

Re: Bridging network adapters in Linux

Source: <http://linux.derkeiler.com/Newsgroups/comp.os.linux.networking/2008-11/msg00226.html>

- *From:* "kiloVolts" <mantrap@xxxxxxxxxx>
 - *Date:* Mon, 24 Nov 2008 15:04:49 -0800
-

"Wolfgang Draxinger" <wdraxinger@xxxxxxxxxxxxxxxxxxxx> wrote in message news:3d3ov5-7rj.ln1@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

kiloVolts wrote:

* * * Cross Posted * * *

comp.os.linux.networking, alt.comp.networking.routers,
alt.comp.networking.connectivity

Hi All,

Tuesday I purchased an Acer Aspire One with Linpus Linux preinstalled and I am not disappointed. Wednesday I discovered how to open a terminal window.

The Aspire One has a competent 11g wireless adapter. This is an accomplishment in a university village environment with many 11n routers nearby. I would like to bridge the wireless adapter and the Ethernet adapter on my Aspire One so I can connect to my Dlink DI604 wired router and feed Internet connectivity to all my other gear. This is easily done by mouse clicks in MS Vista. I presume the command line and some file editing is involved in Linux. Could someone please give me a head start on the solution?

Execute the following commands as root user:

```
brctl addbr br0  
brctl addif $wlandev  
brctl addif $ethdev
```

with \$wlandev being the name of your WLAN adapter and \$ethdev being the name of the ethernet adapter. On my systems those would be "wlan0" and "eth0".

Re: Bridging network adapters in Linux

You can get the names of the available devices with

```
ip link show
```

Then if you're using DHCP, execute some DHCP client on the newly created bridge (dhclient, dhcpcd, pump, udhcp, it depends on what you've got installed).

```
dhclient br0
```

```
or
```

```
dhcpcd br0
```

In case of static addressing, give the bridge device a static address with

```
ip addr add $address/$netbits dev br0
```

```
e.g.
```

```
ip addr add 10.1.2.3/8 dev br0
```

and assign a route.

Either a default route

```
ip route add default via $gateway_address dev br0
```

```
e.g.
```

```
ip route add default via 10.0.0.1 dev br0
```

and/or static routes

```
ip route add $network/$netbits via $gateway
```

```
e.g.
```

```
ip route add 192.168.22.0/24 via 10.0.0.2
```

But eventually you don't want your bridge participate in the network at all, but just pass data between ports as if it were a switch. This is done easily with:

```
brctl addbr br0
```

```
brctl addif $wlandev
```

```
brctl addif $ethdev
```

```
for dev in $intif, $extif ; do ip link set $dev up ; ip link \
set $dev promisc on ; ip link set $dev arp on ; done
```

Note that you asked for a bridge, i.e. a device that will connect two network devices assuming, that the physical networks it bridges are in the same logical address space, thus forming a single subnet in the IP sense. So you need unique IP addresses for every device in the network. The good thing is, that if configured correctly, DHCP will work through the bridge. So if you're on a campus network, where students can simply attach

Re: Bridging network adapters in Linux

their devices and don't worry about correct configuration it will work.

So far we've only set up the interfaces, but you'll also want some address resolver. If using DHCP the nameservers will be automatically put into `/etc/resolv.conf` ; if you're static, then ask your administrator for the nameserver addresses and put them into `/etc/resolv.conf` like the following:

```
echo "nameserver $ns1" > /etc/resolv.conf
echo "nameserver $ns2" >> /etc/resolv.conf
```

(note the double '>', i.e. '>>' in the second command). `$ns1` and `$ns2` are the addresses of your local network's DNS caches. (If there are no local caches, you could setup your own cache, resolving against the global root servers, but that's a whole different story, normally you don't need it).

If what you want is internet connection sharing over a single IP address, then you've got a whole different deal. It's called network address translation (NAT) and gets configured in a whole different way.

First you set up you two devices as usual. The external device is either dynamically or statically configured

```
if dynamic
dhclient $extif
e.g.
dhclient eth0
```

```
if static
ip addr add $extip/$ext_netbits dev $extif
e.g.
ip addr add 10.1.2.3/8 dev eth0
```

and don't forget the default route (if static)

```
ip route add default via $gateway_address dev $extif
e.g.
ip route add default via 10.0.0.1 dev eth0
```

Now set up the internal interface `$intif`. We're choosing some private address range for that, but make sure, that it doesn't clash with your `$extif` address space – however if your `$extif` is in a private address space, you could just go for bridging like outlined above, and be fine. You've to do NAT only, if you got just one global IP address assigned. However NAT works well, even if you're NAT-ing a private net into another NAT-ed private net.

Re: Bridging network adapters in Linux

```
ip addr add $intip/$int_netbits dev $intif  
e.g.  
ip addr add 192.168.1.1/24 dev wlan0
```

Here comes the tricky part: You've to tell your Linux box, that it shall forward/route packets between wlan0 and eth0, but apply a NAT in the process. This is done using the iptables mechanism. First you've to enable forwarding at all:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

(you can set this permanently in /etc/sysctl.conf)

However with that setting it will just route packets between the two networks, but since the external network knows nothing about the structure and addresses in your internal networks, the packets will make it never beyond the first router. So the packet's addresses must be rewritten to appear as if they originated from your NAT forwarding router:

First load the required iptables modules:

```
modprobe ip_tables  
modprobe ip_conntrack  
modprobe ip_conntrack_ftp  
modprobe ip_conntrack_irc  
modprobe iptable_nat  
modprobe ip_nat_ftp  
modprobe ip_nat_irc
```

should cover the basics (those protocol specific nat and conntrack modules are necessary, since some protocols will open additional connections or even open listening ports, which relate to existing a existing connection and thus need matching NAT rules applied, the conntrack modules take the required prerequisites).

Now put the iptables into a "sane" state (depending on the desired configuration, "sane" might look different):

```
iptables -P INPUT ACCEPT  
iptables -F INPUT  
iptables -P OUTPUT ACCEPT  
iptables -F OUTPUT  
iptables -P FORWARD REJECT  
iptables -F FORWARD  
iptables -t nat -F
```

-P sets a default policy, -F empties the table, so that only the default policy applies. Which means for the forwarding table: Don't route anything yet. But you want some routing possible,

Re: Bridging network adapters in Linux

but only for connections from the internal network to the external, and incoming connections only if they relate to existing ones (you remember the conntrack modules above):

```
iptables -A FORWARD -i $extif -o $intif -m state --state \
ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -i $intif -o $extif -j ACCEPT
```

e.g.

```
iptables -A FORWARD -i eth0 -o wlan0 -m state --state \
ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
```

Now tell iptables, that the packets routed between \$intif and \$extif should be NAT-ed

```
iptables -t nat -A POSTROUTING -o $EXTIF -j MASQUERADE
e.g.
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

That about does it. Now on your internal side, you can either configure hosts per DHCP, which requires you to setup a DHCP server.

Or configure them statically: For that you select IP addresses from the same address range in which the internal interface of your router is configured, e.g. 192.168.1.x/24 and tell the devices, that the router (in this example 192.168.1.1) is the default gateway. Since there's no DHCP internally (yet), you must tell the clients also the nameserver addresses. That's just like in the bridging case. Even if the clients are NAT-ed, you tell them the nameserver addresses on the external network.

Now you might wonder: "Do I've to type that stuff myself every time I boot the system?". The answer is "No!" Most distributions have a standard way to store such configuration. And even if they don't you can put all those commands into a shell script to automate things (a shell script is kinda a DOS/Windows .bat file, but a lot more powerfull, there are tons of tutorials on shell scripting out there).

HTH

Wolfgang Draxinger

--

E-Mail address works, Jabber: hexarith@xxxxxxxxxxx, ICQ: 134682867

Thank you.

Re: Bridging network adapters in Linux

The command `brctl` is not installed in Linpus Linux on my Acer Aspire One. How do I Install a package that contains `brctl`?

.