

HOWTO - IPv6 Introduction

IPv6 benefits

- IPv6 has longer addresses than IPv4 — 128 bits (16 bytes), instead of 32 bits (4 bytes) — and thus provides a huge supply of Internet addresses
- A simplified header of seven fields versus 13 in IPv4 to allow routers to process packets faster
- Improved options with the new header, thus routers get another speedup with skipping options not intended for them
- Better security with improvements in authentication and privacy (Not much difference while retrofitted to IPv4).
- full QOS (quality of service) - the sense of urgency is greater

Kernel IPv6 support

First, enable IPv6 support in your kernel: Linux Kernel Configuration: Enable IPv6 Device Drivers --->

Networking support --->

Networking options --->

<*> The IPv6 protocol (EXPERIMENTAL)

[*] IPv6: IPv6-in-IPv4 tunnel (SIT driver)

Then recompile your kernel and reboot. Connect to the IPv6 network

To connect to IPv6 network, you first need an IPv6 address. Few ISP's provide their customers with IPv6 addresses. Tunneling is a way to connect to the IPv6 network without support from ISP. There are a few tunneling techniques, tunnel brokers and 6to4 being the most useful for a home user.

You could get a tunnel broker, as described at [Gentoo IPv6 Router Guide](#), or use the anycast prefix defined in RFC 3068, which should connect you to your nearest 6to4 relay router.

Let's go for the latter.

NOTE: if you are behind a NAT router, you will also need to install Miredo (<http://www.remlab.net/miredo/>) to use anycast prefix. IPv6 tunneling with 6to4

Starting with baselayout-1.12.0_pre16 Gentoo provides 6to4 support without any need for external scripts or manual address calculation. The procedure further down the page still applies to baselayout-1.11.

To begin with, you need to come up with a name for the tunnel interface. If you have not installed iproute2, you have to use interface name sit0. Otherwise the name can be anything you want, Gentoo's own examples dictate using 6to4. The instructions below assume using 6to4 as a name. Replace it with sit0 if you are not using iproute2.

First, symlink net.lo to your new interface: `cd /etc/init.d/`

`ln -s net.lo net.6to4`

Then, edit your `/etc/conf.d/net` and add the following, where "eth0" is your desired host interface with a public IPv4 address. The 6to4 address is calculated automatically by the system. `link_6to4="eth0"`

```
config_6to4=( "ip6to4" )
```

```
depend_6to4() {
    need net.eth0
```

```
}
```

That's all there is to it. You can verify the functionality by issuing `/etc/init.d/net.6to4 start ifconfig 6to4`

and checking that you indeed have a 6to4 address (starting with 2002:).

The instructions below contain more details about the methods involved and still apply to earlier baselayouts.

Procedure for baselayouts up to 1.12.0_pre15:

In 6to4 tunneling, all 6to4 enabled hosts have a special address prefix to distinguish them from other IPv6 users. 6to4 enabled hosts/routers tunnel IPv6 packets to other 6to4 hosts directly. Packets to the rest of the IPv6 network are routed via a special 6to4 proxy. There are several of them all over the world, all of them having an anycast IPv4 address 192.88.99.1. So, when a router tunnels IPv6 packets over IPv4 via 192.88.99.1, the packets are transparently routed to the nearest 6to4 proxy router.

Address prefix is generated from a global IPv4 address, and it has the form `2002:<IPv4ADDR>::/48`. If your IPv4 address is 1.2.3.4, your IPv6 prefix is `2002:0102:0304::/48`. Note that the IPv6 address is written in hexadecimal. You may want

to have a look at [1] to get your IPv6 address calculated automatically.

The next 16 bits of the address are subnet part and the last 64 bits are the host part. It means that, with a single global IPv4 address, you can have $2^{16} = 65536$ subnets with 2^{64} hosts in each. You can choose whatever subnet and host address you want. For the sake of simplicity, we're using the first available address, `2002:<IPv4ADDR>::1`.

To create custom IPv6-over-IPv4 tunnels, you need `iproute2`, which is a network configuration suite that contains `ip`, the famous replacement for `ifconfig`, `route`, `iptunnel` and others. `emerge sys-apps/iproute2`

Now, let's configure the 6to4 tunnel:

```
# Create a tunnel device
ip tunnel add tun6to4 mode sit ttl 255 remote any local <your ipv4 address here>

# Bring the interface up
ip link set dev tun6to4 up

# Add local 6to4 address to interface
# Note the /16 prefix length. This way the packets to other 6to4 hosts get routed directly.
ip -6 addr add <your IPv6 address here>/16 dev tun6to4

# Add default route to the global IPv6 network using the 6to4 proxy anycast IPv4 address
ip -6 route add 2000::/3 via ::192.88.99.1 dev tun6to4 metric 1
```

We're done! Now you should be able to connect to the IPv6 network.

To bring down the tunnel, do the following:

```
# Flush 6to4 routes
ip -6 route flush dev tun6to4

# Bring down the tunnel device
ip link set dev tun6to4 down

# Remove the tunnel device
ip tunnel del tun6to4
```

To automate the 6to4 address generation and tunnel set up on startup, add the following lines to your `/etc/conf.d/net`.

```
postdown() {
if [ "${IFACE}" == eth0 ] ; then
if ip -6 route flush dev tun6to4 &&
ip link set dev tun6to4 down &&
ip tunnel del tun6to4; then
einfo "6to4 tunnel brought down"
else
ewarn "Failed to bring down 6to4 tunnel"
fi
fi
return 0
}

postup() {
if [ "${IFACE}" == eth0 ] ; then
ipv4addr=$(ip addr show dev eth0 | grep "inet "
| sed -e 's/\s*inet \([0-9]*\.[0-9]*\.[0-9]*\.[0-9]*\) .*$/\1/'
| tail -n1)
ipv6addr=$(printf "2002:%02x%02x:%02x%02x::1" ${ipv4addr//./})

if ip tunnel add tun6to4 mode sit ttl 255 remote any local $ipv4addr &&
ip link set dev tun6to4 up &&
ip -6 addr add $ipv6addr/16 dev tun6to4 &&
ip -6 route add 2000::/3 via ::192.88.99.1 dev tun6to4 metric 1; then
einfo "6to4 tunnel set up"
else
```

```
ewarn "Setting up 6to4 tunnel failed"  
fi  
fi  
return 0  
}
```

Note that the script assumes your network device connected to the Internet is eth0. If it's another device, change the references to eth0 according to your configuration. Let your apps use IPv6

- Add ipv6 USE flag to your /etc/make.conf file.
- Then you will have to re-emerge some applications: emerge --newuse world

Some applications need explicit configuration to use IPv6, while some use it automatically. For IPv6 enabled Gentoo mirrors, see <http://www.gentoo.org/main/en/mirrors.xml>, and look for URLs ending in an asterisk (*). To configure wget prefer IPv6 over IPv4 when fetching files from Gentoo mirror, add the following lines in your /etc/make.conf.

```
# Make wget prefer IPv6  
FETCHCOMMAND="/usr/bin/wget --prefer-family=IPv6 -t 5 --passive-ftp \${URI} -P \${DISTDIR}"  
RESUME  
COMMAND="/usr/bin/wget -c --prefer-family=IPv6 -t 5 --passive-ftp \${URI} -P \${DISTDIR}"
```