QEMU/Networking

QEMU supports networking by emulating some popular network cards (NICs), and establishing virtual LANs (VLAN). There are four ways how QEMU guests can be connected then: user mode, socket redirection, Tap and VDE networking.

User mode networking

If no network options are specified, QEMU will default to emulating a single Intel e1000 PCI card with a user-mode network stack that bridges to the host's network. The following two command lines are equivalent:

```
qemu -m 256 -hda disk.img &
qemu -m 256 -hda disk.img -net nic -net user &
```

The guest OS will see an E1000 NIC with a virtual DHCP server on 10.0.2.2 and will be allocated an address starting from 10.0.2.15. A virtual DNS server will be accessible on 10.0.2.3, and a virtual SAMBA file server (if present) will be accessible on 10.0.2.4 allowing you to access files on the host via SAMBA file shares.

User mode networking is great for allowing access to network resources, including the Internet. By default, however, it acts as a firewall and does not permit any incoming traffic. It also doesn't support protocols other than TCP and UDP - so, for example, ping and other ICMP utilities won't work.

Redirecting ports

To allow network connections to the guest OS under user mode networking, you can redirect a port on the host OS to a port on the guest OS. This is useful for supporting file sharing, web servers and SSH servers from the guest OS.

Here is how to set up QEMU with a Windows XP guest sharing files and web pages under user mode networking. TCP port 5555 on the host is redirected to the guest's port 80 (the web server) and TCP port 5556 on the host is redirected to the guest's port 445 (Windows networking):

```
qemu -m 256 -hda disk.img -redir tcp:5555::80 -redir tcp:5556::445 &
...
mkdir -p /mnt/qemu
mount -t cifs //localhost/someshare /mnt/qemu -o user=test,pass=test,dom=workgroup,port=5556
firefox http://localhost:5555/
```

NB: When sharing folders from guest to host via Windows networking, you must specify a password for the user that mount will use to login; if you try to use no password, mount will fail with an I/O error.

TAP interfaces

QEMU can use TAP interfaces to provide full networking capability for the guest OS. This can be useful when the guest OS is running several network services and must be connected to via standard ports; where protocols other than TCP and UDP are required; and where multiple instances of QEMU need to connect to each other (although this can also be achieved in user mode networking via port redirects, or via sockets).

Setting up a TAP interface is a bit more complicated than user mode networking. It requires installing virtual private networking (VPN) on the host OS, and then establishing a bridge between the host's networking and the virtual network.

Here's how to do it on Fedora 8 with static IP assignment. The procedure should be very similar on other Linux distros, and probably not too different on other *nix systems.

TAP/TUN device

According to tuntap.txt (http://www.kernel.org/doc/Documentation/networking/tuntap.txt) , we create the TAP/TUN device first:

```
  $ sudo mkdir /dev/net
  $ sudo mknod /dev/net/tun c 10 200
  $ sudo /sbin/modprobe tun
```

qemu-ifup

First, set up a script to create the bridge and bring up the TAP interface. We'll call this script `/etc/qemu-ifup`.

```
#!/bin/sh
#
# script to bring up the tun device in QEMU in bridged mode
# first parameter is name of tap device (e.g. tap0)
#
# some constants specific to the local host - change to suit your host
#
ETH0IP=192.168.0.3
GATEWAY=192.168.0.1
BROADCAST=192.168.0.255
#
# First take eth0 down, then bring it up with IP 0.0.0.0
#
/sbin/ifdown eth0
/sbin/ifconfig eth0 0.0.0.0 promisc up
#
# Bring up the tap device (name specified as first argument, by QEMU)
#
/usr/sbin/openvpn --mktun --dev $1 --user `id -un`
/sbin/ifconfig $1 0.0.0.0 promisc up
#
# create the bridge between eth0 and the tap device
#
/usr/sbin/brctl addbr br0
/usr/sbin/brctl addif br0 eth0
/usr/sbin/brctl addif br0 $1
#
# only a single bridge so loops are not possible, turn off spanning tree protocol
#
/usr/sbin/brctl stp br0 off
#
# Bring up the bridge with ETH0IP and add the default route
#
/sbin/ifconfig br0 $ETH0IP netmask 255.255.255.0 broadcast $BROADCAST
/sbin/route add default gw $GATEWAY
#
# stop firewall - comment this out if you don't use Firestarter
#
/sbin/service firestarter stop
```

qemu-ifdown

You will also need a script to reset your networking after QEMU exits. To be consistent, we'll call it `/etc/qemu-ifdown`.

```
#!/bin/sh
#
# Script to bring down and delete bridge br0 when QEMU exits
#
# Bring down eth0 and br0
#
/sbin/ifdown eth0
/sbin/ifdown br0
/sbin/ifconfig br0 down
#
# Delete the bridge
#
/usr/sbin/brctl delbr br0
#
# bring up eth0 in "normal" mode
#
/sbin/ifconfig eth0 -promisc
/sbin/ifup eth0
#
# delete the tap device
#
/usr/sbin/openvpn --rmtun --dev $1
#
# start firewall again
#
/sbin/service firestarter start
```

Allowing users to call the scripts

The two scripts above need to be run as root, so that they can modify the network settings of the system. The most convenient way to achieve that is to permit users of QEMU to call the scripts using the `sudo` command. To set this up, add the following to the file `/etc/sudoers`:

```
User_Alias QEMUERS = fred, john, milly, ...

Cmnd_Alias QEMU = /etc/qemu-ifup, /etc/qemu-ifdown

QEMUERS ALL=(ALL) NOPASSWD: QEMU
```

Starting QEMU with a TAP interface

Now create a script to start QEMU with a VLAN, and clean up after itself when it exits. This one uses tap0. Specifying `script=no` tells QEMU to just use the tap device without calling the scripts - we do this so that QEMU can be run as a regular user, not root.

```
#!/bin/sh
sudo /etc/qemu-ifup tap0
qemu -m 256 -hda disk.img -net nic -net tap,ifname=tap0,script=no
sudo /etc/qemu-ifdown tap0
```

Run that script, and it will create a TAP interface, bridge it to eth0, run QEMU, and drop the bridge and TAP interface again on exit.

Sockets

QEMU can connect multiple QEMU guest systems on a VLAN using TCP or UDP sockets.

described here (http://www.gnome.org/~markmc/qemu-networking.html)

SMB server

If the host system has a SMB server installed (SAMBA/CIFS on *nix), QEMU can emulate a virtual SMB server for the guest system using the -smb option. Specify the folder to be shared, and it will be available to the guest as \\10.0.2.4\qemu (or you can put 10.0.2.4 into the hosts or lmhosts file as smbserver and map to \\smbserver\qemu).

```
qemu -m 256 disk.img -smb /usr/workspace/testing01
```

This isn't strictly necessary, because guests in QEMU can typically access SMB servers in the host environment. It can be quite useful, however, for setting up independent workspaces for each QEMU guest without needing to configure SMB shares for each one.

External links

- documentation[1] (http://wiki.qemu.org/Documentation/Networking)
- forum post on bridging QEMU to Linux (http://www.fedoraforum.org/forum/showpost.php?p=530775&postcount=1)

Retrieved from "http://en.wikibooks.org/w/index.php?title=QEMU/Networking&oldid=2380916"