gentoo linux™ (/) **Wiki**

Home Router

From Gentoo Wiki
This document details how to turn an old Gentoo machine into a router for connecting a home network to the Internet.

Contents

Introduction

Building a personal router out of old spare parts has many advantages over buying a pre-made router built by big companies (Linksys, D-Link, Netgear, etc). The biggest advantage by far is control over the connection. The other advantages can be left up to the user's imagination; just about anything can be done in this scenario, it is simply a matter of need.

This guide will provide instructions on how to setup Network Address Translation (NAT) on a custom router (kernel and iptables), add and configure common services (Domain Name System (DNS) via net-dns/dnsmasq (http://packages.gentoo.org/package/net-dns/dnsmasq), DHCP via net-misc/dhcpcd (http://packages.gentoo.org/package/net-misc/dhcpcd), ADSL via net-dialup/ppp (http://packages.gentoo.org/package/net-dialup/ppp)), and conclude with some elaborate and fun things custom routers are capable of (port forwarding, traffic shaping, proxies/caching, etc...).

Before getting started, please review the list of basic requirements:

1. A computer that has at least 2 Network Interface Cards (NICs) installed.
2. The configuration settings for an Internet connection (may include things like IP/DNS/Gateway /username/password).
3. (Optionally) a Wi-Fi card that supports master mode. Master mode is recommended to avoid NICs with a bypass feature since some NICs require proprietary drivers. Without access to the proprietary driver the card will remain in bypass mode permanently and will not be usable.
4. A bit of spare time and Gentoo loving in order to successfully follow this guide and implement a well functioning home router.

The conventions used in this guide are:

- eth0 - NIC connected to the Local Area Network (LAN) or network bridge consisting of multiple NICs
- eth1 - NIC connected to the Wide Area Network (WAN)
- LAN utilizes the private 192.168.0.xxx network
- Router is hardcoded to the standard 192.168.0.1 IP address
- Router is running Linux 2.4 or 2.6; other versions of the kernel are not supported by this guide).

Important
Due to security precautions, it is highly suggested to shut down any unneeded services on the router until the firewall is up. To view the currently running services run `rc-status`.

Kernel setup (know thyself first)

The kernel needs to have the drivers installed for both NICs present on the system. To see if the cards are already setup use the `ifconfig` command. The output may differ slightly from the following example. What matters is that the interface shows up.

root # ifconfig -a

```
eth0      Link encap:Ethernet  HWaddr 00:60:F5:07:07:B8
          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:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
          Interrupt:11 Base address:0x9800

eth1      Link encap:Ethernet  HWaddr 00:60:F5:07:07:B9
          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:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
          Interrupt:10 Base address:0x9400
```

If only one (or none) of the two cards show up try running `lspci | grep Ethernet`.

Note
The `lspci` utility is part of the sys-apps/pciutils (http://packages.gentoo.org/package/sys-apps /pciutils) package and can be installed by running `emerge sys-apps/pciutils`.

Once the make(s)/model(s) of the NIC(s) has been obtained, configure the kernel with support for the correct drivers. For more information on kernel configuration see the Kernel Configuration Guide (/wiki/Kernel/Gentoo_Kernel_Configuration_Guide).

The next thing needed is support for iptables and NAT (and packet shaping if desired). The following list is split up into always required (*), required only for ADSL via PPPoE (a), suggested for everyone (x), and only for shaper (s) features. It does not matter whether the features are built into the kernel or as modules as long as when the feature is needed, the correct module(s) are loaded. For more information on loading modules see the

KERNEL **Network Options**

```
Networking options  --->
   [*] TCP/IP networking
      [*] IP: advanced router
   [*] Network packet filtering (replaces ipchains)
```

When using a 2.4.x kernel, the following must be enabled for DHCP:

KERNEL **Network Options**

```
   [*] Socket Filtering

   IP: Netfilter Configuration  --->
      [*] Connection tracking (required for masq/NAT)
         [x] FTP protocol support
         [x] IRC protocol support
      [*] IP tables support (required for filtering/masq/NAT)
         [*] IP range match support
         [x] MAC address match support
         [*] Multiple port match support
         [*] Packet filtering
            [*] REJECT target support
            [x] REDIRECT target support
         [*] Full NAT
            [*] MASQUERADE target support
         [s] Packet mangling
            [s] MARK target support
         [x] LOG target support

   QoS and/or fair queueing  --->
      [s] QoS and/or fair queueing
         [s] HTB packet scheduler
         [s] Ingress Qdisc

   [a] PPP (point-to-point protocol) support
      [a] PPP filtering
      [a] PPP support for async serial ports
      [a] PPP support for sync tty ports
      [a] PPP Deflate compression
      [a] PPP BSD-Compress compression
      [a] PPP over Ethernet
```

Note
Some things may be slightly different in a 2.4.x vs. a 2.6.x kernel, however things should not be too difficult to figure out. Even among 2.6 kernels, these options have a tendency to move around. Good luck!

Hug the WAN (a.k.a. The Internet)

Intro

There are *many* ways to connect to the Internet, however there are generally only a couple of ways that are used by most of the public. ADSL (PPPoE) and cable modems (static/dynamic) are the two most common methods ISP (Internet Service Providers) provide. If there are other methods available, feel free to add them to this wiki article. Skip any of the following sections in this chapter that are not applicable to the needed use case. This chapter addresses getting the router connected to the Internet via eth1.

ADSL and PPPoE

All the fancy PPPoE software that used to be provided by rp-pppoe (Roaring Penguin (http://www.roaringpenguin.com/)) has been integrated into the standard PPP package (http://samba.org/ppp/). Simply `emerge ppp` to install PPPoE. Remember how username and password information was a requirement listed above? Load up `/etc/conf.d/net` in a favorite text editor and configure it accordingly.

Replace `<username>` and `<user_password>` in the following example with the required username with the password:

FILE **`/etc/conf.d/net`** **Use ADSL over eth1 for ppp0**

```
config_ppp0="ppp"
link_ppp0="eth1"
plugins_ppp0="pppoe"
pppd_ppp0="
  defaultroute
  usepeerdns
"
username_ppp0="<username>"
password_ppp0="<user_password>"
```

```
root # ln -s net.lo /etc/init.d/net.ppp0
root # rc-update add net.ppp0 default
root # /etc/init.d/net.ppp0 start
```

Warning
When the DSL interface comes up, it will create a "ppp0" entry to the output of `ifconfig` command. Although the NIC is called eth1, the IP address is actually bound to ppp0. From now on substitute `eth1` with `ppp0`

Warning
Be sure to change the permissions of the `/etc/conf.d/net` file so that only users with *root* privileges can read/write to it. This important because the a username and password have been entered to the file in plain text format.

Warning
For users transitioning from the net-dialup/rp-pppoe (http://packages.gentoo.org/package/net-dialup/rp-pppoe) package, or for users who experience weird connection resets, see the MTU section in the Troubleshooting chapter below.

Cable and/or dynamic/static IP

If a static IP is necessary then additional configuration details will be required. Static users IP users will need to add the IP address, gateway address, and DNS server addresses.

Dynamic IP Users:

```
root # emerge --ask net-misc/dhcpcd
```

FILE **/etc/conf.d/net** **Dynamic IP addresses**

```
config_eth1="dhcp"
```

Static IP Users:

FILE **/etc/conf.d/net** **Static IP address configuration**

```
config_eth1="66.92.78.102/24 brd 66.92.78.255"
routes_eth1="default via 66.92.78.1"
```

FILE **/etc/resolv.conf** **Adding DNS information**

```
nameserver 123.123.123.123
```

Dynamic and Static Setup:

```
root # ln -s net.lo /etc/init.d/net.eth1
root # rc-update add net.eth1 default
root # /etc/init.d/net.eth1 start
```

After working through the changes above the system should be ready to continue.

Hug the LAN (bring along some friends)

This step is a breeze compared to the previous one. To use *more* than two devices (more than the one for LAN and the one for WAN), a Network bridge (/wiki/Network_bridge) will need to be setup between all NICs using the LAN. This will allow multiple NICs to be reached by the same IP address.

If a network bridge will be necessary, follow the instructions to set up a Network bridge (/wiki /Network_bridge). The name of the bridge (default br0) will then replace eth0 for the LAN device in the steps in this wiki. If a large number of network devices in the home router, consider renaming them via udev to make administration easier. Setting up a bridge and renaming devices is completely optional but recommended for larger home networks.

When creating a Wi-Fi access point make sure the Wi-Fi card supports master mode and set up Hostapd (/wiki/Hostapd).

FILE **/etc/conf.d/net**

```
config_eth0="192.168.0.1/24 brd 192.168.0.255"
```

```
root # rc-update add net.eth0 default
root # /etc/init.d/net.eth0 start
```

LAN Services (because we're nice people)

DHCP server

It would be nice if everyone in the house could plug their computers into the network and things would just work. No need to remember mind-numbing details or make them stare at confusing configuration screens! Life would be grand, eh? Introducing the Dynamic Host Configuration Protocol (DHCP) and why everyone should care.

DHCP is exactly what its name implies: a protocol that allows dynamic configuration of hosts automatically. Run a DHCP server on the router, give it all the information about the network (valid IPs, DNS servers, gateways, etc...), then when the other hosts start up, they can run a DHCP *client* to automatically configure themselves. No fuss, no muss! For more information about DHCP, visit Wikipedia's DHCP article (http://en.wikipedia.org/wiki/DHCP).

This section will use the net-dns/dnsmasq (http://packages.gentoo.org/package/net-dns/dnsmasq) package which will provide both DHCP and DNS services. For now lets focus on the DHCP aspect. Note: to run a different DHCP server, another example can be found in the Fun Things section below. Also, to tinker with the DHCP server settings read the comments in the `/etc/dnsmasq.conf` file.

```
root # emerge --ask net-dns/dnsmasq
```

FILE **/etc/dnsmasq.conf**

```
dhcp-range=192.168.0.100,192.168.0.250,72h
interface=eth0
```

```
root # rc-update add dnsmasq default
root # /etc/init.d/dnsmasq start
```

Setting the interface is very important. Using default dnsmasq settings will open the router to DNS amplification attacks which could create some scary email from the ISP providing the connection. Check to make sure the router is not allowing for DNS amplification attacks by using [1] (http://openresolver.com).

Now the little router is a bona-fide DHCP server. Plug in those computers and watch them work! With Windows systems navigate to the `TCP/IP Properties` and select the `Obtain an IP address automatically` and `Obtain DNS server address automatically` options. Sometimes the changes are not instantaneous, so opening a command prompt and running `ipconfig /release` and `ipconfig /renew` might be necessary. Enough about Windows, time to get back to everyone's favorite penguin!

DNS server

When people want to visit a place on the Internet, they remember names, not a string of funky numbers. After all, what is easier to remember, eBay.com or 66.135.192.87? This is where the DNS steps in. DNS servers run all over the Internet, and whenever someone wants to visit eBay.com, these servers turn the text "eBay.com" (what we understand) into IP address "66.135.192.87" (what computers understand). For more information about DNS visit Wikipedia (http://en.wikipedia.org /wiki/DNS).

Since dnsmasq is being used for the DHCP server, and it includes a DNS server, there is nothing left to do here! The little router is already providing DNS to its DHCP clients. Shouldn't everything be this easy? ;)

It is possible to choose other DNS servers if they are more comfortable to setup. dnsmasq is used in this article because it was designed to do exactly what this guide required. It is a little DNS caching/forwarding server for local networks. The scope of this howto is not to provide DNS for a domain; but it does offer simple DNS services to every user of a home-based LAN.

NAT (a.k.a. IP-masquerading)

At this point, people on the network can talk to each other and look up hostnames via DNS, but they still ca not actually connect to the Internet. While the network administrator (the person reading this) may think it is great (more bandwidth for the Admin!), the other users are probably not very happy without an Internet connection.

This is where Network Address Translation (NAT) steps in. NAT is a way of connecting multiple computers in a private LAN to the Internet when a small number of public IP addresses are available. Typically a home Internet user is provided with 1 public IP address by an ISP for the whole house to connect to the Internet. NAT is the magic that makes this possible. For more information about NAT, please visit Wikipedia (http://en.wikipedia.org/wiki/NAT).

Note

Before getting started, make sure IPtables has been installed on the system. If it is not installed, install it: `emerge iptables`

After IPtables is installed, flush the current rules:

```
root # iptables -F
root # iptables -t nat -F
```

Setup default policies to handle unmatched traffic:

```
root # iptables -P INPUT ACCEPT
root # iptables -P OUTPUT ACCEPT
root # iptables -P FORWARD DROP
```

Copy and paste the following:

```
root # export LAN=eth0
root # export WAN=eth1
```

The next step locks the services so they only work from the LAN:

```
root # iptables -I INPUT 1 -i ${LAN} -j ACCEPT
root # iptables -I INPUT 1 -i lo -j ACCEPT
root # iptables -A INPUT -p UDP --dport bootps ! -i ${LAN} -j REJECT
root # iptables -A INPUT -p UDP --dport domain ! -i ${LAN} -j REJECT
```

(Optional) Allow access to the ssh server from the WAN:

```
root # iptables -A INPUT -p TCP --dport ssh -i ${WAN} -j ACCEPT
```

Drop TCP / UDP packets to privileged ports:

```
root # iptables -A INPUT -p TCP ! -i ${LAN} -d 0/0 --dport 0:1023 -j DROP
root # iptables -A INPUT -p UDP ! -i ${LAN} -d 0/0 --dport 0:1023 -j DROP
```

Finally add the rules for NAT:

```
root # iptables -I FORWARD -i ${LAN} -d 192.168.0.0/255.255.0.0 -j DROP
root # iptables -A FORWARD -i ${LAN} -s 192.168.0.0/255.255.0.0 -j ACCEPT
root # iptables -A FORWARD -i ${WAN} -d 192.168.0.0/255.255.0.0 -j ACCEPT
root # iptables -t nat -A POSTROUTING -o ${WAN} -j MASQUERADE
```

Inform the kernel that IP forwarding is OK:

```
root # echo 1 > /proc/sys/net/ipv4/ip_forward
root # for f in /proc/sys/net/ipv4/conf/*/rp_filter ; do echo 1 > $f ; done
```

Instruct the IPtables daemon to save the changes to the rules, then add IPtables to the default runlevel:

```
root # /etc/init.d/iptables save
root # rc-update add iptables default
```

FILE **/etc/sysctl.conf**

```
net.ipv4.ip_forward = 1
net.ipv4.conf.default.rp_filter = 1
```

For dynamic Internet the following setting should be enabled:

FILE **/etc/sysctl.conf**

```
net.ipv4.ip_dynaddr = 1
```

Once the above text has been entered the rest of the network users should now be able to use the Internet as if they were directly connected themselves.

The `ip_dynaddr` option is useful for dial on demand systems or when the ISP gives out dynamic addresses. This works around the problem where a connection is attempted before the Internet interface is fully setup. This provides a smoother network experience for users behind the router.

Fun things (for a rainy day)

Intro

Believe it or not, it is done! From here on out, some other common topics that may interest will be covered. Everything in the following sections are completely optional.

Port forwarding

Sometimes users need to be able to host services on a computer behind the router, or need to be able to connect remotely to a computer behind the router. Perhaps a FTP, HTTP, SSH, or VNC server is needed on one or more machines behind the router and outsiders need to connect to them all. The only caveat to Port Forwarding is only one service/machine combo can be established per port. For example, there is no practical way to setup three FTP servers behind the router and connect to them all through port 21; only one system can be on port 21 while the others would need to be on other ports (port 123 and port 567 would be fine options).

All the port forwarding rules are of the form `iptables -t nat -A PREROUTING [-p protocol] --dport [external port on router] -i ${WAN} -j DNAT --to [ip/port to forward to]`. Unfortunately, iptables does not accept hostnames when port forwarding. When forwarding an external port to the same port on the internal machine, omit the destination port. See the iptables(8) man page for more information.

```
root # export LAN=eth0
root # export WAN=eth1
```

Forward port 2 to ssh on an internal host:

```
root # iptables -t nat -A PREROUTING -p tcp --dport 2 -i ${WAN} -j DNAT --to
192.168.0.2:22
```

FTP forwarding to an internal host:

```
root # iptables -t nat -A PREROUTING -p tcp --dport 21 -i ${WAN} -j DNAT --to
192.168.0.56
```

HTTP forwarding to an internal host:

```
root # iptables -t nat -A PREROUTING -p tcp --dport 80 -i ${WAN} -j DNAT --to
192.168.0.56
```

VNC forwarding for internal hosts:

```
root # iptables -t nat -I PREROUTING -p tcp --dport 5900 -i ${WAN} -j DNAT --to
192.168.0.2
root # iptables -t nat -I PREROUTING -p tcp --dport 5901 -i ${WAN} -j DNAT --to
192.168.0.3:5900
```

To VNC in to 192.168.0.3, then add `:1` to the router's hostname.

SAMBA forwarding to an internal host (excess ports to cover Windows):

```
root # iptables -t nat -I PREROUTING -p tcp --dport 135 -i ${WAN} -j DNAT --to
192.168.0.2
root # iptables -t nat -I PREROUTING -p tcp --dport 139 -i ${WAN} -j DNAT --to
192.168.0.2
root # iptables -t nat -I PREROUTING -p tcp --dport 445 -i ${WAN} -j DNAT --to
192.168.0.2
root # iptables -t nat -I PREROUTING -p udp --dport 137:138 -i ${WAN} -j DNAT --to
```

```
192.168.0.2
root # iptables -t nat -I PREROUTING -p udp --dport 445 -i ${WAN} -j DNAT --to
192.168.0.2
```

Bittorrent forwarding:

```
root # iptables -t nat -A PREROUTING -p tcp --dport 6881:6889 -i ${WAN} -j DNAT --to
192.168.0.2
```

eDonkey/eMule forwarding:

```
root # iptables -t nat -A PREROUTING -p tcp --dport 4662 -i ${WAN} -j DNAT --to
192.168.0.55
```

Game Cube Warp Pipe support:

```
root # iptables -t nat -A PREROUTING -p udp --dport 4000 -i ${WAN} -j DNAT --to
192.168.0.56
```

Playstation 2 Online support:

```
root # iptables -t nat -A PREROUTING -p tcp --dport 10070:10080 -i ${WAN} -j DNAT --to
192.168.0.11
root # iptables -t nat -A PREROUTING -p udp --dport 10070:10080 -i ${WAN} -j DNAT --to
192.168.0.11
```

Xbox Live:

```
root # iptables -t nat -A PREROUTING -p tcp --dport 3074 -i ${WAN} -j DNAT --to
192.168.0.69
root # iptables -t nat -A PREROUTING -p udp --dport 3074 -i ${WAN} -j DNAT --to
192.168.0.69
root # iptables -t nat -A PREROUTING -p udp --dport 88 -i ${WAN} -j DNAT --to
192.168.0.69
```

Identd (for IRC)

Internet Relay Chat utilizes the ident service pretty heavily. Now that the IRC clients are behind the router, a way to host ident for both the router and the clients is needed. A server has been created for this purpose. It is called net-misc/midentd (http://packages.gentoo.org/package/net-misc/midentd).

```
root # emerge --ask net-misc/midentd
root # rc-update add midentd default
root # /etc/init.d/midentd start
```

There are a few other ident servers in the Portage tree. Other viable options are net-misc/oidentd (http://packages.gentoo.org/package/net-misc/oidentd) and net-misc/fakeidentd (http://packages.gentoo.org/package/net-misc/fakeidentd).

Time server

Keeping the system time correct is essential to maintaining a healthy system. One of the most common ways of accomplishing this is with the Network Time Protocol (NTP) and the net-misc/ntp (http://packages.gentoo.org/package/net-misc/ntp) package (which provides implementations for both server and client).

Many users run ntp clients on their computers. Obviously, the more clients in the world, the larger the load ntp servers need to shoulder. In environments like home networks an NTP server can be setup locally to help keep the load down on public servers while still providing the proper time to local systems. As an added bonus, private updates will be a lot faster for the local clients! The setup is simple: run a NTP server on the router that synchronizes itself with the public Internet servers while, at the same time, providing the time to the rest of the computers in the network. To get started, simply `emerge ntp` on the router and edit `/etc/conf.d/ntp-client` as desired.

```
root # rc-update add ntp-client default
```

FILE **/etc/ntp.conf**

```
restrict default ignore
restrict 192.168.0.0 mask 255.255.255.0 notrust nomodify notrap
```

These will allow only NTP clients with an IP address in the 192.168.0.xxx range to use the NTP server.

```
root # rc-update add ntpd default
root # /etc/init.d/ntp-client start
root # /etc/init.d/ntpd start
```

Note
Make sure to allow inbound and outbound communication on the NTP port (123/udp) when setting up the server. The client just needs outbound access on port 123 over UDP.

Now, on the clients, run `emerge ntp`. By running the NTP client setup is a lot simpler.

In `/etc/conf.d/ntp-client`, change the `pool.ntp.org` server in the `NTPCLIENT_OPTS` variable to `192.168.0.1`

```
root # rc-update add ntp-client default
root # /etc/init.d/ntp-client start
```

Rsync server

For those who run multiple Gentoo boxes on the same LAN, it is wise to prevent every machine running `emerge --sync` with remote servers. By setting up a *local* rsync, both personal bandwidth and the Gentoo rsync servers' bandwidth is saved. The process is relatively simple.

Note
For a much more in-depth rsync guide, please see the official rsync guide (/wiki /Project:Infrastructure/Rsync).

Since every Gentoo machine requires rsync, there is no need to emerge it. Edit the default `/etc/rsyncd.conf` config file, uncomment the `[gentoo-portage]` section, and make sure to add an `address` option. All other defaults should be already set correctly.

FILE **/etc/rsyncd.conf**

```
pid file = /var/run/rsyncd.pid
use chroot = yes
read only = yes
address = 192.168.0.1

[gentoo-portage]
  path = /mnt/space/portage
  comment = Gentoo Linux Portage tree
  exclude = /distfiles /packages
```

The service then needs to be started (again, the defaults are OK).

```
root # /etc/init.d/rsyncd start
root # rc-update add rsyncd default
```

Only thing left is to set tell the clients to sync against the router.

FILE **/etc/portage/make.conf** **Setup SYNC variable to new rsync server**

```
SYNC="rsync://192.168.0.1/gentoo-portage"
```

Mail server

Sometimes it is nice to run a Simple Mail Transfer Protocol (SMTP) server on the router. Each user may have their own reason for wanting to do so, however one advantage to running SMTP on the router is the users see mail as being sent instantly and the work of retrying/routing is left up to the mail server. Some ISPs do not allow for mail relaying for accounts that are not part of their network (like Verizon). Also, throttling the delivery of mail may be needed so that large attachments will not seriously lag the Internet connection.

```
root # emerge --ask mail-mta/netqmail
```

Make sure the output of the `hostname` command is correct:

```
root # emerge --config netqmail
root # iptables -I INPUT -p tcp --dport smtp ! -i ${LAN} -j REJECT
root # ln -s /var/qmail/supervise/qmail-send /service/qmail-send
root # ln -s /var/qmail/supervise/qmail-smtpd /service/qmail-smtpd
root # cd /etc/tcprules.d
```

Edit `tcp.qmail-smtp` and add an entry like so to the allow section:

FILE **niltcp.qmail-smtp**

```
192.168.0.:allow,RELAYCLIENT=""
```

```
root # make
root # rc-update add svscan default
root # /etc/init.d/svscan start
```

When e-mail is setup on the hosts in the network, tell them the SMTP server is 192.168.0.1. Visit the netqmail homepage (http://netqmail.org/) for more documentation on netqmail usage.

Full DHCP server

Earlier dnsmasq was used to provide DHCP service to all DHCP clients. For most people with a simple small LAN, this is perfect, however there may needs something with more features. Thus a full-featured DHCP server is provided by the ISC (http://www.isc.org/products/DHCP) folks for users who crave the maximum.

```
root # emerge --ask net-misc/dhcp
```

FILE **/etc/dhcp/dhcpd.conf** **Here is a simple configuration file**

```
authoritative;
ddns-update-style interim;
subnet 192.168.0.0 netmask 255.255.255.0 {
    range 192.168.0.100 192.168.0.250;
    default-lease-time 259200;
    max-lease-time 518400;
    option subnet-mask 255.255.255.0;
    option broadcast-address 192.168.0.255;
    option routers 192.168.0.1;
    option domain-name-servers 192.168.0.1;
}
```

In `/etc/conf.d/dhcpd` set `IFACE` to "eth0".

```
root # rc-update add dhcpd default
root # /etc/init.d/dhcpd start
```

This is the minimal setup required to replace the dnsmasq DHCP functionality used earlier. The DHCP features in dnsmasq should be disabled? If not, comment out the `dhcp-range` setting in `/etc/dnsmasq.conf` and restart the service.

Connect another LAN (or two or three)

Sometimes the router must be connected to another LAN. This can be done to hook up a group of friends temporarily or to section off different groups of computers. Whatever the reason, extending the router to other LAN networks should is straightforward. In the following examples, This article presumes that the *new* network is connected via a third ethernet card, namely `eth2`.

First configure the interface. Take the instructions in this section (/index.php?title=Hug_the_LAN_.28bring_along_some_friends.29&action=edit&redlink=1) and replace `eth0` with `eth2` and `192.168.0` with `192.168.1`.

Tweak dnsmasq to service the new interface. Edit the `/etc/conf.d/dnsmasq` file again and append `-i eth2` to `DNSMASQ_OPTS`; using -i multiple times is OK. Then edit `/etc/dnsmasq.conf` and add another line like the dhcp-range line in this section, replacing `192.168.0` with `192.168.1`. Having multiple dhcp-range lines is OK too.

Finally, see the rules in this section and duplicate the rules that have `-i ${LAN}` in them. Another variable may need to be created, say `LAN2`, to make things easier.

Troubleshooting

Useful tools

When having trouble getting computers to communicate try out the following tools (they can all be found in the net-analyzer Portage category):

Utility	Description
wireshark	GUI tool to view all raw network data according to filters
tcpdump	Console tool to dump all raw network data according to filters
iptraf	ncurses based IP LAN monitor
ettercap	ncurses based network monitor/control

DHCP fails to start

When starting the dhcp init.d script for the first time, it may fail to load but neglect to provide any useful information.

```
root # /etc/init.d/dhcp start
```

```
 * Setting ownership on dhcp.leases ...           [ ok ]
 * Starting dhcpd ...                             [ !! ]
```

The trick is used to know where dhcpd is sending its output. Browse to `/var/log` and read the log files. Since the exact log file depends on the package using a syslog, try running `grep -Rl dhcpd /var/log` to narrow down the possibilities. Chances are a typo was made in the configuration file. Another command to try running: `dhcpd -d -f` (short for debug / foreground). This aids in debugging the errors based upon the output.

Incorrect MTU value

If odd errors are experienced (such as not being able to access some webpages while others load fine), it might be Path MTU Discovery trouble. The quick way to test for this is to run the following iptables command:

```
root # iptables -A FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --clamp-mss-to-pmtu
```

This will affect all new connections; refresh the problematic website in order to test the fix. In case it helps, the standard MTU value for 100mbit ethernet connections is `1500` ; this value also applies to PPPoA. For PPPoE connections it is `1492` . For more info, read Chapter 15 of the Linux Advanced Routing & Traffic Control HOWTO (http://lartc.org/howto/).

If the above command does not work, consider putting the rule into the mangle table. Simply add `-t mangle` to the command.

Unable to connect two machines directly

If (for whatever reason) connecting two machines directly together without a hub or switch is required, a regular ethernet cable will likely not work, unless an Auto MDI/MDI-X (also known as "autosensing") capable network adapter is available. A different cable called a crossover cable will be needed for direct NIC to NIC connections. This Wikipedia (http://en.wikipedia.org/wiki/Ethernet_crossover_cable) page explains the low level details.

Final notes

There are no other final notes. If any troubles with this guide are experienced either update this article with the correct information or leave a brief message on this article's talk page with a summary of what is broken. Eventually someone should be able to correct any issue(s). It is also possible to file a bug on Gentoo's Bugtracking Website (https://bugs.gentoo.org/). If there are any other interesting bits that would enhance this guide, by all means include them! The worst that could happen is they could be removed.

This article is based on a document formerly found on our main website gentoo.org (http://www.gentoo.org/).
The following people have contributed to the original document: **Mike Frysinger (mailto:vapier@gentoo.org)**
They are listed here as the Wiki history does not provide for any attribution. If you edit the Wiki article, please do **not** add yourself here, your contributions are recorded on the history page.

Retrieved from "http://wiki.gentoo.org/index.php?title=Home_Router&oldid=341734 (http://wiki.gentoo.org/index.php?title=Home_Router&oldid=341734)"

Categories (/wiki/Special:Categories):
Documents containing Metadata (/wiki/Category:Documents_containing_Metadata)
| Server and Security (/wiki/Category:Server_and_Security)

- This page was last modified on 14 July 2015, at 09:02.