Chapter 6. Virtual networking

Table of Contents

As briefly mentioned in Section 3.9, “Network settings”, VirtualBox provides up to eight virtual PCI Ethernet cards for each virtual machine. For each such card, you can individually select

1. the hardware that will be virtualized as well as
2. the virtualization mode that the virtual card will be operating in with respect to your physical networking hardware on the host.

Four of the network cards can be configured in the "Network" section of the settings dialog in the graphical user interface of VirtualBox. You can configure all eight network cards on the command line via VBoxManage modifyvm; see Section 8.8, “VBoxManage modifyvm”.

This chapter explains the various networking settings in more detail.

6.1. Virtual networking hardware

For each card, you can individually select what kind of *hardware* will be presented to the virtual machine. VirtualBox can virtualize the following six types of networking hardware:

- AMD PCNet PCI II (Am79C970A);
- AMD PCNet FAST III (Am79C973, the default);
- Intel PRO/1000 MT Desktop (82540EM);
- Intel PRO/1000 T Server (82543GC);
- Intel PRO/1000 MT Server (82545EM);
- Paravirtualized network adapter (virtio-net).

The PCNet FAST III is the default because it is supported by nearly all operating systems out of the box, as well as the GNU GRUB boot manager. As an exception, the Intel PRO/1000 family adapters are chosen for some guest operating system types that no longer ship with drivers for the PCNet card, such as Windows Vista.

The Intel PRO/1000 MT Desktop type works with Windows Vista and later versions. The T Server variant of the Intel PRO/1000 card is recognized by Windows XP guests without additional driver installation. The MT Server variant facilitates OVF imports from other platforms.

The **"Paravirtualized network adapter (virtio-net)"** is special. If you select this, then VirtualBox does *not* virtualize common networking hardware (that is supported by common guest operating systems out of the box). Instead, VirtualBox then expects a special software interface for virtualized environments to be provided by the guest, thus avoiding the complexity of emulating networking

hardware and improving network performance. Starting with version 3.1, VirtualBox provides support for the industry-standard "virtio" networking drivers, which are part of the open-source KVM project.

The "virtio" networking drivers are available for the following guest operating systems:

- Linux kernels version 2.6.25 or later can be configured to provide virtio support; some distributions also back-ported virtio to older kernels.
- For Windows 2000, XP and Vista, virtio drivers can be downloaded and installed from the KVM project web page.[30]

VirtualBox also has limited support for so-called **jumbo frames**, i.e. networking packets with more than 1500 bytes of data, provided that you use the Intel card virtualization and bridged networking. In other words, jumbo frames are not supported with the AMD networking devices; in those cases, jumbo packets will silently be dropped for both the transmit and the receive direction. Guest operating systems trying to use this feature will observe this as a packet loss, which may lead to unexpected application behavior in the guest. This does not cause problems with guest operating systems in their default configuration, as jumbo frames need to be explicitly enabled.

6.2. Introduction to networking modes

Each of the eight networking adapters can be separately configured to operate in one of the following modes:

Not attached

> In this mode, VirtualBox reports to the guest that a network card is present, but that there is no connection -- as if no Ethernet cable was plugged into the card. This way it is possible to "pull" the virtual Ethernet cable and disrupt the connection, which can be useful to inform a guest operating system that no network connection is available and enforce a reconfiguration.

Network Address Translation (NAT)

> If all you want is to browse the Web, download files and view e-mail inside the guest, then this default mode should be sufficient for you, and you can safely skip the rest of this section. Please note that there are certain limitations when using Windows file sharing (see Section 6.3.3, “NAT limitations” for details).

NAT Network

> The NAT network is a new NAT flavour introduced in VirtualBox 4.3. See 6.4 for details.

Bridged networking

> This is for more advanced networking needs such as network simulations and running servers in a guest. When enabled, VirtualBox connects to one of your installed network cards and exchanges network packets directly, circumventing your host operating system's network stack.

Internal networking

> This can be used to create a different kind of software-based network which is visible to selected virtual machines, but not to applications running on the host or to the outside world.

Host-only networking

> This can be used to create a network containing the host and a set of virtual machines, without the need for the host's physical network interface. Instead, a virtual network interface (similar to a loopback interface) is created on the host, providing connectivity among virtual machines and the host.

Generic networking

> Rarely used modes share the same generic network interface, by allowing the user to select a driver which can be included with VirtualBox or be distributed in an extension pack.

At the moment there are potentially two available sub-modes:

UDP Tunnel

This can be used to interconnect virtual machines running on different hosts directly, easily and transparently, over existing network infrastructure.

VDE (Virtual Distributed Ethernet) networking

This option can be used to connect to a Virtual Distributed Ethernet switch on a Linux or a FreeBSD host. At the moment this needs compiling VirtualBox from sources, as the Oracle packages do not include it.

The following table provides a quick overview of the most important networking modes:

Table 6.1. Overview

	VM ↔ Host	VM1 ↔ VM2	VM → Internet	VM ← Internet
Host-only	+	+	–	–
Internal	–	+	–	–
Bridged	+	+	+	+
NAT	–	–	+	Port forwarding
NAT Network	–	+	+	Port forwarding

The following sections describe the available network modes in more detail.

6.3. Network Address Translation (NAT)

Network Address Translation (NAT) is the simplest way of accessing an external network from a virtual machine. Usually, it does not require any configuration on the host network and guest system. For this reason, it is the default networking mode in VirtualBox.

A virtual machine with NAT enabled acts much like a real computer that connects to the Internet through a router. The "router", in this case, is the VirtualBox networking engine, which maps traffic from and to the virtual machine transparently. In VirtualBox this router is placed between each virtual machine and the host. This separation maximizes security since by default virtual machines cannot talk to each other.

The disadvantage of NAT mode is that, much like a private network behind a router, the virtual machine is invisible and unreachable from the outside internet; you cannot run a server this way unless you set up port forwarding (described below).

The network frames sent out by the guest operating system are received by VirtualBox's NAT engine, which extracts the TCP/IP data and resends it using the host operating system. To an application on the host, or to another computer on the same network as the host, it looks like the data was sent by the VirtualBox application on the host, using an IP address belonging to the host. VirtualBox listens for replies to the packages sent, and repacks and resends them to the guest machine on its private network.

The virtual machine receives its network address and configuration on the private network from a DHCP server integrated into VirtualBox. The IP address thus assigned to the virtual machine is usually on a completely different network than the host. As more than one card of a virtual machine can be set up to use NAT, the first card is connected to the private network 10.0.2.0, the second card to the network 10.0.3.0 and so on. If you need to change the guest-assigned IP range for some reason, please refer to Section 9.11, “Fine-tuning the VirtualBox NAT engine”.

6.3.1. Configuring port forwarding with NAT

As the virtual machine is connected to a private network internal to VirtualBox and invisible to the host, network services on the guest are not accessible to the host machine or to other computers on the

same network. However, like a physical router, VirtualBox can make selected services available to the world outside the guest through **port forwarding.** This means that VirtualBox listens to certain ports on the host and resends all packets which arrive there to the guest, on the same or a different port.

To an application on the host or other physical (or virtual) machines on the network, it looks as though the service being proxied is actually running on the host. This also means that you cannot run the same service on the same ports on the host. However, you still gain the advantages of running the service in a virtual machine -- for example, services on the host machine or on other virtual machines cannot be compromised or crashed by a vulnerability or a bug in the service, and the service can run in a different operating system than the host system.

To configure Port Forwarding you can use the graphical Port Forwarding editor which can be found in the Network Settings dialog for Network Adaptors configured to use NAT. Here you can map host ports to guest ports to allow network traffic to be routed to a specific port in the guest.

Alternatively command line tool `VBoxManage` could be used; for details, please refer to Section 8.8, "VBoxManage modifyvm".

You will need to know which ports on the guest the service uses and to decide which ports to use on the host (often but not always you will want to use the same ports on the guest and on the host). You can use any ports on the host which are not already in use by a service. For example, to set up incoming NAT connections to an `ssh` server in the guest, use the following command:

```
VBoxManage modifyvm "VM name" --natpf1 "guestssh,tcp,,2222,,22"
```

With the above example, all TCP traffic arriving on port 2222 on any host interface will be forwarded to port 22 in the guest. The protocol name `tcp` is a mandatory attribute defining which protocol should be used for forwarding (`udp` could also be used). The name `guestssh` is purely descriptive and will be auto-generated if omitted. The number after `--natpf` denotes the network card, like in other parts of VBoxManage.

To remove this forwarding rule again, use the following command:

```
VBoxManage modifyvm "VM name" --natpf1 delete "guestssh"
```

If for some reason the guest uses a static assigned IP address not leased from the built-in DHCP server, it is required to specify the guest IP when registering the forwarding rule:

```
VBoxManage modifyvm "VM name" --natpf1 "guestssh,tcp,,2222,10.0.2.19,22"
```

This example is identical to the previous one, except that the NAT engine is being told that the guest can be found at the 10.0.2.19 address.

To forward *all* incoming traffic from a specific host interface to the guest, specify the IP of that host interface like this:

```
VBoxManage modifyvm "VM name" --natpf1 "guestssh,tcp,127.0.0.1,2222,,22"
```

This forwards all TCP traffic arriving on the localhost interface (127.0.0.1) via port 2222 to port 22 in the guest.

It is possible to configure incoming NAT connections while the VM is running, see Section 8.13, "VBoxManage controlvm".

6.3.2. PXE booting with NAT

PXE booting is now supported in NAT mode. The NAT DHCP server provides a boot file name of the form `vmname.pxe` if the directory `TFTP` exists in the directory where the user's `VirtualBox.xml` file is kept. It is the responsibility of the user to provide `vmname.pxe`.

6.3.3. NAT limitations

There are four **limitations** of NAT mode which users should be aware of:

ICMP protocol limitations:

Some frequently used network debugging tools (e.g. `ping` or tracerouting) rely on the ICMP protocol for sending/receiving messages. While ICMP support has been improved with VirtualBox 2.1 (`ping` should now work), some other tools may not work reliably.

Receiving of UDP broadcasts is not reliable:

The guest does not reliably receive broadcasts, since, in order to save resources, it only listens for a certain amount of time after the guest has sent UDP data on a particular port. As a consequence, NetBios name resolution based on broadcasts does not always work (but WINS always works). As a workaround, you can use the numeric IP of the desired server in the `\\server\share` notation.

Protocols such as GRE are unsupported:

Protocols other than TCP and UDP are not supported. This means some VPN products (e.g. PPTP from Microsoft) cannot be used. There are other VPN products which use simply TCP and UDP.

Forwarding host ports < 1024 impossible:

On Unix-based hosts (e.g. Linux, Solaris, Mac OS X) it is not possible to bind to ports below 1024 from applications that are not run by `root`. As a result, if you try to configure such a port forwarding, the VM will refuse to start.

These limitations normally don't affect standard network use. But the presence of NAT has also subtle effects that may interfere with protocols that are normally working. One example is NFS, where the server is often configured to refuse connections from non-privileged ports (i.e. ports not below 1024).

6.4. Network Address Translation Service

The Network Address Translation (NAT) service works in a similar way to a home router, grouping the systems using it into a network and preventing systems outside of this network from directly accessing systems inside it, but letting systems inside communicate with each other and with systems outside using TCP and UDP over IPv4 and IPv6.

A NAT service is attached to an internal network. Virtual machines which are to make use of it should be attached to that internal network. The name of internal network is chosen when the NAT service is created and the internal network will be created if it does not already exist. An example command to create a NAT network is:

```
VBoxManage natnetwork add --netname natnet1 --network "192.168.15.0/24" --enable
```

Here, "natnet1" is the name of the internal network to be used and "192.168.15.0/24" is the network address and mask of the NAT service interface. By default in this static configuration the gateway will be assigned the address 192.168.15.1 (the address following the interface address), though this is subject to change. To attach a DHCP server to the internal network, we modify the example as follows:

```
VBoxManage natnetwork add --netname natnet1 --network "192.168.15.0/24" --enable --dhcp on
```

or to add a DHCP server to the network after creation:

```
VBoxManage natnetwork modify --netname natnet1 --dhcp on
```

To disable it again, use:

```
VBoxManage natnetwork modify --netname natnet1 --dhcp off
```

DHCP server provides list of registered nameservers, but doesn't map servers from 127/8 network.

To start the NAT service, use the following command:

```
VBoxManage natnetwork start --netname natnet1
```

If the network has a DHCP server attached then it will start together with the NAT network service.

```
VBoxManage natnetwork stop --netname natnet1
```

stops the NAT network service, together with DHCP server if any.

To delete the NAT network service use:

```
VBoxManage natnetwork remove --netname natnet1
```

This command does not remove the DHCP server if one is enabled on the internal network.

Port-forwarding is supported (using the `--port-forward-4` switch for IPv4 and `--port-forward-6` for IPv6):

```
VBoxManage natnetwork modify --netname natnet1 --port-forward-4 "ssh:tcp:[]:1022:[192.168.15.5]:22"
```

This adds a port-forwarding rule from the host's TCP 1022 port to the port 22 on the guest with IP address 192.168.15.5. Host port, guest port and guest IP are mandatory. To delete the rule, use:

```
VBoxManage natnetwork modify --netname natnet1 --port-forward-4 delete ssh
```

It's possible to bind NAT service to specified interface:

```
VBoxManage setextradata global "NAT/win-nat-test-0/SourceIp4" 192.168.1.185
```

To see the list of registered NAT networks, use:

```
VBoxManage list natnetworks
```

6.5. Bridged networking

With bridged networking, VirtualBox uses a device driver on your *host* system that filters data from your physical network adapter. This driver is therefore called a "net filter" driver. This allows VirtualBox to intercept data from the physical network and inject data into it, effectively creating a new network interface in software. When a guest is using such a new software interface, it looks to the host system as though the guest were physically connected to the interface using a network cable: the host can send data to the guest through that interface and receive data from it. This means that you can set up routing or bridging between the guest and the rest of your network.

For this to work, VirtualBox needs a device driver on your host system. The way bridged networking works has been completely rewritten with VirtualBox 2.0 and 2.1, depending on the host operating system. From the user perspective, the main difference is that complex configuration is no longer necessary on any of the supported host operating systems.[31]

> **Note**
>
> Even though TAP is no longer necessary on Linux with bridged networking, you *can* still use TAP interfaces for certain advanced setups, since you can connect a VM to any host interface -- which could also be a TAP interface.

To enable bridged networking, all you need to do is to open the Settings dialog of a virtual machine, go to the "Network" page and select "Bridged network" in the drop down list for the "Attached to" field. Finally, select desired host interface from the list at the bottom of the page, which contains the physical network interfaces of your systems. On a typical MacBook, for example, this will allow you to select between "en1: AirPort" (which is the wireless interface) and "en0: Ethernet", which represents the interface with a network cable.

> **Note**
>
> Bridging to a wireless interface is done differently from bridging to a wired interface, because most wireless adapters do not support promiscuous mode. All traffic has to use the MAC address of the host's wireless adapter, and therefore VirtualBox needs to replace the source MAC address in the Ethernet header of an outgoing packet to make sure the reply will be sent to the host interface. When VirtualBox sees an incoming packet with a destination IP address that belongs to one of the virtual machine adapters it replaces the destination MAC address in the Ethernet header with the VM adapter's MAC address and passes it on. VirtualBox examines ARP and DHCP packets in order to learn the IP addresses of virtual machines.

Depending on your host operating system, the following limitations should be kept in mind:

- On **Macintosh** hosts, functionality is limited when using AirPort (the Mac's wireless networking)

for bridged networking. Currently, VirtualBox supports only IPv4 and IPv6 over AirPort. For other protocols (such as IPX), you must choose a wired interface.

- On **Linux** hosts, functionality is limited when using wireless interfaces for bridged networking. Currently, VirtualBox supports only IPv4 and IPv6 over wireless. For other protocols (such as IPX), you must choose a wired interface.

  Also, setting the MTU to less than 1500 bytes on wired interfaces provided by the sky2 driver on the Marvell Yukon II EC Ultra Ethernet NIC is known to cause packet losses under certain conditions.

  Some adapters strip VLAN tags in hardware. This does not allow to use VLAN trunking between VM and the external network with pre-2.6.27 Linux kernels nor with host operating systems other than Linux.

- On **Solaris** hosts, there is no support for using wireless interfaces. Filtering guest traffic using IPFilter is also not completely supported due to technical restrictions of the Solaris networking subsystem. These issues would be addressed in a future release of Solaris 11.

  Starting with VirtualBox 4.1, on Solaris 11 hosts (build 159 and above), it is possible to use Solaris' Crossbow Virtual Network Interfaces (VNICs) directly with VirtualBox without any additional configuration other than each VNIC must be exclusive for every guest network interface.

  Starting with VirtualBox 2.0.4 and up to VirtualBox 4.0, VNICs can be used but with the following caveats:

  - A VNIC cannot be shared between multiple guest network interfaces, i.e. each guest network interface must have its own, exclusive VNIC.
  - The VNIC and the guest network interface that uses the VNIC must be assigned identical MAC addresses.

  When using VLAN interfaces with VirtualBox, they must be named according to the PPA-hack naming scheme (e.g. "e1000g513001"), as otherwise the guest may receive packets in an unexpected format.

6.6. Internal networking

Internal Networking is similar to bridged networking in that the VM can directly communicate with the outside world. However, the "outside world" is limited to other VMs on the same host which connect to the same internal network.

Even though technically, everything that can be done using internal networking can also be done using bridged networking, there are security advantages with internal networking. In bridged networking mode, all traffic goes through a physical interface of the host system. It is therefore possible to attach a packet sniffer (such as Wireshark) to the host interface and log all traffic that goes over it. If, for any reason, you prefer two or more VMs on the same machine to communicate privately, hiding their data from both the host system and the user, bridged networking therefore is not an option.

Internal networks are created automatically as needed, i.e. there is no central configuration. Every internal network is identified simply by its name. Once there is more than one active virtual network card with the same internal network ID, the VirtualBox support driver will automatically "wire" the cards and act as a network switch. The VirtualBox support driver implements a complete Ethernet switch and supports both broadcast/multicast frames and promiscuous mode.

In order to attach a VM's network card to an internal network, set its networking mode to "internal networking". There are two ways to accomplish this:

- You can use a VM's "Settings" dialog in the VirtualBox graphical user interface. In the "Networking" category of the settings dialog, select "Internal Networking" from the drop-down list of networking modes. Now select the name of an existing internal network from the drop-down below or enter a new name into the entry field.

- You can use

  ```
  VBoxManage modifyvm "VM name" --nic<x> intnet
  ```

  Optionally, you can specify a network name with the command

  ```
  VBoxManage modifyvm "VM name" --intnet<x> "network name"
  ```

  If you do not specify a network name, the network card will be attached to the network `intnet` by default.

Unless you configure the (virtual) network cards in the guest operating systems that are participating in the internal network to use static IP addresses, you may want to use the DHCP server that is built into VirtualBox to manage IP addresses for the internal network. Please see Section 8.38, "VBoxManage dhcpserver" for details.

As a security measure, by default, the Linux implementation of internal networking only allows VMs running under the same user ID to establish an internal network. However, it is possible to create a shared internal networking interface, accessible by users with different UUIds.

6.7. Host-only networking

Host-only networking is another networking mode that was added with version 2.2 of VirtualBox. It can be thought of as a hybrid between the bridged and internal networking modes: as with bridged networking, the virtual machines can talk to each other and the host as if they were connected through a physical Ethernet switch. Similarly, as with internal networking however, a physical networking interface need not be present, and the virtual machines cannot talk to the world outside the host since they are not connected to a physical networking interface.

Instead, when host-only networking is used, VirtualBox creates a new software interface on the host which then appears next to your existing network interfaces. In other words, whereas with bridged networking an existing physical interface is used to attach virtual machines to, with host-only networking a new "loopback" interface is created on the host. And whereas with internal networking, the traffic between the virtual machines cannot be seen, the traffic on the "loopback" interface on the host can be intercepted.

Host-only networking is particularly useful for preconfigured virtual appliances, where multiple virtual machines are shipped together and designed to cooperate. For example, one virtual machine may contain a web server and a second one a database, and since they are intended to talk to each other, the appliance can instruct VirtualBox to set up a host-only network for the two. A second (bridged) network would then connect the web server to the outside world to serve data to, but the outside world cannot connect to the database.

To change a virtual machine's virtual network interface to "host only" mode:

- either go to the "Network" page in the virtual machine's settings notebook in the graphical user interface and select "Host-only networking", or
- on the command line, type `VBoxManage modifyvm "VM name" --nic<x> hostonly`; see Section 8.8, "VBoxManage modifyvm" for details.

Before you can attach a VM to a host-only network you have to create at least one host-only interface, either from the GUI: "File" → "Preferences" → "Network" → "Host-only network" → "(+)Add host-only network", or via command line with

```
VBoxManage hostonlyif create
```

see Section 8.37, "VBoxManage hostonlyif" for details.

For host-only networking, like with internal networking, you may find the DHCP server useful that is built into VirtualBox. This can be enabled to then manage the IP addresses in the host-only network since otherwise you would need to configure all IP addresses statically.

- In the VirtualBox graphical user interface, you can configure all these items in the global settings via "File" → "Preferences" → "Network", which lists all host-only networks which are presently in

use. Click on the network name and then on the "Edit" button to the right, and you can modify the adapter and DHCP settings.

- Alternatively, you can use `VBoxManage dhcpserver` on the command line; please see Section 8.38, "VBoxManage dhcpserver" for details.

> **Note**
> On Linux and Mac OS X hosts the number of host-only interfaces is limited to 128. There is no such limit for Solaris and Windows hosts.

6.8. UDP Tunnel networking

This networking mode allows to interconnect virtual machines running on different hosts.

Technically this is done by encapsulating Ethernet frames sent or received by the guest network card into UDP/IP datagrams, and sending them over any network available to the host.

UDP Tunnel mode has three parameters:

Source UDP port

The port on which the host listens. Datagrams arriving on this port from any source address will be forwarded to the receiving part of the guest network card.

Destination address

IP address of the target host of the transmitted data.

Destination UDP port

Port number to which the transmitted data is sent.

When interconnecting two virtual machines on two different hosts, their IP addresses must be swapped. On single host, source and destination UDP ports must be swapped.

In the following example host 1 uses the IP address 10.0.0.1 and host 2 uses IP address 10.0.0.2. Configuration via command-line:

```
VBoxManage modifyvm "VM 01 on host 1" --nic<x> generic
VBoxManage modifyvm "VM 01 on host 1" --nicgenericdrv<x> UDPTunnel
VBoxManage modifyvm "VM 01 on host 1" --nicproperty<x> dest=10.0.0.2
VBoxManage modifyvm "VM 01 on host 1" --nicproperty<x> sport=10001
VBoxManage modifyvm "VM 01 on host 1" --nicproperty<x> dport=10002
```

and

```
VBoxManage modifyvm "VM 02 on host 2" --nic<y> generic
VBoxManage modifyvm "VM 02 on host 2" --nicgenericdrv<y> UDPTunnel
VBoxManage modifyvm "VM 02 on host 2" --nicproperty<y> dest=10.0.0.1
VBoxManage modifyvm "VM 02 on host 2" --nicproperty<y> sport=10002
VBoxManage modifyvm "VM 02 on host 2" --nicproperty<y> dport=10001
```

Of course, you can always interconnect two virtual machines on the same host, by setting the destination address parameter to 127.0.0.1 on both. It will act similarly to "Internal network" in this case, however the host can see the network traffic which it could not in the normal Internal network case.

> **Note**
> On Unix-based hosts (e.g. Linux, Solaris, Mac OS X) it is not possible to bind to ports below 1024 from applications that are not run by `root`. As a result, if you try to configure such a source UDP port, the VM will refuse to start.

6.9. VDE networking

Virtual Distributed Ethernet (VDE[32]) is a flexible, virtual network infrastructure system, spanning across multiple hosts in a secure way. It allows for L2/L3 switching, including spanning-tree protocol, VLANs, and WAN emulation. It is an optional part of VirtualBox which is only included in the source

code.

The basic building blocks of the infrastructure are VDE switches, VDE plugs and VDE wires which interconnect the switches.

The VirtualBox VDE driver has one parameter:

VDE network

The name of the VDE network switch socket to which the VM will be connected.

The following basic example shows how to connect a virtual machine to a VDE switch:

1. Create a VDE switch:

   ```
   vde_switch -s /tmp/switch1
   ```

2. Configuration via command-line:

   ```
   VBoxManage modifyvm "VM name" --nic<x> generic

   VBoxManage modifyvm "VM name" --nicgenericdrv<x> VDE
   ```

   To connect to automatically allocated switch port, use:

   ```
   VBoxManage modifyvm "VM name" --nicproperty<x> network=/tmp/switch1
   ```

   To connect to specific switch port <n>, use:

   ```
   VBoxManage modifyvm "VM name" --nicproperty<x> network=/tmp/switch1[<n>]
   ```

   The latter option can be useful for VLANs.

3. Optionally map between VDE switch port and VLAN: (from switch CLI)

   ```
   vde$ vlan/create <VLAN>

   vde$ port/setvlan <port> <VLAN>
   ```

VDE is available on Linux and FreeBSD hosts only. It is only available if the VDE software and the VDE plugin library from the VirtualSquare project are installed on the host system[33]. For more information on setting up VDE networks, please see the documentation accompanying the software.[34]

6.10. Limiting bandwidth for network I/O

Starting with version 4.2, VirtualBox allows for limiting the maximum bandwidth used for network transmission. Several network adapters of one VM may share limits through bandwidth groups. It is possible to have more than one such limit.

> **Note**
> VirtualBox shapes VM traffic only in the transmit direction, delaying the packets being sent by virtual machines. It does not limit the traffic being received by virtual machines.

Limits are configured through `VBoxManage`. The example below creates a bandwidth group named "Limit", sets the limit to 20 Mbit/s and assigns the group to the first and second adapters of the VM:

```
VBoxManage bandwidthctl "VM name" add Limit --type network --limit 20m
VBoxManage modifyvm "VM name" --nicbandwidthgroup1 Limit
VBoxManage modifyvm "VM name" --nicbandwidthgroup2 Limit
```

All adapters in a group share the bandwidth limit, meaning that in the example above the bandwidth of both adapters combined can never exceed 20 Mbit/s. However, if one adapter doesn't require bandwidth the other can use the remaining bandwidth of its group.

The limits for each group can be changed while the VM is running, with changes being picked up immediately. The example below changes the limit for the group created in the example above to 100 Kbit/s:

```
VBoxManage bandwidthctl "VM name" set Limit --limit 100k
```

To completely disable shaping for the first adapter of VM use the following command:

```
VBoxManage modifyvm "VM name" --nicbandwidthgroup1 none
```

It is also possible to disable shaping for all adapters assigned to a bandwidth group while VM is running, by specifying the zero limit for the group. For example, for the bandwidth group named "Limit" use:

```
VBoxManage bandwidthctl "VM name" set Limit --limit 0
```

6.11. Improving network performance

VirtualBox provides a variety of virtual network adapters that can be "attached" to the host's network in a number of ways. Depending on which types of adapters and attachments are used the network performance will be different. Performance-wise the *virtio* network adapter is preferable over *Intel PRO/1000* emulated adapters, which are preferred over *PCNet* family of adapters. Both *virtio* and *Intel PRO/1000* adapters enjoy the benefit of segmentation and checksum offloading. Segmentation offloading is essential for high performance as it allows for less context switches, dramatically increasing the sizes of packets that cross VM/host boundary.

> **Note**
>
> Neither *virtio* nor *Intel PRO/1000* drivers for Windows XP support segmentation offloading. Therefore Windows XP guests never reach the same transmission rates as other guest types. Refer to MS Knowledge base article 842264 for additional information.

Three attachment types: *internal*, *bridged* and *host-only*, have nearly identical performance, the *internal* type being a little bit faster and using less CPU cycles as the packets never reach the host's network stack. The *NAT* attachment is the slowest (and safest) of all attachment types as it provides network address translation. The generic driver attachment is special and cannot be considered as an alternative to other attachment types.

The number of CPUs assigned to VM does not improve network performance and in some cases may hurt it due to increased concurrency in the guest.

Here is the short summary of things to check in order to improve network performance:

1. Whenever possible use *virtio* network adapter, otherwise use one of *Intel PRO/1000* adapters;
2. Use *bridged* attachment instead of *NAT*;
3. Make sure segmentation offloading is enabled in the guest OS. Usually it will be enabled by default. You can check and modify offloading settings using `ethtool` command in Linux guests.
4. Perform a full, detailed analysis of network traffic on the VM's network adaptor using a 3rd party tool such as Wireshark. To do this, a promiscuous mode policy needs to be used on the VM's network adaptor. Use of this mode is only possible on networks: NAT Network, Bridged Adapter, Internal Network and Host-only Adapter.

   To setup a promiscuous mode policy, either select from the drop down list located in the Network Settings dialog for the network adaptor or use the command line tool `VBoxManage`; for details, refer to Section 8.8, “VBoxManage modifyvm”.

   Promiscuous mode policies are:

   a. `deny` (default setting) which hides any traffic not intended for this VM's network adaptor.

   b. `allow-vms` which hides all host traffic from this VM's network adaptor, but allows it to see traffic from/to other VMs.

   c. `allow-all` which removes all restrictions - this VM's network adaptor sees all traffic.

[30] http://www.linux-kvm.org/page/WindowsGuestDrivers.

[31] For Mac OS X and Solaris hosts, net filter drivers were already added in VirtualBox 2.0 (as initial support for Host Interface Networking on these platforms). With VirtualBox 2.1, net filter drivers were also added for the Windows and Linux hosts, replacing the mechanisms previously present in VirtualBox for those platforms; especially on Linux, the earlier method required creating TAP interfaces and bridges, which was complex and varied from one distribution to the next. None of this is necessary anymore. Bridged network was formerly called "Host Interface Networking" and has been renamed with version 2.2 without any change in functionality.

[32] VDE is a project developed by Renzo Davoli, Associate Professor at the University of Bologna, Italy.

[33] For Linux hosts, the shared library libvdeplug.so must be available in the search path for shared libraries

[34] http://wiki.virtualsquare.org/wiki/index.php/VDE_Basic_Networking.