@ericdwhite | github://ericdwhite

« /ericdwhite/

permalink : 02 April 2011

g a Wireless Bridge on Your Laptop using Proxy ARP

Since I have had to figure this out twice from scratch, I decided to write it down for posterity.

In a previous article I explained how to bootstrap Ubuntu VMs and launch them with KVM. Part of this involved setting up a wireless bridge using **`parprouted`**. Since then I have learned of an easier method using Proxy ARP.

The details can be found here: Proxy ARP with Linux

I will summarize the relevant sections for working with KVM.

The assumptions

1. You have a KVM host and are only concerned with setting up a wireless bridge
2. You have **`sudo`** access on the host

High-level Overview

Steps:

1. Enable IPv4 tunnelling (/proc/sys/net/ipv4/ip_forward)
2. Create the tap interface (tunctl -u $USER -t tap0)
3. Enable Proxy ARP in the kernel (/proc/sys/net/ipv4/conf/[wlan0/tap0]/proxy_arp)
4. Configure the tap interface (ip)
5. Add a routes to the hosts via the tap (ip, route)

Please note that you will need **`root`** access, or sudo to complete a number of the steps.

Wireless Bridge for Laptop Hosts with KVM Guests

The idea is for the host to provide **`tap0`** interface to the KVM guest to use for connectivity. I have chosen

to bridge my wireless interface to the **tap0**.

Make Sure IP Forwarding is Enabled

Before creating a bridge IP forwarding must be enabled.

Is IP forwarding enabled? This can be verified by looking in the **/proc** file system. Note a value of ip_forward=1 means IP forwarding is enabled.

```
$ cat /proc/sys/net/ipv4/ip_forward
0
```

IP forwarding can be turned on temporarily using:

```
$ sudo sh -c 'echo 1 > /proc/sys/net/ipv4/ip_forward'
```

This change can be made permanent by changing:

```
/etc/sysctl.conf
...
  net.ipv4.ip_forward=1
```

Create the tap interface

The **tap0** interface is created using **tunctl** and is made accessible to the host user launching the KVM instance via the **-u** parameter.

```
$ On the HOST

$ route -n
  Kernel IP routing table
  Destination     Gateway         Netmask         Flags Metric Ref    Use Iface
  10.20.80.0      0.0.0.0         255.255.255.0   U     2      0        0 wlan0
  0.0.0.0         10.20.80.1      0.0.0.0         UG    0      0        0 wlan0

$ sudo tunctl -u $USER -t tap0
  Set 'tap0' persistent and owned by uid 1000
```

Note at this point the **tap0** interface has no IP.

```
$ /sbin/ifconfig tap0
tap0      Link encap:Ethernet  HWaddr 7a:b2:e6:51:ef:c0
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Enable Proxy ARP on the wireless interface and tap0

Again the proxy ARP is configured in the kernel, and it current status can be determined by running:

```
$ cat /proc/sys/net/ipv4/conf/wlan0/proxy_arp
0

$ cat /proc/sys/net/ipv4/conf/tap0/proxy_arp
0
```

A `1` means it's enabled. If disabled then enable it:

```
$ sudo sh -c 'echo 1 > /proc/sys/net/ipv4/conf/wlan0/proxy_arp'
$ sudo sh -c 'echo 1 > /proc/sys/net/ipv4/conf/tap0/proxy_arp'
```

Configure the tap interface with an IP address

The tap interface is given a static IP address `10.20.80.50`, and promiscuous mode enabled.

```
$ sudo ip addr add 10.20.80.50 dev tap0
$ sudo ip link set tap0 up
$ sudo ip link set tap0 promisc on

$ /sbin/ifconfig tap0
  tap0      Link encap:Ethernet  HWaddr ce:0a:d7:98:70:14
            inet addr:10.20.80.50  Bcast:0.0.0.0  Mask:255.255.255.255
            inet6 addr: fe80::cc0a:d7aa:fa23:7014/64 Scope:Link
            UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
            RX packets:0 errors:0 dropped:0 overruns:0 frame:0
            TX packets:0 errors:0 dropped:25 overruns:0 carrier:0
            collisions:0 txqueuelen:500
            RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Configuring the HOST routes

Add a route from the host to each guest using `ip route`. This requires the guests to have static IP addresses.

In this example the guest is on IP 10.20.80.21.

```
$ sudo ip route add 10.20.80.21 dev tap0

$ route -n
  Kernel IP routing table
  Destination     Gateway         Netmask         Flags Metric Ref    Use Iface
  10.20.80.21     0.0.0.0         255.255.255.255 UH    0      0        0 tap0
  10.20.80.0      0.0.0.0         255.255.255.0   U     2      0        0 wlan0
  0.0.0.0         10.20.80.1      0.0.0.0         UG    0      0        0 wlan0
```

Testing

The first step is to launch the guest with the network interface provided by **tap0**. I will assume the guest has a fixed IP and the IP is known.

```
$ pwd (As described in the previous article)
maverick-amd64-bootstrap

kvm -drive file=ubuntu-maverick-amd64.img,index=0,media=disk \
    -net tap,ifname=tap0,script=no,downscript=no -net nic \
    -m 256m
```

Next from the host try to ping the guest:

```
host$ ping 10.20.80.21
  PING 10.20.80.21 (10.20.80.21) 56(84) bytes of data.
  64 bytes from 10.20.80.21: icmp_req=1 ttl=64 time=0.443 ms
  ...
```

Next from the guest try to ping the host:

```
guest$ ping 10.20.80.50
  PING 10.20.80.50 (10.20.80.50) 56(84) bytes of data.
  64 bytes from 10.20.80.50: icmp_req=1 ttl=64 time=0.443 ms
  ...
```

Then try to ping the outside world from the guest (I start with the gateway of the host).

```
guest$ ping 10.20.80.1
  PING 10.20.80.1 (10.20.80.1) 56(84) bytes of data.
  64 bytes from 10.20.80.1: icmp_req=1 ttl=64 time=0.443 ms
  ...
```

Finally, if at all possible connect to another host and see if the guest is reachable.

Troubleshooting

If things are not working I found that I had:

1. Not set **proxy_arp** on one of the interface
2. Did not enable IP forwarding
3. Did not define the route from the host to guest (or used the wrong guest IP)

```
$ cat /proc/sys/net/ipv4/ip_forward
$ cat /proc/sys/net/ipv4/conf/wlan0/proxy_arp
$ cat /proc/sys/net/ipv4/conf/tap0/proxy_arp
$ route -n
```

Aside: Killing off the tapped interface.

```
$ sudo ifconfig tap0 down
$ sudo tunctl -u $USER -d tap0
$ sudo sh -c 'echo 0 > /proc/sys/net/ipv4/ip_forward'
$ sudo sh -c 'echo 0 > /proc/sys/net/ipv4/conf/wlan0/proxy_arp'
$ sudo sh -c 'echo 0 > /proc/sys/net/ipv4/conf/tap0/proxy_arp'

$ route -n
```

Resources

These sites have been invaluable and I have tried to summarise a condensed version of them here which meet my goals, but this would not have been possible without them:

- Proxy ARP with Linux
- Debian+KVM: proxy_arp for individual ip addresses
- QEMU Networking Tips

Bob • a year ago

Sorry, in my last message I said XEN doesn't work the same as XEN. I meant it doesn't work the same as KVM. Also I'm using XEN 4.1 with Debian distro.

• Reply • Share ›

Bob • a year ago

Eric, I've spent a month trying to figure out how to add wireless access to a XEN WinXP guest. Even though XEN doesn't work quite the same as XEN, your instructions we're the most helpful. With them I was finally able to get my WinXP guest to access the internet via wireless!!! I'm willing to provide detailed steps on how I did it, if you're interested. If not maybe you could point me someone else that might be interested. It took me so long to figure, I would like to share what I did to help others. Thanks again for providing this. Best Regards Bob

• Reply • Share ›

Eric D. White Mod Bob • a year ago

Bob, I'm glad you this helped you. Feel free to put the tips for getting it to work on WinXP in the comments. Most of this blog is about documenting things I found difficult to get working, with the hope it will help the next person.

• Reply • Share ›