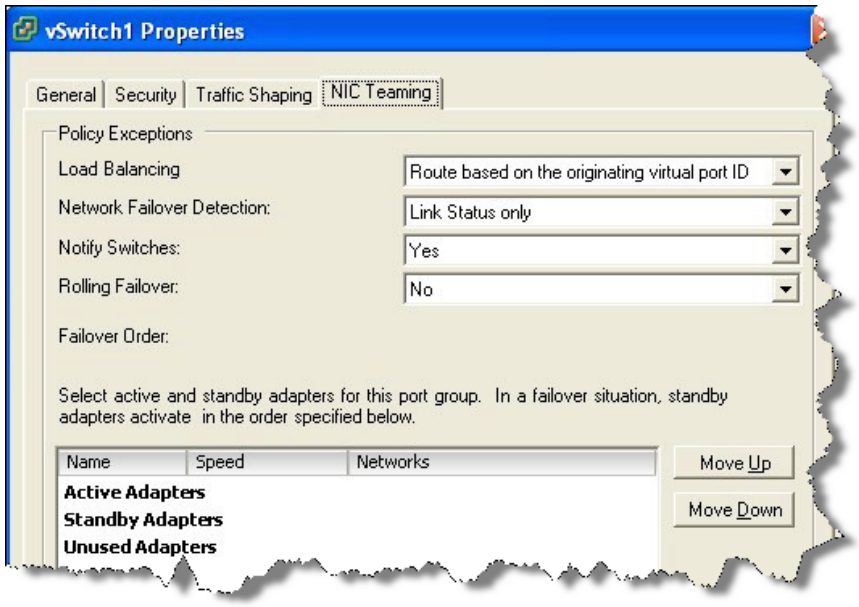
The peak acts a cap – an upper maximum over which the virtual machines cannot rise. The burst acts a little like a window size. As all bandwidth is measured by time over the amount of data we can send (57kps, 512kps), the burst size basically controls this. When the burst size is full, i.e. we have used all of our allocation, then ESX pushes the VMs back down under their average line. In this respect, peak is always more than average, and it's actually the average value that acts as throttle on network bandwidth.

The NIC Teaming Tab

This is a busy dialog box which has lots of very interesting settings. Having said this, I only change two on a regular basis. Some of the options are Boolean – by which I mean there are only two choices, such as "Network Failover Detection" and "Notify Switches." Alternatively, some of the settings have 4 options associated with them, such as the "Load-balancing" options. I want to clear the simpler settings, then move on to more complex settings, so I can gradually drill down to increasing complexity – rather than jumping in head first.

Figure 2.9 shows the standard default settings for the NIC Teaming Tab.

Figure 2.9



Network Failover Detection

This setting controls how ESX detects a network failure, thus triggering the movement of packets from one adapter to another in vSwitch with more than one NIC attached to it. I usually change this to "Beacon Probing" from its default of "Link Status Only." These settings control how ESX detects a network failure. The default is very similar to the guest OS systems you run. This will detect a failed NIC, failed cable, or failure of a switch/hub to which the server is directly attached. After that, "link status only" has no idea if other paths are valid. In contrast, "Beacon Probing" is able to see beyond the first uplink to the physical switch by sending a packet between the NICs in the vSwitch. This has some issues; a dropped packet, regardless of the reason, can make the system think a nic is down when it isn't. Beacon probing is rarely used in production ESX environments, and I won't be recommending it here.

Notify Switches

The "yes" default setting here indicates that when failures occur, a Reverse Address Resolution Protocol packet (ARP is the protocol that relates your MAC

address to IP Address) is sent out to physical switches. This updates the lookup tables on physical switches that VMs packets must be sent to different physical NICs because the original network path is now unavailable. This is a good setting because it ensures that packets will be successfully delivered and reduces the chance of any lost packets during Vmotion. In VMotion the VMs virtual MAC address remains unchanged, once the VM has been moved from one ESX host to another; the *physical* MAC addresses have most definitely changed. Additionally, notifications are issued whenever you add or remove a physical NIC from the vSwitch. For the most part this setting is a good one and should not be changed unless you are using Microsoft Network Load-balancing (NLB) within a VM. This setting is incompatible with Microsoft NLB when NLB is set to unicast mode. However, if you are using Microsoft NLB in multicast mode there is not a problem.

Load-Balancing within a vSwitch

Once you create a vSwitch with more than one network card you have, by default, created a format of load-balancing. In this dialog box you can tweak the network settings considerably. There are in fact, four different ways of configuring multiple NICs with a vSwitch. They are:

1. Originating Port ID
2. Source MAC Address Hash
3. IP Hash
4. Explicit Failover

In reality, options 1-3 represent real load-balancing options, whereas "Explicit Fail Over" allows us to manually configure how multiple cards are used and works in conjunction with the "Rolling Failover" setting and the active/standby feature. Let's deal with Option 1-3 in the first instance.

Originating Port

Basically, this algorithm makes each VM use the next available NIC in the bundle. It's a bit like a round-robin effect; if I had a virtual switch with three NICs mapped to it, it would cycle between them. So VM1 would use vmnic1; VM2 would use vmnic2, and then VM3 would use vmnic3. After that VM4, VM5,

and VM6 would use vmnic1, 2, and 3. The effect is a crude form of load-balancing on each of the NICs. Sounds good, doesn't it? In most cases, it is just fine and a default used in 90+% of environments. It should be noted though, that there is no awareness of link speeds, how heavily saturated a link is, or the latency on a given uplink. Its main merit is its compatibility with all physical switches and reduces complexity on the networking side.

Source MAC Address Hash

This method used to be the default in ESX 2.x and is an algorithm that does a computation via the MAC address. This is somewhat more intelligent than "originating port." However, as routers (and some switches) hide the MAC address of the source MAC address, it is load-balancing with limits. The algorithm simply isn't intelligent enough to work out the best "trip" for the packet with the limited amount of information it has at its disposal.

IP Hash

This is the Rolls Royce of load-balancing. It uses the source and destination IP address to work out the best trip for the packet to take. It is aware of latency because it is IP based – and it's also aware when a link is saturated. Lastly, the IP hash is aware of clients that have communicated across a router to the VM running on an ESX host. With all these attributes at its disposal, the IP hash has much better knowledge of the effect of network traffic on overall performance. The trade-off is a more complex configuration that needs to be maintained. This configuration is used in a small number of environments, and often (for our customers) found to be too complex for the benefits seen when weighed against the needs of the VMs. I mean, really, have you looked at network utilization on 95% of your servers? They often average below 5MB.

GOTCHA:

Again, as with some of the security settings, the IP Hash method is incompatible with Microsoft Network Load Balancing (NLB) in unicast mode. If you have NLB working in multicast mode there is not a problem.

Explicit Fail-Over

This is not so much a load-balancing technique as a stricter way of controlling what happens when NICs fail. If you engage it, you have two other settings: "Rolling Failover" and Active/Standby Adapter section of the dialog box.

Explicit Failover, as the name suggests, allows you to set so many adapters as "active" and so many adapters as "standby." The active ones can have load-balancing features (as long as there is more than one listed!), and the standby NICs only engage if the actives fail. Explicit failover does not wait for all the active adapters to fail before engaging the standbys. As soon as one active NIC fails, a standby is selected from the order in the list and is used.

Rolling Failover controls what happens when a failed NIC becomes available again. It's actually quite a confusing label for VMware to use. This is because far from controlling failover, it is actually used to control fail back settings. That is to say – what happens when an active NIC fails, and then later what happens when that NIC returns to an active state.

If you set rolling failover to "no" when a failed NIC comes on stream again the "active" adapter can be used immediately, taking over the role of the standby adapter. If rolling failover is set to "yes" the NIC that comes back on stream again is not used until another "active" adapter fails. The assumption here is that if you have lost an NIC, you might begin to distrust its reliability so setting "yes" would stop it being used again – unless you had another failure. One way of using this feature is when you have a team of two NICs and one is 1000mps and the other is 100mps. You would make the 1000mps card active, and put the 100mps card on standby. Rolling failover would be set to "no." Now, if the 1000mps card failed it would failover the 100mps card when the 1000mps card was active again, ESX would return to the "home" adapter.

Using this feature allows you to have very good control over which VMs use which network card because we now set these parameters on port groups. In the past, these settings were only available on a vSwitch. This was a bit "one size fits all." The downside of this kind of configuration is that could get very complicated. You might prefer to follow the "KISS" principle – keep it simple, stupid.

Using all the features together

I could create one vSwitch with three port groups – accounts, sales and distribution each with a separate VLAN ID. However, the vSwitch could have 6 physical network interfaces which are set-up with "trunk ports." Therefore, any one of these NICs could be used to send data to the correct VLAN as long as the right VMs are plugged into the right port groups. I could use "explicit failover" to allocate which was their preferred NIC like this:

Accounts port group

Active: vmnic1, vmnic2

Standby: vmnic3, vmnic4, vmnic5, vmnic6

Sales port group

Active: vmnic3, vmnic4

Standby: vmnic1, vmnic2, vmnic5, vmnic6

Distribution port group

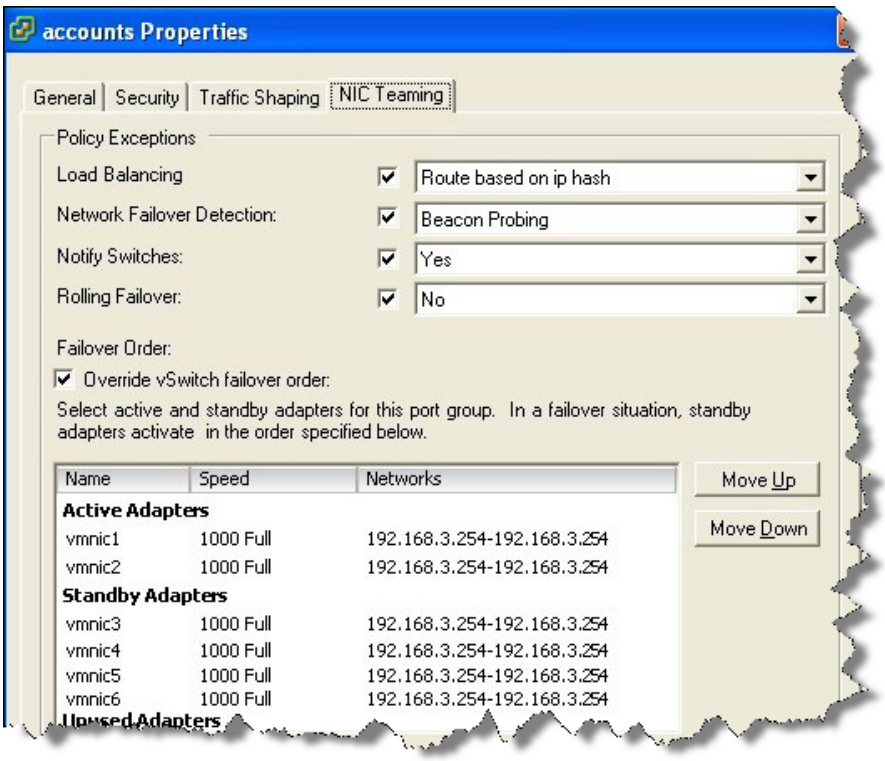
Active: vmnic5, vmnic6

Standby: vmnic1, vmnic2, vmni3, vmnic4

Figure 2.10 shows this particular configuration. Normally, each port group has its own dedicated NICs which are configured for "IP hash" load-balancing, and Beacon Probing has been enabled. Under normal operations the accounts port group would prefer to use vmnic1 and vmnic2. This means that if there was a lot of network activity within the sales or distribution port groups it would have no impact as they prefer to use vmnic3/4 and vmnic5/6. We get excellent redundancy because there are six NICs that could be possibly used at any one time. Rolling failover has been set to "no" so if vmnic1 or 2 go offline, when

they come back on stream again we get the separation of network traffic under normal operations. This configuration also allows for specific security and traffic shaping settings to be applied to accounts (and the virtual machines configured to use it) without it affecting the sales and distribution port groups.

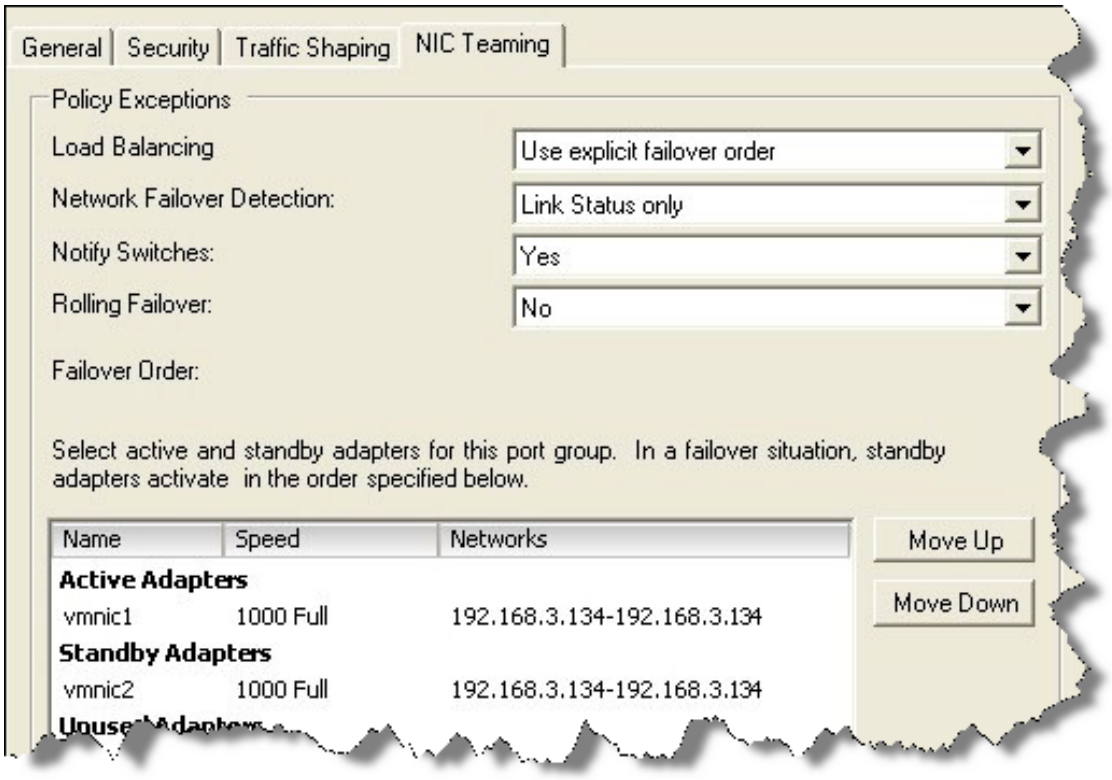
Figure 2.10



Creating Standby vSwitches

It is possible to set up a configuration by which a vSwitch has more than one adapter where some of the NICs are active and some of the NICs are standby. The idea behind this is to guarantee a degree of quality of service. So, if you lost one NIC you could replace it with the same level of bandwidth. To do this, you configure a vSwitch to have more than one NIC and set some to be Active and others to be standby; next, set NIC Teaming options for Load-balancing to be "Use Explicit Failover order." Figure 2.11 shows the configuration in question.

Figure 2.11



After clicking "OK," the VI client will refresh and indicate which NIC is active and which is standby. Figure 2.12 and Figure 2.13 show this configuration, when the vmnic1 is functioning, and when failover has occurred because it has malfunctioned, respectively.

Figure 2.12 without a failover

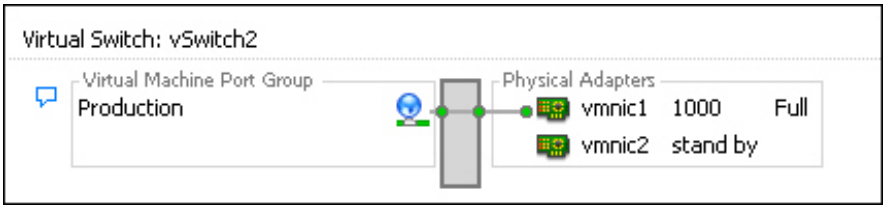
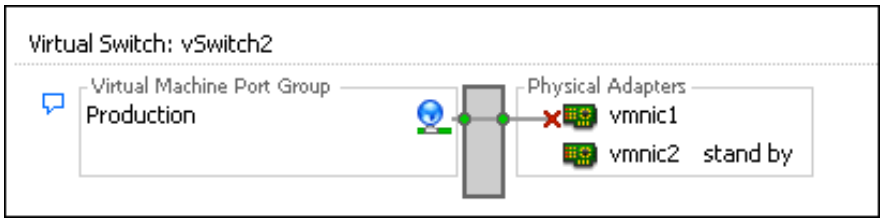
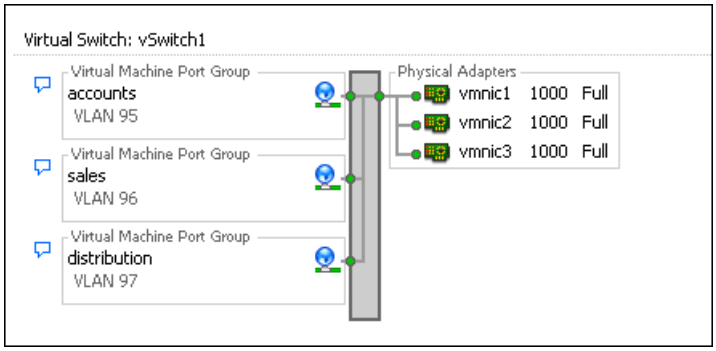


Figure 2.13 with a failover



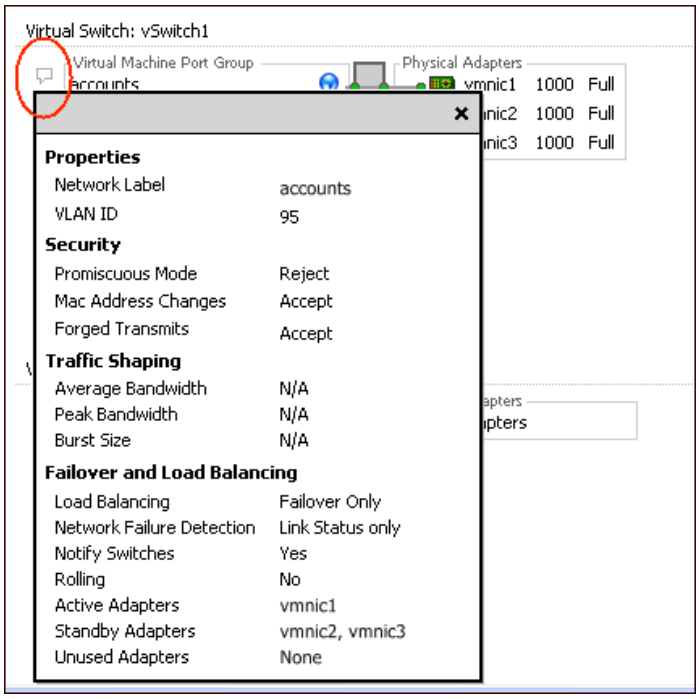
As illustrated, it is very easy to see the settings if active and standby NICs are configured at vSwitch level. It is less clear in the VI client if you enable active and standby at port group level. Figure 2.14 shows a vSwitch with many port groups, and I have configured different active and standby settings. As you can see, it is not immediately clear what kind of configuration I've created with respect to active and standby adapters.

Figure 2.14



By clicking the small blue "speech bubble" icon next to each port group it is possible to view each setting. This stops you from unnecessarily navigating through layers of dialog boxes, properties options, and edit buttons to view your settings. Figure 2.15 shows the pop-up settings box.

Figure 2.15

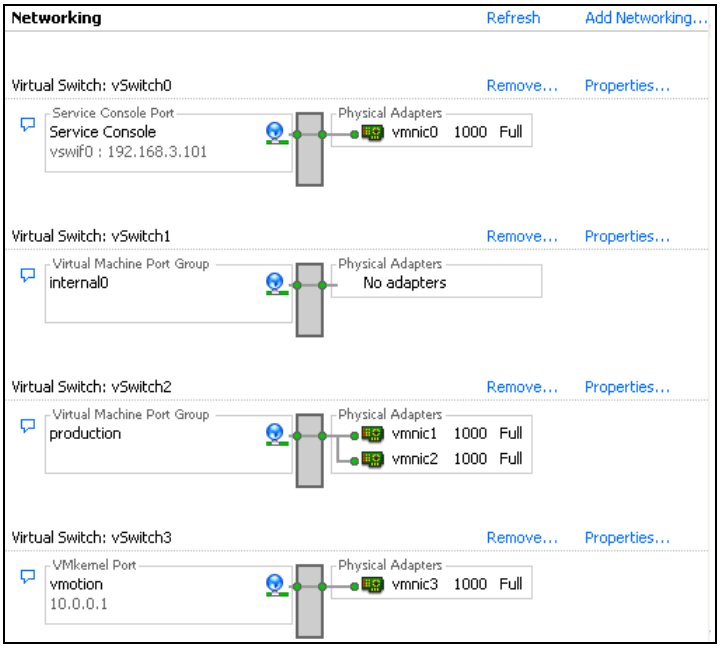


Managing Service Console Networking

Meanwhile, back in the real world, all my NICs are now fully allocated on my 4-NIC server. If I want to configure other possibilities, I might have to free up NICs by removing one NIC from production vSwitch or by removing the VMotion switch I just created to free up its NIC.

Figure 2.16 shows what my cleanly installed ESX server looks like now from a networking perspective:

Figure 2.16



Now I want to show you some configuration settings related specifically to the Service Console, such as adding a "backup" Service Console connection and how to modify your IP settings originally configured during the installation.

Creating a "Backup" Port group for the Service Console

Up until ESX 3.x, there's always been a single point of failure on the ESX host - the Service Console network. If the single NIC (vmnic0) on vSwitch0 failed we would have no management over the ESX host except by its ILO card. Remember, if our VMs were on a different vSwitch configured with many vmNICs they would remain unaffected.

In ESX 3.x the networking architecture treats the Service Console as if it were just another VM connected to a virtual switch. This default switch is called vSwitch0 and, incidentally, the default number of ports on the vSwitch is just 24 rather than 56.

These Service Console ports have a special name; "vswif." which stands for **V**irtual **Sw**itch **I**nter**f**ace. They are serialized as you create them, with the first being vswif0, the second being vswif1, and so on.

If you have “spare” network adapters, it’s very easy to protect the Service Console network by adding another vSwitch with the Service Console port group – or adding an additional Service Console port group to an existing vSwitch.

1. Select the Configuration Tab.
2. In the Hardware Pane, select Networking.
3. Select the Properties… link next to the production vSwitch.
4. Select Service Console.

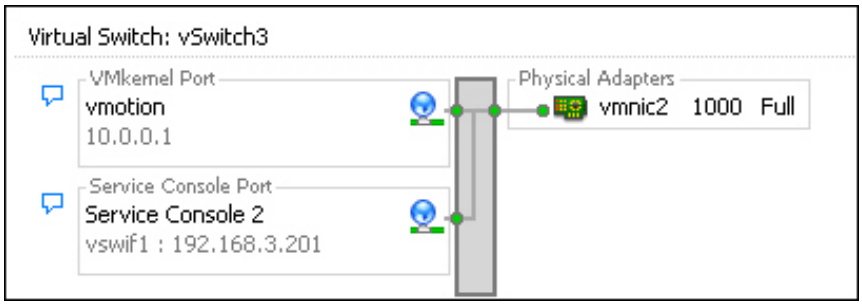
   Note:

   As we already have a service Console port, the system names this Service Console 2.

5. Click Next.
6. Complete the IP Settings as befits your network infrastructure.
7. Click Next.

Figure 2.17 shows how I have created a backup Service Console port group on my VMotion switch. This is safe because it’s not my primary network for managing VMs and VMotion is not event that happens very frequently. In Chapter 3, on Storage, you will learn that a second Service Console network is often required to configure the iSCSI Software Adapter. Additionally, in Chapter 10, which deals with High Availability, you will learn that this “backup” Service Console network can mitigate an unwanted event called the “split brain” phenomena in VMware HA.

Figure 2.17



Note:

It's a good idea to confirm you can connect to this new port with ping/Putty and by pointing the VI client to this IP/Name.

Note:

As another way of "protecting" the Service Console network, try just by adding an additional NIC vSwitch; by doing so you will create a bond for the Service Console. However, most people would agree if I said this was a waste of bandwidth. If we have spare NICs we should really allocate them to the VMs. After all, they are the ones that require bandwidth and fault-tolerance.

Adding a second NIC is also a safer procedure than switching the NIC on the Service Console switch. For example, add in the second NIC (100mps) and remove the first NIC (1000mps) and you shouldn't lose connectivity to the Service Console.

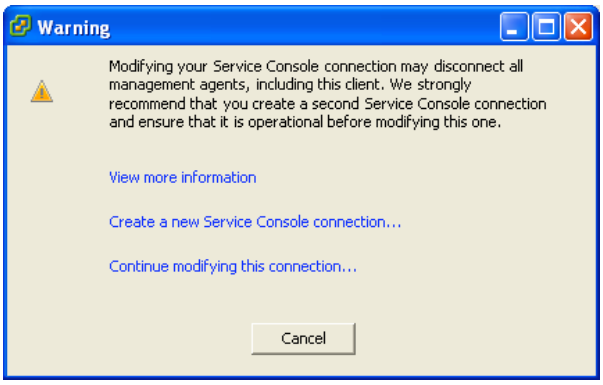
Changing the Service Consoles Network Settings

One reason to change your Service Console's settings is to change your IP Address, Subnet Mask, or Default Gateway. One of the dangers is if you make a mistake, it's a bit like sitting on a tree branch with a saw and sawing your way through the very branch that is holding you there!

If you are going to modify the primary connection for the Service Console, this could cause a disconnection. If this change goes horribly wrong, it is not the end of the world – we will be able to use the backup Service Console 2 network connection.

If you don't have a backup connection you will receive a warning. Figure 12.18 shows you the warning.

Figure 2.18



This dialog box *only* appears if you do not have a second "backup" Service Console connection.

The safest method is to connect your backup Service Console 2 network to the ESX host and then change Service Console Properties. This way you will not be disconnected in the process of changing the IP Address. If you do make a mistake you still have your "backup" connection in place.

1. Close your VI client.
2. Open the VI client using the Backup Service Console IP Address in the Server.
3. Select the Configuration Tab.
4. In the Hardware Pane, select Networking.
5. Select Properties… of vSwitch0 with the Port Group name of Service Console.
6. In the dialog box, select the Port Group of Service Console, and click Edit.
7. In the Service Console Properties Dialog box you can change your IP settings.

Troubleshooting the Service Console Networking

During the installation you may select the NIC used for the Service Console and also configure its IP settings. Occasionally, people get this wrong and select both the entirely wrong NIC and IP values. When this happens you have three options:

1. Re-install, and try again.
2. Walk to the server room and re-route the network cables to the correct switch.
3. Connect via ILO, and use command-line tools to correct your mistakes.

As you might gather, step 3 is the recommended method. Remember, if you are using the command-line it IS case-sensitive.

Correcting the "wrong NIC selected during installation" problem

We can rectify this problem by gradually adding in NICs while pinging the Service Console's IP address (assuming that the IP settings are correct). I normally use ping –t to do this – and as I add the NICs in, I can judge by whether I get a reply as an indication that I have found the correct network card. First, I type the command which lists my vSwitch configuration using the –l switch.

esxcfg-vswitch –l

This produces the output displayed in Figure 2-19. Here you can see that there is one vSwitch, which is vSwitch0, with one port group called "Service Console." The installer has selected, by default, the first NIC it could find – vmnic0.

Figure 2.19

```
Switch Name Num Port    Used Ports  Uplink
vSwitch0    32          3           vmnic0
```

To see what NICs are available including settings such as vmnic name, PCI Bus details, driver, link speed, duplex, and vendor make and model , you can use the command

esxcfg-NICs –l

To link another NIC to the switch we can use the –L switch.

esxcfg-vswitch -L vmnic1 **vSwitch0**

I keep on doing this until I have a response. Once I have worked out, by a process of elimination, the correct NIC I unlink the NICs which were patched to vSwitch0/vswif0 using the –U switch.

esxcfg-vswitch -U vmnic0 **vSwitch0**

Using the esxcfg-vswitch –l, -L and –U command you can quickly change the preferred NIC for vSwitch0 and be up and running with the VI client in a short period of time.

Correcting your IP Settings

Occasionally people type the wrong IP address or subnet mask. One common problem that people have is not noticing that the ESX installer does some work for you by automatically setting the default gateway based on the IP address and subnet mask. If you set 192.168.3.101 and 255.255.255.0 as your IP address and subnet mask, respectively, the installer will pre-fill 192.168.3.254 as your default gateway and 192.168.3.1 as your primary DNS server.

Fortunately, if you have ILO or physical access to the server these settings are very easy to correct, either with esxcfg command-line tools or with a text editor to modify the configuration files. Again, you could use a ping –t against the IP address of the Service Console to check to see if you get responses.

To view your current IP address and subnet mask you can use the –l switch on the esxcfg-vswif command like so:

esxcfg-vswif –l

To change them, you can use the command with –i and –n switch. You also need to indicate which vswif interface you are changing; as you know you can have more than one for backup purposes.

esxcfg-vswif vswif0 –i 192.168.3.104 –n 255.255.255.0

If your problems reside with your default gateway or you incorrectly set the DNS settings, these can be fixed with a text editor. You could use a text called Vi if you wish; personally, I prefer to use a tool called nano. Linux users prefer to use Vi, as it is on every single Linux distribution, whereas nano only exists on Redhat Linux. If you do use nano remember to use the –w switch. This switches off the "word-wrap" feature which could corrupt the file if you're not careful.

If you're a novice and don't use the command-line very frequently, the nano tool should be sufficient for your infrequent usage. To correct your default gateway you need to edit the "network" file with nano.

nano –w /etc/sysconfig/network

Modify your default gateway, and exit and save using [ctrl+x] to exit nano, [Y] to accept changes, and "enter" to overwrite the original file. To make this change take effect we need to restart the networking for the Service Console with Redhat Linux's "service" command (again, the service command is not available on all Linux distributions but is available within the ESX Service Console).

service network restart

If your problems reside with the DNS settings you can edit these with

nano –w /etc/resolv.conf

Generally, if you have basic connectivity to the Service Console, many of these settings can actually be altered through the VI client. I've chosen to show you the esxcfg command because these may help in extreme cases where you simply cannot connect to your ESX host with the VI client!

If you do wish to change your default gateway, ESX host name, or DNS settings, navigate to the **Configuration Tab**, then the **Software Pane**, select DNS and Routing and choose **Properties…** in the top right-hand corner.

Using Third Party ESX Host Network Tools

By now you are probably noticing a significant administrative overhead in the area of configuring networking. Each ESX host's networking is configured separately and the VI client is not really geared up for "bulk" edits or multiple edits of many ESX hosts. This is especially burdensome considering that one of things I have stressed is the importance of consistently labeled port groups. There are two approaches to this problem right now.

In the first approach, you could script your installations and use command-line tools such as esxcfg-vswitch, esxcfg-vswif, and esxcfg-vmknic. The trouble with these tools is that they do not allow for complete configuration of all the network features. For example, none of these tools allow you to enable VMotion on a VMkernel port group. This is because their design by VMware's intention) was to use them for "troubleshooting" purposes only.

The second approach is using a third party tool that does support multiple selections for creating switches. This is also useful if you need to reconfigure ESX host sometime after their initial configuration by scripting or by hand. This tool is called VM Client and was written by Flores Eken of the Netherlands, who currently works for a consulting firm called ITQ. Flores' work is hosted on:

http://www.run-virtual.com/?page_id=160

Right now the tool is great for modifying port groups across multiple switches and setting additional information, like VLAN information. What it lacks is the ability to create vSwitches.