Virtual Networking

How the virtual networks used by guests work

Networking using libvirt is generally fairly simple, and in this section you'll learn the concepts you need to be effective with it.

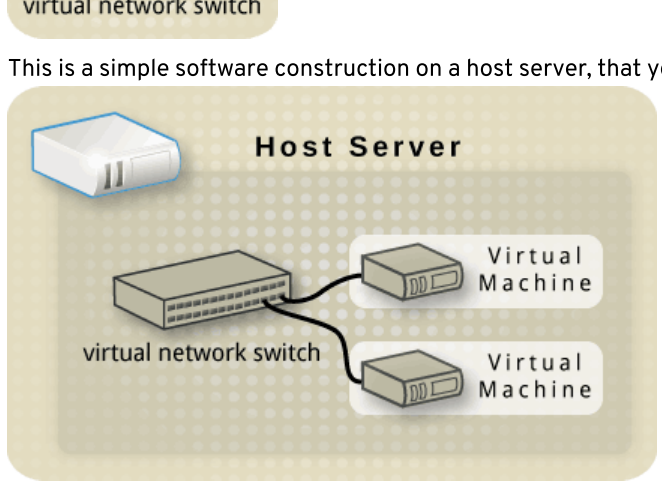
Also please bear in mind that advanced users can change important parts of how the network layer operates, far past the concepts outlined here. This section will be enough to get you up and running though. :)

Virtual network switches

Firstly, libvirt uses the concept of a *virtual network switch*.

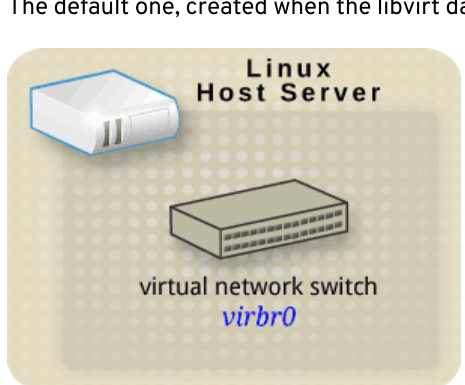
virtual network switch

This is a simple software construction on a host server, that your virtual machines "plug in" to, and direct their traffic through.



On a Linux host server, the virtual network switch shows up as a network interface.

The default one, created when the libvirt daemon is first installed and started, shows up as **virbr0**.



If you're familiar with the **ifconfig** command, you can use that to show it:

```
$ ifconfig virbr0
virbr0    Link encap:Ethernet  HWaddr 1A:D4:92:CF:FD:17
          inet addr:192.168.122.1  Bcast:192.168.122.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:11 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:3097 (3.0 KiB)
```

If you're more familiar with the **ip** command instead, this is how it looks:

```
$ ip addr show virbr0
3: virbr0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN
    link/ether 1a:d4:92:cf:fd:17 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
```

Showing it in context, with the other network interfaces on the host:

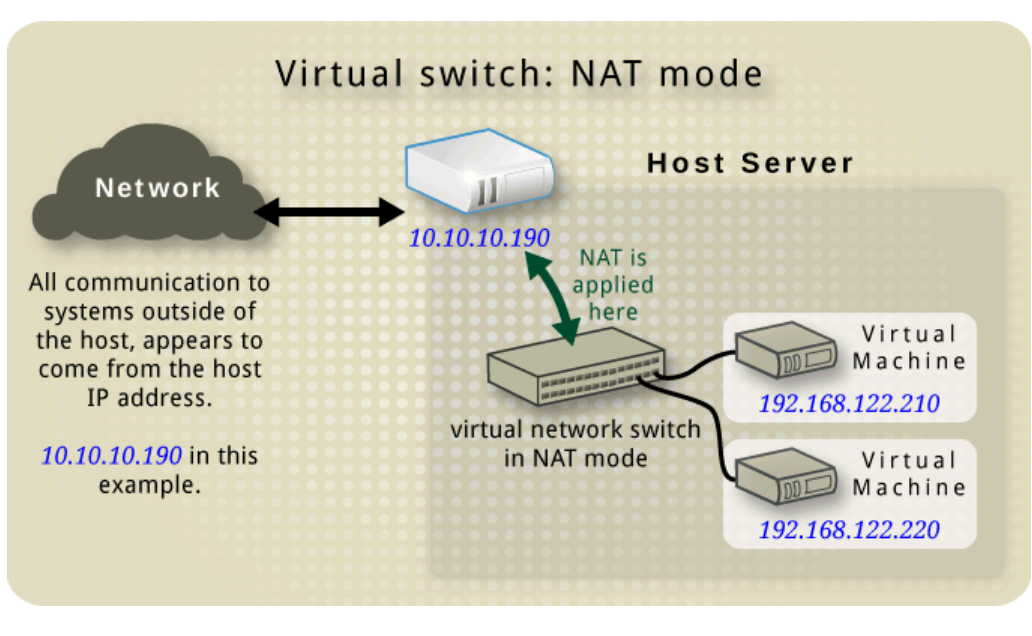
```
$ ifconfig -a
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:13 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:892 (892.0 b)  TX bytes:892 (892.0 b)

eth0      Link encap:Ethernet  HWaddr 00:1B:21:43:33:30
          inet addr:10.10.10.190  Bcast:10.10.255.255  Mask:255.255.0.0
          inet6 addr: fe80::21b:21ff:fe43:3330/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1942 errors:0 dropped:0 overruns:0 frame:0
          TX packets:829 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:985906 (962.7 KiB)  TX bytes:142753 (139.4 KiB)
          Memory:fbea0000-fbec0000

virbr0    Link encap:Ethernet  HWaddr 1A:D4:92:CF:FD:17
          inet addr:192.168.122.1  Bcast:192.168.122.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:11 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:3097 (3.0 KiB)
```

```
$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
    link/ether 00:1b:21:43:33:30 brd ff:ff:ff:ff:ff:ff
    inet 10.10.10.190/16 brd 10.10.255.255 scope global eth0
    inet6 fe80::21b:21ff:fe43:3330/64 scope link
       valid_lft forever preferred_lft forever
3: virbr0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN
    link/ether 1a:d4:92:cf:fd:17 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
```

Network Address Translation (NAT)

By default, a virtual network switch operates in NAT mode (using IP masquerading rather than SNAT or DNAT).

This means any guests connected through it, use the host IP address for communication to the outside world. Computers external to the host can't initiate communications to the guests inside, when the virtual network switch is operating in NAT mode.

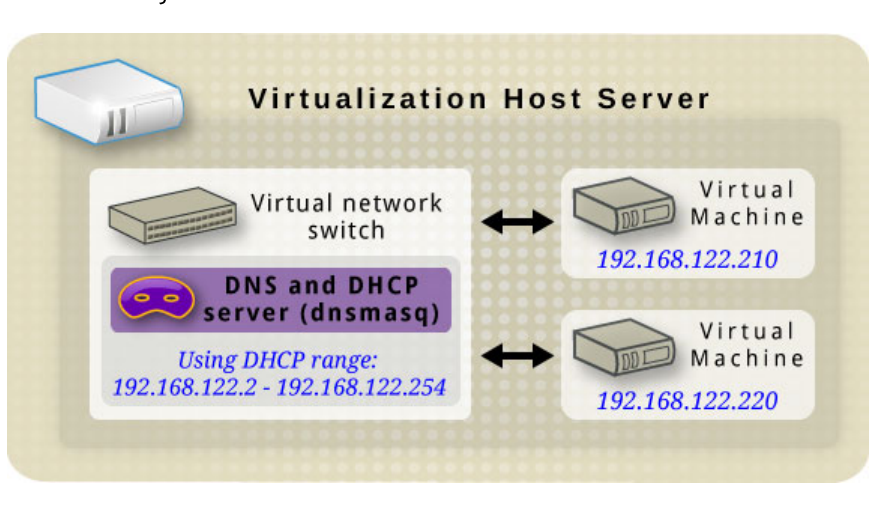


WARNING - The NAT is set up using *iptables* rules. Be careful if you change these while the virtual switch is running. If something goes wrong with the iptables rules, your virtual machines may stop communicating properly.

DNS & DHCP

Each virtual network switch can be given a range of IP addresses, to be provided to guests through DHCP.

Libvirt uses a program, **dnsmasq**, for this. An instance of dnsmasq is automatically configured and started by libvirt for each virtual network switch needing it.



Other virtual network switch routing types

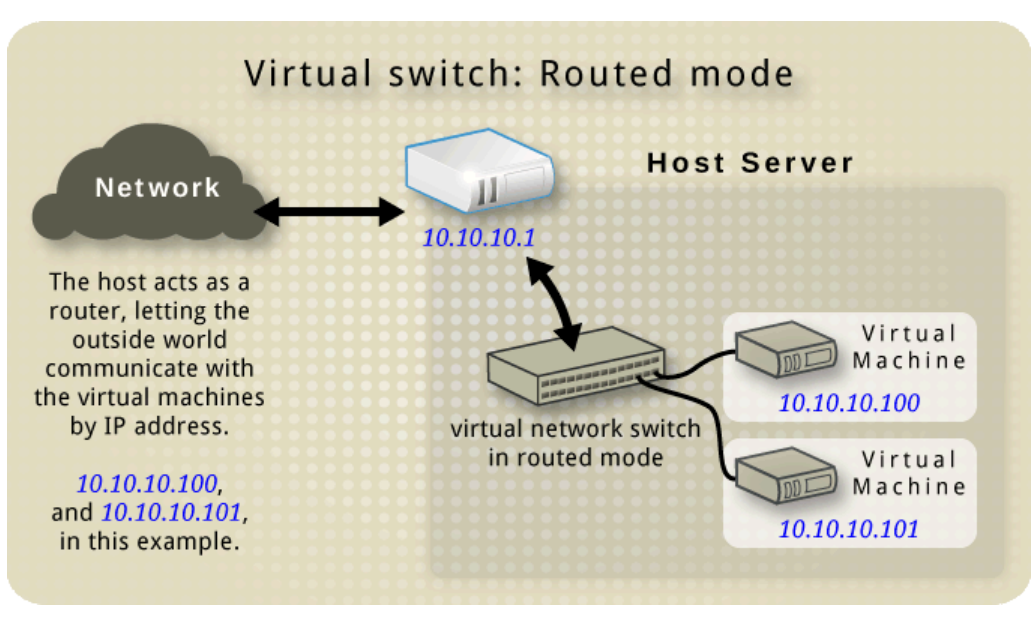
Virtual network switches can operate in two other modes, instead of NAT:

Routed mode

With **routed** mode, the virtual switch is connected to the physical host LAN, passing guest network traffic back and forth without using NAT.

The virtual switch sees the IP addresses in each packet, using that information when deciding what to do.

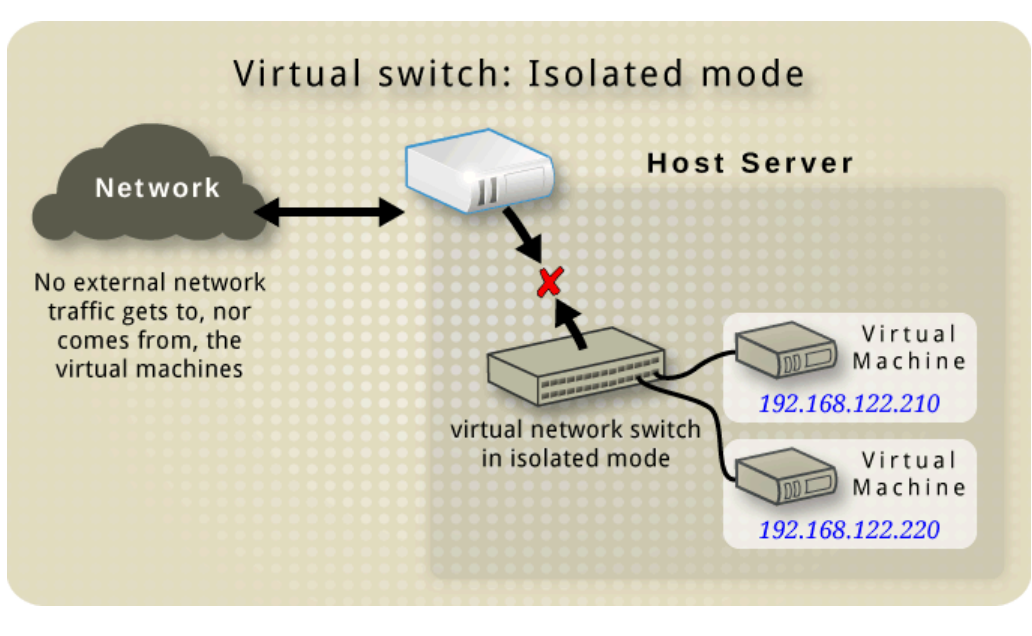
In this mode all virtual machines are in a subnet routed through the virtual switch. This on its own is not sufficient. because no other hosts on the physical network know this subnet exists or how to reach it. It is thus necessary to configure routers in the physical network (e.g. using a static route).



If you are familiar with the ISO 7 layer network model, this mode operates on layer 3, the Network layer.

Isolated mode

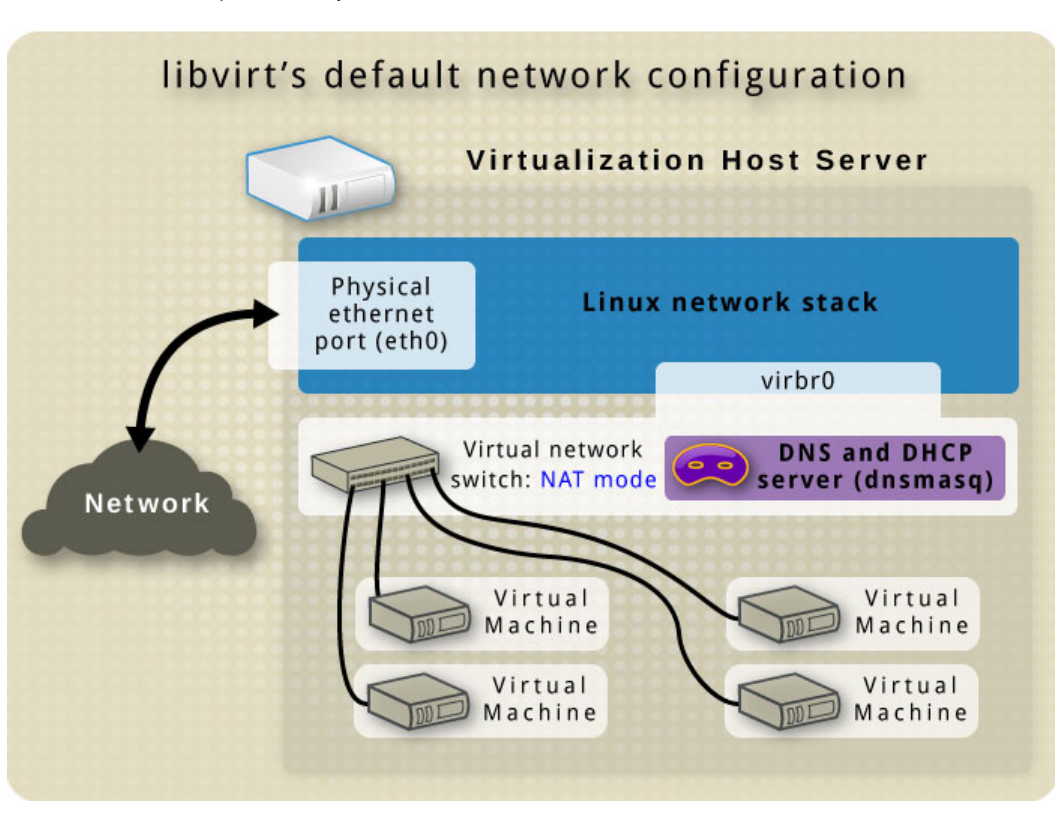
In this mode, guests connected to the virtual switch can communicate with each other, and with the host. However, their traffic will not pass outside of the host, nor can they receive traffic from outside the host.



The use of dnsmasq in this mode is possible and in fact needed since it is used to answer DHCP requests. However, even if this network is isolated from any physical network, DNS names are still resolved. Therefore one can get into the situation where DNS is resolved but guests are unable to ping.

The default configuration

When the libvirt daemon is first installed on a server, it comes with an initial *virtual network switch* configuration. This virtual switch is in NAT mode, and is used by installed guests for communication. (ie to the outside network)



The libvirt daemon puts this configuration into effect when it starts up, so if you have the libvirt daemon set to start automatically on each boot it should always be present.

If the libvirt daemon is only started manually instead, this is when the default virtual network switch will become available on the host.

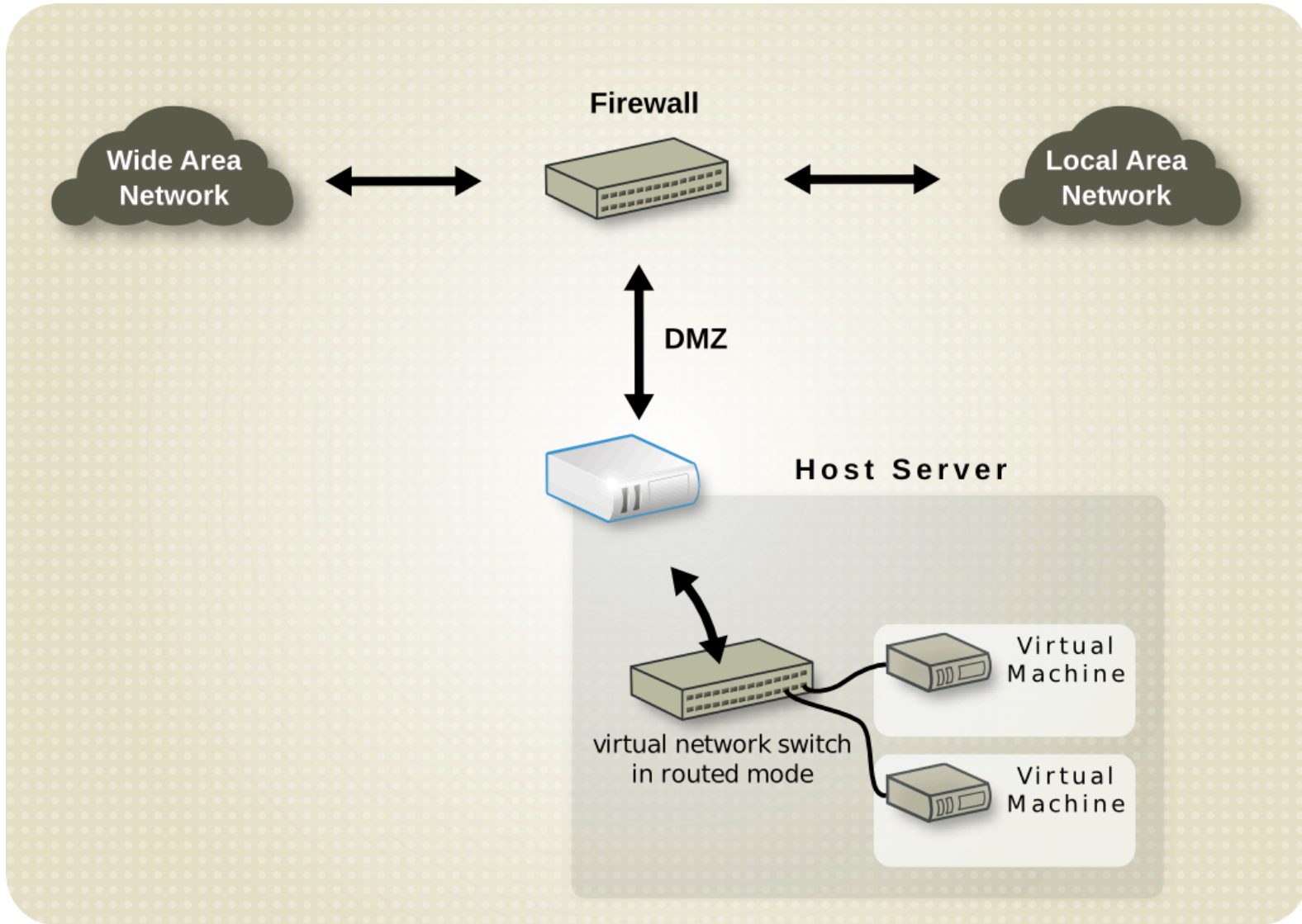
Restricting virtual network traffic to a specific interface

As stated above, a virtual network can be connected to a physical netwok. Its traffic might be restricted to use a specific interface, e.g. on a system with eth0/1/2 one can limit the virtual network to use eth0 only. However, this only makes sense in routed and nat modes. The restriction can be defined in XML (dev="" attribute) or in virt-manager when creating a new virtual network.

Examples of common scenarios

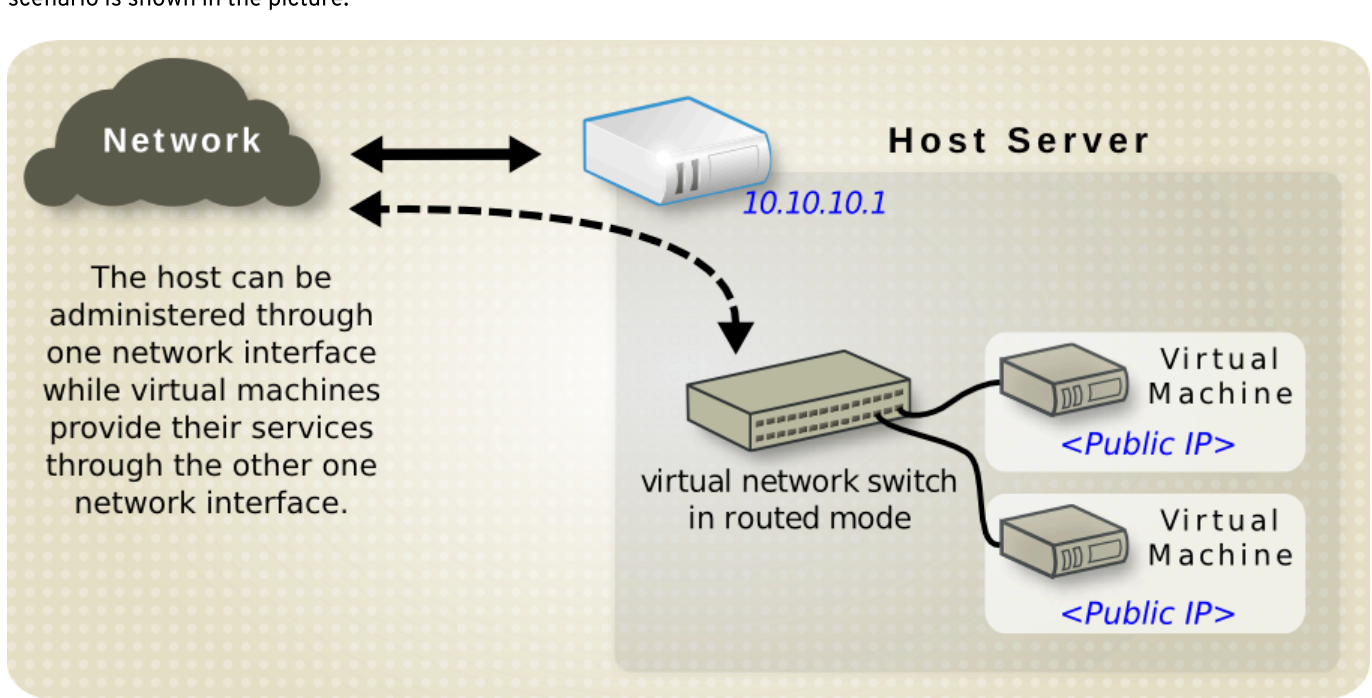
Routed mode example

Suppose, there is a network where a node or bunch of nodes need to be in special subnetwork for let's say security reasons. This is called DMZ - Demilitarized Zone. How this networks look like is shown in the picture:

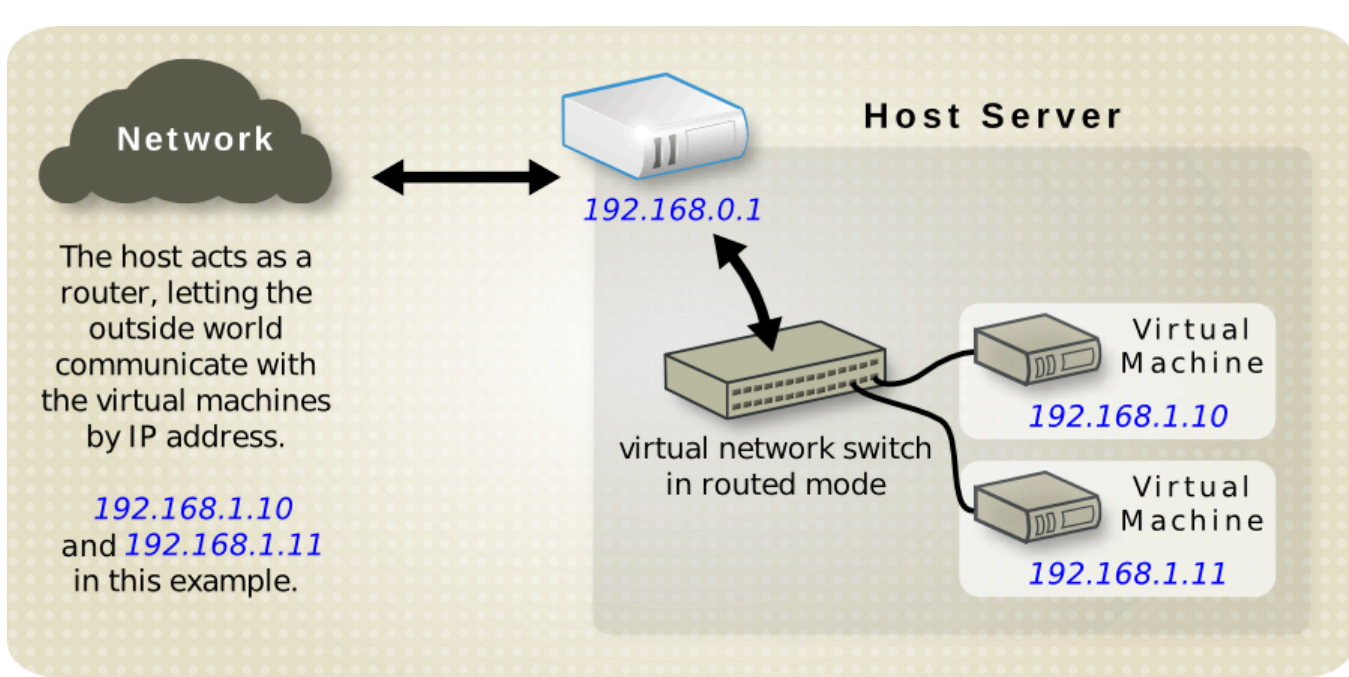


Hosts in DMZ provide services both to LAN hosts and WAN. Therefore, they need to be accessible by other computers on the intranet and also by computers in the internet. Since it wouldn't be secure to have them on LAN (attacker could access LAN after successful attack), they are in special subnet. In addition, it is obvious they can't be in NAT or isolated mode.

Other scenario where routed mode is suitable is this. Consider virtual server hosting company. Each host have two physical network connections. One is for general management, accounting etc. The other is for the virtual machines to use. Each virtual machine has its own public IP address. Hosts however use private IPs, because virtual machine management is allowed to company administrators only. Whole scenario is shown in the picture:



Again, it is obvious virtual network switch can't be in NAT nor isolated mode. Special case of this is another example. Host has public IP and virtual machines have static public IPs. But one can't use bridged networking, since provider accept only packets from the MAC address of the host. Whole situation is shown in the picture:



NAT mode example

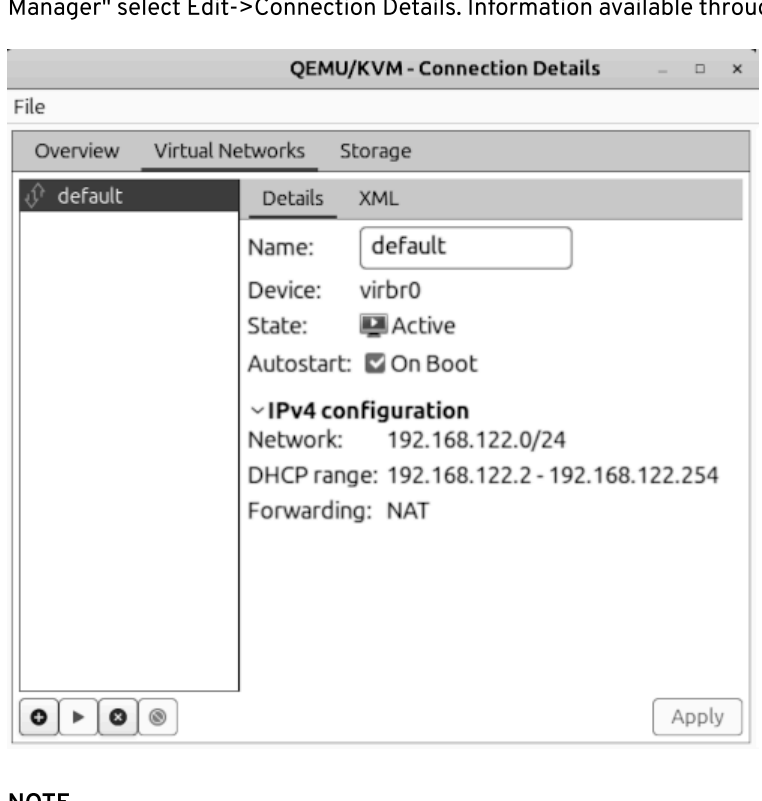
This is the default mode and requires no additional configuration at all. It is used when anywhere there is no need for 'being seen on the network'. For instance, a web developer who optimizes web pages for different operating systems and web browsers. Or any other developer, who need to try things out in different configurations, environments, or operating systems.

Isolated mode example

An example where this mode would be useful is running simulations in the security field, where the spread of malware is being watched. Virtual machines can communicate with each other, but since they are cut off from the physical network, no real damage can be done.

The Virtual Machine Manager (virt-manager)

In virt-manager is possibility to view and manage virtual networks. To open "QEMU/KVM - Connection Details" window in "Virtual Machine Manager" select Edit->Connection Details. Information window through virt-manager can be seen in this image:



NOTE

- - Need to include which versions of virt-manager have this (ie from 0.x.y onwards)
  - Also need to list which drivers support this. ie qemu+ssh:// might, whereas qemu:// might not (that's an example only, but recent quick testing showed up some unexpected things here)

Creating a virtual network

Creating virtual networks is easy when using the Virtual Machine Manager GUIT.

The following pages take you through the steps for each of the main network types:

- Creating a NAT Virtual Network
- Creating a Routed Virtual Network
- Creating an Isolated Virtual Network

Starting a virtual network

In virt-manager by clicking Start Network, or in virsh *net-start*. This command takes one mandatory argument, the network name. When starting a virtual network, libvirt will automatically set iptables and dnsmasq. However, transient networks are created and started at once.

Stopping a virtual network

Stopping virtual network can be done by clicking the appropriate button in Virtual Manager or by *net-destroy*. If it is a transient network being stopped, it is also removed.

Removing a virtual network

Again, removing a virtual network is possible in Virtual Manager or in virsh by *net-undefine*. Please keep in mind, only inactive networks can be removed.

Changing a virtual network

Making changes is only available via the virsh console tool. 'The 'net-edit *command allows the user to edit the XML configuration of a virtual network.*

- Stats collection in virt-manager
  - Need to include which versions of virt-manager have this (ie from 0.x.y onwards)
  - Implications of stats collection (performance impact?)
  - How to enable/disable collection of stats in virt-manager
  - Display of stats

Basic command line usage for virtual networks

Introduces the basic virsh net-* commands for virtual network management. Here, the *<network-identifier>* stands for either network name or network UUID.

net-list - List the virtual networks libvirt is aware of, along with some basic status and autostart flag information. Used without parameters it shows active virtual networks only.

```
Usage: net-list [--all] [--inactive].
```

net-start - Starts an inactive, previously defined virtual network.

```
Usage: net-start [--network] <network-identifier>
```

net-destroy - Stops an active network and deallocates all resources used by it, e.g. stopping appropiate dnsmasq process, releasing the bridge. The virtual network being stopped can be persistent or transient.

```
Usage: net-destroy [--network] <network-identifier>
```

net-undefine - Removes an inactive presistent virtual network from the libvirt configuration.

```
Usage: net-undefine [--network] <network-identifier>
```

net-autostart - Marks or unmarks automatic startup of a persistent virtual network. Networks with the autostart flag enabled are started whenever libvirt daemon starts. To disable autostart use the *--disable* switch.

```
Usage: net-autostart [--network] <network-identifier> [--disable]
```

net-name - Returns the network name corresponding to the given UUID.

```
Usage: net-name [--network] <network-uuid>
```

net-uuid - Returns the UUID corresponding to the given network-name.

```
Usage: net-uuid [--network] <network-name>
```

net-dumpxml - Outputs the XML configuration for a virtual network.

```
Usage: net-dumpxml [--network] <network-identifier>
```

Advanced

Further dnsmasq info

dnsmasq

- dnsmasq does more than just plain DNS forwarding. It also includes the entries from /etc/hosts (on the virtualization host) as replies to DNS queries. This is a useful way to easily create local DNS entries, or override upstream DNS ones.

Persistent vs non-persistent virtual networks

Libvirt allows a virtual network to be persistent or transient. A transient network, once created (using *net-create*) lasts until destroyed or the libvirt daemon restarts.

The alternative is a persistent network (*net-define*) which lasts until explicitly destroyed. Persistent networks, in addition, can be autostarted. This means when the libvirt daemon is starting up it will also run the virtual network.

XML format

The root element required for all virtual networks is named 'network' and has no attributes. The first elements provide basic metadata about the virtual network.

```
<network>
  <name>default</name>
  <uuid>f01bd721-af12-4d20-9cf2-390c7375b17c</uuid>
  ...
```

- name - The content provides the name for the virtual network. The name should contain only alpha-numeric characters and is required to be unique within a single host, because it is used for the filename for storing the persistent configuration file.
- uuid - The content provides a globally unique identifier for the virtual network. The format must be RFC 4122 compliant. If not specified when defining or creating a new network, a random UUID is generated.

The next two elements defines the virtual network's connectivity to the physical network (if any).

```
...
<forward dev='eth0' mode='nat'/>
<bridge name='virbr0' stp='on' delay='0' />
...
```

- forward - This element is optional. When not defined, the virtual network will work in isolated mode. However, inclusion of this element indicates that the virtual network is to be connected to the physical network. The element can have two attributes, 'mode' and 'dev'. The first one specifies the mode in which will the virtual bridge operates. Allowed values are 'nat' and 'route'. The second attribute is used whenever one wants to restrict forwarding to the named device only. If no attributes are set, NAT forwarding will be used for connectivity. Firewall rules will allow forwarding to any other network device.
- bridge - The 'name' attribute of this element defines the name of a bridge device which will be used to construct the virtual network. The next two attributes specify whether the Spanning Tree Protocol is used on the defined bridge to prevent bridge loops and forward delay.

The final set of elements define the IPv4 address range available and optionally enable DHCP.

```
...
<ip address="192.168.122.1" netmask="255.255.255.0">
  <dhcp>
    <range start="192.168.122.100" end="192.168.122.254" />
    <host mac="00:16:3e:e2:ed" name="foo.example.com" ip="192.168.122.10" />
  </dhcp>
</ip>
...
```

- ip - The attributes of this element define an IPv4 address for the bridge and the subnet.
- dhcp - This optional element enables DHCP services on the virtual network. It can have one or more 'range' child elements.
- range - Two attributes specify the boundaries of a pool of IPv4 addresses to be provided to DHCP clients. The whole range must lie within the scope of the network defined on the parent 'ip' element.
- host - This element is optional and may occur zero or more times within the 'dhcp' element. It is used for static DHCP, when one wants to always assign the same IP address and name to some interface.

Location of XML files on the host

XML definition files of persistent virtual networks are stored in the */etc/libvirt/<hypervisor>/networks/* directory. In addition, if the network is marked as autostart, the symbolic link to its XML file is created under the *autostart/* subdirectory.

virsh XML commands

net-edit - Edits the XML configuration of a virtual network. net-edit launches the editor defined in *$EDITOR* environment variable passing it a temporary copy of the XML configuration file for the virtual network. When the user finishes editing, net-edit checks the temporary file for changes and errors and redefines the virtual network.

```
Usage: net-edit [--network] <network-identifier>
```

net-create - Creates a running transient virtual network. Command takes one argument, the full path to an XML file containing network settings.

```
Usage: net-create [--file] <file-name>
```

net-define - Creates a persistent virtual network, without starting it, from the given XML file. To start the network use net-autostart and/or net-start.

```
Usage: net-define [--file] <file-name>
```

brctl commands

The bridge control commands (brctl) should definitely be covered, as they're used to understand how the network topology is put together.

Also, some people will want to know how to set up their own bridges manually, rather than have libvirt do it.

This should probably go into its own sub-section, as there's a decent amount of topic in it to cover properly.

Another idea might be to read some manual page about brctl.

NOTE - When covering the *brctl* addbr command, specifically point out that a random MAC address will be displayed for it if ifconfig is used, even though the bridge interface doesn't actually have a MAC address. It is important, as it's misleading and can confuse a person that is wondering "how/why is ARP propagating through this, when it has a MAC address? ARP isn't supposed to propagate..." (this caught me out). When the bridge has its first network interface assigned to it, it will then use that interface's MAC address from then on. (It only uses the MAC of the first interface, not of any further interfaces plugged in).