

Benedikt Stockebrand

# IPv6 in Practice

A Unixer's Guide to the  
Next Generation Internet

**NetBSD 3.1/OpenBSD 4.1  
Supplement**

Version 1.0 (2007-05-24)



**1 p. VIII**

**NetBSD 3.1, OpenBSD 4.1** All the BSDs use the KAME IPv6 implementation. While the integration into the boot scripts may be considered less elegant than with FreeBSD 6.1, most if not all features of the KAME stack are available with the other BSDs, too. So are its limitations.

Beyond the TCP/IP stack itself, IPv6 support in various programs occasionally lags behind.

**2 p. X**

**NetBSD 3.1** is the BSD that supports virtually all hardware platforms available today.

**OpenBSD 4.1** focuses particularly on security. Both the most commonly used SSH implementation and the `pf` packet filter were originally developed for OpenBSD.

**3 p. 11**

**NetBSD 3.1, OpenBSD 4.1** by default install plain vanilla `GENERIC` kernels which support IPv6 without problems.

**4 p. 11**

**NetBSD 3.1, OpenBSD 4.1** both install without the `bash` I use throughout the book, but for both it is available as a port or package from the 2007Q1 `pkgsrc` repository (for NetBSD 3.1) or the ports/packages shipped with OpenBSD 4.1.

**5 p. 11/12**

**NetBSD 3.1, OpenBSD 4.1** both install the man pages and an up-to-date `Whatis` index even with a standard installation.

## 6 p. 14

**NetBSD 3.1, OpenBSD 4.1** Just like FreeBSD 6.1, the kernels already support IPv6 and the loopback interface has IPv6 support enabled by default. The `ifconfig` syntax to enable IPv6 on a physical interface is also the same except that the interface names may vary.

## 7 p. 14

**NetBSD 3.1** To enable IPv6 permanently on a physical interface `wm0` we need to create a file `/etc/ifconfig.wm0` with the single line of content

```
/etc/ifconfig.wm0
```

```
up
```

**OpenBSD 4.1** Similar to NetBSD 3.1 we configure a physical interface `em0` with a file named `/etc/hostname.em0`. Again, all it must contain is the keyword `up`:

```
/etc/hostname.em0
```

```
up
```

## 8 p. 15

**NetBSD 3.1, OpenBSD 4.1** Similar to FreeBSD 6.1, adding the line

```
/usr/src/sys/arch/i386/conf/CUSTOM
```

```
options INET6
```

to a kernel configuration file `/usr/src/sys/arch/i386/conf/CUSTOM` re-enables IPv6 support.

## 9 p. 15

**NetBSD 3.1, OpenBSD 4.1** The `sysctl` interface with FreeBSD 6.1 also works with the other BSDs.

## 10 p. 16/17

**NetBSD 3.1, OpenBSD 4.1** both also support the *pf* packet filter, so the rules shown for FreeBSD 6.1 can be used after a few trivial modifications.

*NetBSD 3.1* also ships with Darren Reed's *ipf* packet filter suite, which by now also supports IPv6. But *pf* is widely considered more state of the art, so we won't address *ipf* and its IPv6 support.

## 11 p. 17–19

**NetBSD 3.1** To enable the *pf* filter at boot time we need to add the lines

```
/etc/rc.conf
```

```
lkm=YES
pf=YES
```

to */etc/rc.conf* and the line

```
/etc/lkm.conf
```

```
pf.o      -      -      -      -      -
```

to */etc/lkm.conf*. This will load the *pf* kernel module at boot time. Alternatively we could build a kernel with the *pf* pseudo device compiled in. The line *pf=YES* in *rc.conf* also causes the configuration from */etc/pf.conf* to be loaded at boot time.

Except for the interface name in the first line and a typo in the *antispoof* rule, the configuration file */etc/pf.conf* from FreeBSD 6.1 can be copied verbatim to a NetBSD 3.1 setup; in the *antispoof* line, add the keyword *for* after the *quick* keyword.

The commands

```
# modload /usr/lkm/pf.o           || Load the kernel module
# pfctl -f /etc/pf.conf           || Load filter rules
# pfctl -e                        || Enable filter
```

temporarily install the *pf* filter module, load the filter rules and enable the filter. As with FreeBSD 6.1, the command

```
# pfctl -s rules
```

displays the currently installed rules.

**OpenBSD 4.1** The *GENERIC* kernel shipping with OpenBSD 4.1 already contains the *pf* filter, so there is no need to load a kernel module. Otherwise OpenBSD 4.1 behaves as NetBSD 3.1, so everything short of the kernel module related aspects there applies to OpenBSD 4.1 as well. Only the *pf=YES* line belongs in */etc/rc.conf.local* rather than */etc/rc.conf*.

12 p. 24

**NetBSD 3.1** There is a complete *ipv6calc* port/package available including the manual page.

**OpenBSD 4.1** Like FreeBSD 6.1 there is an *ipv6calc* port/package available which only contains the binaries but no man pages.

14 p. 26

**NetBSD 3.1, OpenBSD 4.1** Unsurprisingly, all BSDs behave the same, so just like FreeBSD 6.1 they all set up a link-local address on the loopback interface.

15 p. 26

**NetBSD 3.1, OpenBSD 4.1** Again, all the BSDs show the same behaviour and their *ping6* commands support both the *-I* *<interface>* option as well as the percent sign notation.

16 p. 36

**NetBSD 3.1, OpenBSD 4.1** Yet again, the *ifconfig* command included with all the BSDs behaves exactly as with FreeBSD 6.1 as far as the configuration of IPv6 addresses is concerned.

## 17 p. 37/38

**NetBSD 3.1, OpenBSD 4.1** For a change, the output format of the `ifconfig` command differs between the BSDs. As far as the address configuration and the associated command line options are concerned they all behave the same, though.

## 18 p. 38

**NetBSD 3.1, OpenBSD 4.1** Since adding addresses with `ifconfig` works the same with all the BSDs it doesn't surprise that removing them again also works the same.

## 19 p.39

**NetBSD 3.1** The persistent address configuration for an interface `wm0` is kept in `/etc/ifconfig.wm0`. The format

```
/etc/ifconfig.wm0
```

```
inet6 2001:db8:fedc:abcd::4
inet6 2001:db8:fedc:cdef::4
```

works as expected despite the fact that the `ifconfig.if(5)` man page shows only more complex examples.

**OpenBSD 4.1** The only difference between NetBSD 3.1 and OpenBSD 4.1 is the name of the configuration file; for an interface `em0` the file name OpenBSD 4.1 uses is `/etc/hostname.em0` and the associated man page is `hostname.if(5)`. In every other respect both NetBSD 3.1 and OpenBSD 4.1 behave exactly the same at least at this point..

## 20 p. 41

**NetBSD 3.1, OpenBSD 4.1** All the BSDs use the same `ndp` command shown with FreeBSD 6.1, so `ndp -a` works here as well.

**NetBSD 3.1** First we must configure the network interface again. In addition to the unicast address we must explicitly configure the subnet router anycast addresses, too:

```
/etc/ifconfig.wm0
```

```
inet6 2001:db8:fedc:abcd::4
inet6 2001:db8:fedc:abcd:: anycast
inet6 2001:db8:fedc:cdef::4
inet6 2001:db8:fedc:cdef:: anycast
```

Next we must tell the router that it is a router, i.e. it forwards packets, and that we want it to run the router advertisement daemon `rtadvd` on interface `wm0`. To do so we add the lines

```
/etc/rc.conf
```

```
ip6mode=router           || Enable forwarding
rtadvd=YES               || Start rtadvd
rtadvd_flags="wm0"      || On the wm0 interface
```

to `/etc/rc.conf`. After a quick reboot the router should work as expected.

**OpenBSD 4.1** Similar to NetBSD 3.1 the first step is to configure the subnet router anycast addresses. Different than the NetBSD 3.1 we must also add an explicit prefix length of 64 here:

```
/etc/ifconfig.em0
```

```
inet6 2001:db8:fedc:abcd::5
inet6 2001:db8:fedc:abcd:: 64 anycast
inet6 2001:db8:fedc:cdef::5
inet6 2001:db8:fedc:cdef:: 64 anycast
```

To enable IPv6 packet forwarding in the kernel we next uncomment the line

```
/etc/sysctl.conf
```

```
net.inet6.ip6.forwarding=1
```

in `/etc/sysctl.conf`.

Finally we start the `rtadvd` by changing the `rtadvd_flags` variable in `/etc/rc.conf.local` to contain the interface or interfaces we want to run the `rtadvd` to run on:

`/etc/rc.conf.local`

```
rtadvd_flags="em0"
```

As usual, after a quick reboot the router should start to advertise its prefixes.

22 p. 49

**NetBSD 3.1** First we tell the boot scripts that we are willing to accept router advertisements for interface configurations. To do so we set the variable `ip6mode` in `/etc/rc.conf` accordingly:

`/etc/rc.conf`

```
ip6mode=autohost
```

Additionally we must bring up the interface we want to use and run the `rtsol` command on it. For this reason we need the lines

`/etc/ifconfig.wm0`

```
up                                || Bring up the interface first
!rtsol $int                       || ... and then do a router solicitation
```

in the interface configuration file `/etc/ifconfig.wm0`. After the next reboot the host should configure itself using autoconfiguration.

**OpenBSD 4.1** To configure a node as a host we first set the variables

`/etc/sysctl.conf`

```
# net.inet6.ip6.forwarding=1    || Disable forwarding
net.inet6.ip6.accept_rtadv=1   || Accept router advertisements
```

in `/etc/sysctl.conf`. Then we ensure that an interface `em0` does a router solicitation when it comes up by putting the single line

`/etc/hostname.em0`

```
rtsol
```

in the interface configuration file `/etc/hostname.em0`. After the next reboot the host should be properly configured through autoconfiguration.

**NetBSD 3.1** By default, NetBSD 3.1 mixes static and autoconfigured addresses. The line

```
/etc/rc.conf
```

```
ip6mode=host
```

in `/etc/rc.conf` suppresses autoconfiguration. There is no documented way to suppress address configuration but keep default route configuration, so on a statically configured host the default route must be set manually.

**OpenBSD 4.1** As soon as an interface has a statically configured address with a given prefix, autoconfiguration won’t add a dynamically configured address with that prefix.

There doesn’t seem to be a way to configure OpenBSD 4.1 to behave differently; neither is the default behaviour documented.

**NetBSD 3.1** As with FreeBSD 6.1 the `-L` option of `ifconfig` displays the lifetimes.

**OpenBSD 4.1** Even without an explicit option, `ifconfig` always displays the lifetimes.

**NetBSD 3.1, OpenBSD 4.1** The `pf` packet filter on all BSDs behaves as explained for FreeBSD 6.1.

**NetBSD 3.1, OpenBSD 4.1** The `pf` packet filter on all BSDs behaves as explained for FreeBSD 6.1.

27 p. 56/57

**NetBSD 3.1, OpenBSD 4.1** *Except for the dynamic interface-to-address resolution bug with FreeBSD 6.1, the pf packet filter on all BSDs behaves as explained for FreeBSD 6.1.*

28 p. 57–63

**NetBSD 3.1, OpenBSD 4.1** *The pf.conf shown for FreeBSD 6.1 also works with the other BSDs; just change the interface names at the beginning macros accordingly.*

31 p. 68

**NetBSD 3.1, OpenBSD 4.1** *Both ping6 and traceroute6 behave the same as on FreeBSD 6.1.*

32 p. 69

**NetBSD 3.1** *The resolver uses the same /etc/nsswitch.conf as Debian Sarge and FreeBSD 6.1. The default settings already make the resolver search the local /etc/hosts file and then the DNS.*

**OpenBSD 4.1** *The default behaviour of this resolver is to search in the DNS first and then in /etc/hosts. To change this, for example to search the local file first, we must add a line like*

```
/etc/resolv.conf
```

```
lookup file bind
```

*to /etc/hosts; a /etc/nsswitch.conf file isn't supported.*

33 p. 78

**NetBSD 3.1, OpenBSD 4.1** *Since all the BSDs use the pf packet filter, they also behave all like FreeBSD 6.1.*

34 p. 82

**NetBSD 3.1, OpenBSD 4.1** *The `inetd` daemon from the base installation supports IPv6.*

35 p. 83–85

**NetBSD 3.1, OpenBSD 4.1** *Different than FreeBSD 6.1, these two BSDs don’t support the `tcp46` and `udp46` keywords; to start a service for both IPv4 and IPv6 it must use separate configuration lines for IPv4 and IPv6. The `tcp` and `udp` keywords mean IPv4-only.*

36 p. 86/87

**NetBSD 3.1** *uses the same options to `netstat` as FreeBSD 6.1 and also comes with the `sockstat` command.*

**OpenBSD 4.1** *uses the same options to `netstat` as FreeBSD 6.1 but doesn’t ship with the `sockstat` command.*

37 p. 88

**NetBSD 3.1, OpenBSD 4.1** *both ship with an IPv6-capable OpenSSH client and server in the core distribution.*

38 p. 90

**NetBSD 3.1** *The standard NTP implementation shipped with the core system supports IPv6 just fine.*

**OpenBSD 4.1** *The NTP implementation here is not based on the University of Delaware code base, uses a different configuration file (and lacks administration tools like `ntpd` and `ntpq`). It seems to support IPv6 without problems.*

**39 p. 91**

**NetBSD 3.1** *The syslogd implementation supports IPv6 but lacks the -a, -b, -4 and -6 options of its FreeBSD 6.1 peer.*

**OpenBSD 4.1** *This syslogd implementation here doesn't support IPv6.*

**41 p. 92**

**NetBSD 3.1, OpenBSD 4.1** *The Sendmail versions included in the core installations support IPv6.*

**42 p. 94**

**NetBSD 3.1** *All web browsers mentioned support IPv6 except for Lynx, which supports hostnames in URLs that resolve to IPv6 addresses but no addresses in URLs, and Links, which still doesn't support IPv6 at all.*

**OpenBSD 4.1** *Besides the problems with Lynx and Links, Konqueror doesn't support addresses in URLs. All other browsers work without problems.*

**43 p. 94**

**OpenBSD 4.1** *The Apache 1.3.29 web server shipping with OpenBSD doesn't support IPv6, so it is necessary to install the Apache2 package. The main configuration file is /etc/apache2/httpd2.conf.*

**44 p. 95**

**NetBSD 3.1, OpenBSD 4.1** *The Apache2 packages of both BSDs have been built without IPv4-mapped IPv6 address support.*

45 p. 96

**NetBSD 3.1, OpenBSD 4.1** *The proxy configuration works out of the box.*

47 p. 96

**NetBSD 3.1** *The `ffproxy` package from the `ports/packages` collection expects its configuration in `/usr/pkg/etc/ffproxy.conf` and its access control configuration in `/usr/pkg/share/ffproxy/db/access.ip`.*

**OpenBSD 4.1** *The `ffproxy` package from the `ports/packages` collection expects its configuration in `/etc/ffproxy.conf` and its access control configuration in `/var/ffproxy/db/access.ip`.*

48 p. 97

**NetBSD 3.1** *Both the `rpcbind` portmapper and the NFS implementation included with NetBSD 3.1 support IPv6.*

**OpenBSD 4.1** *There is no IPv6-capable portmapper shipping with OpenBSD 4.1.*

49 p. 97

**NetBSD 3.1** *The access control features described for FreeBSD 6.1 are also supported by NetBSD 3.1.*

50 p. 98

**NetBSD 3.1** *The option `-noresvmt` in `/etc/exports` tells the NFS server that clients don't have to use a low port to access a share.*

52 p. 99/100

**NetBSD 3.1, OpenBSD 4.1** *The same considerations as for FreeBSD 6.1 also apply to the other BSDs.*

53 p. 101

**NetBSD 3.1, OpenBSD 4.1** *The same considerations as for FreeBSD 6.1 also apply to the other BSDs.*

54 p. 101

**NetBSD 3.1, OpenBSD 4.1** *The same considerations as for FreeBSD 6.1 also apply to the other BSDs.*

55 p. 106–108

**NetBSD 3.1** *With regard to static routes the `netstat` and `route` commands use the same syntax as their FreeBSD 6.1 counterparts.*

*A default router can be given either in the `defaultroute6` variable in `/etc/rc.conf` or in the file `/etc/mygate6`.*

**OpenBSD 4.1** *With regard to static routes the `netstat` and `route` commands use the same syntax as their FreeBSD 6.1 counterparts.*

**NetBSD 3.1, OpenBSD 4.1** *There is no explicit support for general static routes in the boot scripts, which leaves us only with the option to add the appropriate `route` invocations to `/etc/rc.local`.*

*In certain situations this is too late, for example if a service that needs access to the name server starts before `/etc/rc.local` is run and the name server is only reachable through that static route. In this case it may be necessary to change the boot scripts accordingly.*

56 p. 109

**NetBSD 3.1** *To enable the `route6d` RIPng router included with a base installation, we just add the line*

```
/etc/rc.conf
```

```
route6d=YES
```

*to `/etc/rc.conf` and either reboot the router or run the command `/etc/rc.d/route6d start` by hand.*

**OpenBSD 4.1** To enable the `route6d` RIPng router included with a base installation, we just add the line

```
/etc/rc.conf.local
```

```
route6d_flags=""
```

to `/etc/rc.conf.local` and either reboot the router or start the routing daemon by invoking `route6d` by hand.

57 p. 110/111

**NetBSD 3.1, OpenBSD 4.1** According to the `route6d(8)` man page the `route6d` supports the `-R` option mentioned with FreeBSD 6.1. Tests show however that this feature doesn’t currently work.

58 p. 119

**NetBSD 3.1, OpenBSD 4.1** As with FreeBSD 6.1 we need to start the `route6d` with the additional option `-s` to make it announce the static routes it finds in the kernel. Adding it to the configuration variable `route6d_flags` in `/etc/rc.conf` (NetBSD 3.1) or `/etc/rc.conf.local` (OpenBSD 4.1) is all it takes.

59 p. 121/122

**NetBSD 3.1, OpenBSD 4.1** The FreeBSD 6.1 configuration can be used with the other BSDs without modification.

60 p. 122/123

**NetBSD 3.1, OpenBSD 4.1** The FreeBSD 6.1 configuration can be used with the other BSDs without modification.

62 p. 129

**NetBSD 3.1, OpenBSD 4.1** As with FreeBSD 6.1, the `pf` filter applies all rules to both IPv4 and IPv6 unless we specify the protocol family with the `inet` or `inet6` keyword.

**NetBSD 3.1, OpenBSD 4.1** The *GENERIC* kernel doesn't include the `faith` pseudo-device, so we first build and install a new kernel with the option

```
/usr/src/sys/arch/i386/conf/CUSTOM
```

```
pseudo-device faith
```

As with *FreeBSD 6.1* we configure the translator machine with an IPv4 address of `192.0.2.3` and an IPv6 address `2001:db8:fedc:6666::1` and enable the router advertisement daemon as usual.

The boot scripts don't have support for the `faith` configuration, so we have to take care of everything by ourselves from the `/etc/rc.local` script:

```
/etc/rc.local
```

```
sysctl -w net.inet6.ip6.keepfaith=1
ifconfig faith0 create
ifconfig faith0 up
route add -inet6 2001:db8:fedc:4444:4444:4444:: \
    -prefixlen 96 ::1
route change -inet6 2001:db8:fedc:4444:4444:4444:: \
    -prefixlen 96 -ifp faith0
faithd ssh /usr/sbin/sshd sshd -i
```

The first line effectively enables `faith` support in the kernel; we could have put this part of the configuration in `/etc/sysctl.conf`, but I personally prefer to keep everything in one place.

The second and third line create and enable the interface.

The two `route` commands set up the routing so all traffic to the TRT prefix is routed through the `faith` interface.

Finally, the last line starts the userland `faithd` daemon which rewrites the IP headers of all traffic passing through the `faith` interface on the Ssh port. Alternatively we can run the `faithd` daemon through the `inetd` as the *FreeBSD 6.1* example shows.

After a quick reboot the check list shown with *FreeBSD 6.1* on page 138 should apply fine.

Next we install the `totd` package and create the `totd.conf` file. It should contain the same lines as the one for *FreeBSD 6.1* but it must be

saved in `/usr/pkg/etc` on *NetBSD 3.1* and in `/etc` on *OpenBSD 4.1*. To start the `totd` we add a final line

```
/etc/rc.local
```

<code>/usr/pkg/sbin/totd</code>	<i>NetBSD 3.1</i>
<code>/usr/local/sbin/totd</code>	<i>OpenBSD 4.1</i>

to `/etc/rc.local`. After yet another reboot the translator should be up and running. With the resolver configuration from page 139 on the client everything should work fine and the second check list, on page 139, shouldn’t discover any problems.

64 p. 140

**NetBSD 3.1, OpenBSD 4.1** Unsurprisingly, the same reasoning as for *FreeBSD 6.1* applies.

65 p. 151

**NetBSD 3.1, OpenBSD 4.1** All the BSDs show the same behaviour as documented for *FreeBSD 6.1*.

66 p. 152–155

**NetBSD 3.1, OpenBSD 4.1** Configuring a tunnel manually with the `ifconfig` command works the same as with *FreeBSD 6.1*. The boot scripts don’t explicitly support tunnel interfaces, but we can use the “!” feature in the standard `/etc/ifconfig.gif0` (*NetBSD 3.1*) or `/etc/hostname.gif0` (*OpenBSD 4.1*) files to run `ifconfig`. For the left tunnel router from figure 12.3 the configuration may look like this:

```
/etc/{ifconfig|hostname}.gif0
```

<code>!ifconfig gif0 tunnel 192.0.2.1 192.0.2.2 up</code>	
<code>!ifconfig gif0 inet6 2001:db8:fedc:4646::1 up</code>	<i>Optional</i>

To configure the tunnel interface accordingly it is easiest to reboot the machine.

The link-local address problem can be solved with the same script shown for *FreeBSD 6.1*. Only for *OpenBSD 4.1* the “`localaddr=[...]`”

line must be slightly modified: replace the string “`tunnel`” from the book with “`physical`”.

67 p. 156/157

**NetBSD 3.1, OpenBSD 4.1** The `route` command doesn’t support the `-iface` option with IPv6 the way it does with FreeBSD 6.1. This forces us to specify the IPv6 address of the remote tunnel endpoint instead. Additionally, the OpenBSD 4.1 implementation doesn’t support the slash-prefixlength notation, either, so we have to use the `-prefixlen` option instead.

Since the boot scripts don’t support static routes properly we must run the `route` command either from the `/etc/rc.local` script or using the “!” notation from the `/etc/ifconfig.gif0` (NetBSD 3.1) or `/etc/hostname.gif0` (OpenBSD 4.1) files.

68 p. 158

**NetBSD 3.1, OpenBSD 4.1** As with FreeBSD 6.1, automatic tunnels are not supported.

69 p. 160/161

**NetBSD 3.1** The default `GENERIC` kernel doesn’t support `6to4`, so we have to build, install and boot a custom kernel with the additional option

```
/usr/src/sys/arch/i386/conf/CUSTOM
```

```
pseudo-device stf
```

before we can use `6to4` tunnels.

The `ifconfig` syntax from FreeBSD 6.1 applies here, too. The boot scripts don’t support `6to4` explicitly, so we can either set up the interface from `/etc/rc.local` or use the “!” notation in an interface configuration file again.

As with FreeBSD 6.1, only a single `stf` interface is supported.

**OpenBSD 4.1** *The OpenBSD community has apparently decided that 6to4 is too much of a security risk and ripped the code out of the KAME stack. This makes it effectively impossible to use 6to4 tunnels with OpenBSD 4.1.*

70 p. 164/165

**NetBSD 3.1** *The same route invocation shown with FreeBSD 4.1 also works here. Since there is no explicit support for 6to4 tunnels in the boot scripts we have to add a permanent configuration through the “!” notation in an interface configuration file or in /etc/rc.local again.*

72 p. 166/167

**NetBSD 3.1** *Again we have to set up the configuration manually, using the command*

```
# route add -inet6 default 2002:c058:6301::
```

*For a permanent configuration, we have to invoke that command during boot as explained above.*

73 p. 170/171

**NetBSD 3.1** *The same ifconfig invocations shown for FreeBSD 6.1 also work with NetBSD 3.1. Note however that in the first line the keyword inet6 is missing between create and tunnel.*

*The implementation shows an annoying problem: Pings don’t work across the tunnel. Possibly other ICMP packets are also affected.*



**OpenBSD 4.1** *The ifconfig command can’t create an interface and configure the tunnel in a single invocation. For this reason it is necessary to do three ifconfig invocations, like this:*

```
# ifconfig gif0 create
# ifconfig gif0 inet6 tunnel 2001:db8:fedc:1::1 \
                           2001:db8:fedc:1::2 up
# ifconfig gif0 10.0.0.1 10.0.0.2
```

74 p. 172

**NetBSD 3.1, OpenBSD 4.1** *The same syntax as with FreeBSD 6.1 also applies here.*

75 p. 173/174

**NetBSD 3.1, OpenBSD 4.1** *The same `ifconfig` invocations shown for FreeBSD 6.1 also work with the other BSDs.*

76 p. 174

**NetBSD 3.1, OpenBSD 4.1** *The `-iface` option doesn't work here, so we need to specify the address of the tunnel peer instead:*

```
# route add -inet6 2001:db8:fedc:4::/64 2001:db8:fedc:66:2
```

78 p. 175

**NetBSD 3.1, OpenBSD 4.1** *Like FreeBSD 6.1, the BSDs all automatically configure a link-local address on the tunnel interface.*

79 p. 179

**NetBSD 3.1, OpenBSD 4.1** *The `pf` filter supports the same filtering criteria as with FreeBSD 6.1.*

80 p. 181/182

**NetBSD 3.1** *Since the `ifconfig` command doesn't support the slash notation for prefix lengths, we need the slightly modified commands*

```
# ifconfig gre0 create tunnel 192.0.2.1 192.0.2.129
# ifconfig gre0 inet6 2001:db8:fedc:6666::1 prefixlen 128 \
2001:db8:fedc:6666::2 up
```

*to bring up the tunnel.*

**OpenBSD 4.1** Since the `ifconfig` command doesn’t support creating and configuring an interface in one go or the slash notation for prefix lengths, the commands to bring up a `gre` interface are

```
# ifconfig gre0 create
# ifconfig gre0 tunnel 192.0.2.133 192.0.2.4
# ifconfig gre0 inet6 2001:db8:fedc:6666::5 prefixlen 128 \
                    2001:db8:fedc:6666::4 up
# sysctl -w net.inet.gre.allow=1
```

For security reasons *OpenBSD 4.1* disables `gre` interfaces by default. The last line explicitly enables them through a `sysctl` variable.

81 p. 184

**NetBSD 3.1** While *NetBSD 3.1* does ship with a fairly recent version of *OpenVPN* in its `ports/packages` collection, its `tun` tunnel interface implementation doesn’t support IPv6, so IPv6 through *OpenVPN* tunnels doesn’t work at this time.

**OpenBSD 4.1** The `ports/packages` collection includes a reasonably up-to-date *OpenVPN* version which we can install without complications. To enable the `openvpn` daemon, the easiest approach is to add a line

```
/etc/rc.local
```

```
/usr/local/sbin/openvpn --config /etc/openvpn.conf
```

in `/etc/rc.local`. When the machine boots, it will automatically start the `openvpn` daemon. The configuration file is `/etc/openvpn.conf` in this case.

82 p. 186

**OpenBSD 4.1** A problem which both *Debian Sarge* and *OpenVPN 4.1* have in common is that they don’t configure a link-local address when they bring a tunnel interface up. To work around this we need to configure an explicit link-local address on the tunnel interface. The equivalent to the *Debian Sarge* script looks like this:

```
/etc/openvpn.up
```

```
#!/bin/sh

localaddr="'ifconfig -a \
    | sed '/inet /!d;/127\.0\.0\.1/d;s/.*inet //;s/ .*//' \
    | head -n 1'"

/sbin/ifconfig $dev up
/sbin/ifconfig $dev inet6 fe80::$localaddr
/sbin/route add -inet6 default -iface fe80::$localaddr%$dev
```

In the last line note how the `-iface` option with OpenBSD 4.1 requires not an interface name but an address with an interface qualifier.

83 p. 192/193

**NetBSD 3.1, OpenBSD 4.1** The `pf`-based reverse NAT configuration also works with these BSDs.

85 p. 194

**NetBSD 3.1, OpenBSD 4.1** There is no way to configure the nesting level as with FreeBSD 6.1.

86 p. 196

**NetBSD 3.1, OpenBSD 4.1** The same `ifconfig` syntax as with FreeBSD 6.1 and Solaris 10 also applies here.

87 p. 196/197

**NetBSD 3.1** For `gif` interfaces the two `sysctl` variables shown for FreeBSD 6.1 also exist.

For `gre` or `6to4` interfaces there is no way to set the hop limit.

**OpenBSD 4.1** There is no documented way to set the hop limit for any of the tunnel interfaces.

**NetBSD 3.1** There is an IPv6-enabled version of the `pppd` shipping with the core system. The port/package `userppp` of the userland PPP implementation doesn't currently support IPv6.

**OpenBSD 4.1** The `pppd` shipping with the OpenBSD 4.1 core system doesn't support IPv6, but the userland PPP implementation is also included in the core system and does support IPv6.

**NetBSD 3.1** The boot scripts expect a list of peer names in the `ppp_peers` variable in `/etc/rc.conf`, like

```
/etc/rc.conf
```

```
ppp_peers="peer1 peer2"
```

and the configuration for each such peer in an accordingly named separate file `/etc/ppp/peers/peer1` with the minimalistic contents

```
/etc/ppp/peers/peer1
```

```
lock
persist
noauth
ipv6 ,
noip
noccp
```

as we have already seen for the other Unixen running the `pppd` daemon in the book.

Unfortunately, at least in my VMware test setup NetBSD 3.1 shows the same problem with the `persist` option explained for Debian Sarge.



As a workaround I add the `nodetach` option to the configuration file again, remove the `ppp_peers` variable from `/etc/rc.conf` again and add a line

```
/etc/ttys
```

```
tty01 "/usr/sbin/pppd call peer1" unknown on
[...]
```

to `/etc/ttys`. After a quick reboot or invoking `pkill -HUP 1` the `init` process takes care of restarting the `pppd` daemon for me.

**OpenBSD 4.1** The configuration file `/etc/ppp/ppp.conf` looks exactly the same as the one shown for FreeBSD 6.1 with the single exception that the serial devices are called differently: Instead of `/etc/cua0` the one for the first serial interface is `/etc/cua00` and so on.

The boot scripts don't have any explicit support for running the `ppp` daemon. For this reason even a single PPP connection requires a line like

```
/etc/rc.local
```

```
/usr/sbin/ppp -dedicated serial0
```

in `/etc/rc.local` for every single PPP interface. After a reboot, or after issuing that same command on the command line, the PPP connection should start up.

90 p. 202–204

**NetBSD 3.1** Like on Solaris 10 we can add routable addresses and static routes to a PPP interface using the `/etc/ppp/ipv6-up` file. All we have to do is to adapt the exact invocations of `ifconfig` and `route` to the NetBSD 3.1 syntax.

```
/etc/ppp/ipv6-up
```

```
#!/bin/sh
case $1 in
ppp0) /sbin/ifconfig ppp0 inet6 2001:db8:fedc:13::3
      /sbin/route add -inet6 2001:db8:fedc:4::/64 \
                    -iface 2001:db8:fedc:13::3
;;
[... ]
esac
```

Again, this script receives the interface name as its first parameter.

Different than the Solaris 10 implementation, all `ppp` interfaces are automatically released when the `pppd` daemon shuts down. In most cases it is therefore unnecessary to write an explicit `/etc/ppp/ipv6-down` script.

**OpenBSD 4.1** Similar to the FreeBSD 6.1 configuration we first add a file `/etc/ppp/ppp.linkup` with the contents

```
/etc/ppp/ppp.linkup
```

```
MYADDR6: shell /etc/ppp/ipv6-up LABEL INTERFACE MYADDR6
```

which differs from the FreeBSD 6.1 version in that it adds a third parameter to the `/etc/ppp/ipv6-up` script invocation which holds the local IPv6 address of the interface. We need this extra parameter because the `route` command on OpenBSD 4.1 doesn’t let us specify a route through a point-to-point interface by the interface name alone. The script should look something like this:

```
/etc/ppp/ipv6-up
```

```
#!/bin/sh
case $1 in
serial0)
    /sbin/ifconfig $2 inet6 2001:db8:fedc:13::4 up
    /sbin/route add -inet6 2001:db8:fedc:3:: -prefixlen 64 \
        -iface $3$2
    ;;
[...])
esac
```

After a reboot or a manual restart of the `ppp` daemon the routable address and static route should be properly configured.

92 p. 205/206

**NetBSD 3.1, OpenBSD 4.1** The `rtadvd` behaves the same as on FreeBSD 6.1.

93 p. 207

**NetBSD 3.1** The same problem as with Debian Sarge and Solaris 10 also applies to the `pppd` daemon on NetBSD 3.1.

**OpenBSD 4.1** We can use the same extension as on FreeBSD 6.1 to configure an interface by the user connecting to it.

**94** p. 207

**NetBSD 3.1** We can use the `/etc/ppp/ipv6-down` script mentioned with Debian Sarge.

**OpenBSD 4.1** We can use the `/etc/ppp/ppp.linkdown` script mentioned with FreeBSD 6.1 on OpenBSD 4.1 as well.

**95** p. 213

**NetBSD 3.1, OpenBSD 4.1** The `-0` option shown with FreeBSD 6.1 is also available with the other BSDs.

**96** p. 215/216

**NetBSD 3.1, OpenBSD 4.1** On all the BSDs the `sysctl` variable `net.inet.ip6.v6only` controls the use of mapped addresses. By default they all set it to 1 according to the standard.

**97** p. 218/219

**NetBSD 3.1, OpenBSD 4.1** There doesn't seem to be support for temporary addresses available.

**98** p. 220

**NetBSD 3.1, OpenBSD 4.1** There doesn't seem to be support for temporary addresses available.

**99** p. 223

**NetBSD 3.1, OpenBSD 4.1** There is no configurable address selection policy table available.

100 p. 224/225

**NetBSD 3.1, OpenBSD 4.1** *The /etc/radvd.conf configuration from FreeBSD 6.1 also applies here.*

101 p. 225/226

**NetBSD 3.1, OpenBSD 4.1** *The /etc/radvd.conf configuration from FreeBSD 6.1 also applies here.*

102 p. 227/228

**NetBSD 3.1, OpenBSD 4.1** *The /etc/radvd.conf configuration from FreeBSD 6.1 also applies here.*

103 p. 228-230

**NetBSD 3.1, OpenBSD 4.1** *The /etc/radvd.conf configuration from FreeBSD 6.1 also applies here.*

104 p. 236

**NetBSD 3.1** *Quagga is readily available from the ports/packages collections. Unfortunately it shows an annoying bug: It doesn’t save the interface address configuration in its configuration file if we write the running configuration to the file.*

**OpenBSD 4.1** *A seriously outdated version of Quagga is available from the ports/packages collection. It doesn’t support router advertisements, so we’ll have to continue using radvd for this purpose. Its RIPng and OSPFv3 support are apparently broken, too. Building a more recent version (0.98.6) from sources also fails.*

*For these reasons Quagga and OpenBSD 4.1 don’t seem to be much of a winning team.*

105 p. 236/237

**NetBSD 3.1** The configuration directory is `/usr/pkg/etc/zebra`.

106 p. 237/238

**NetBSD 3.1** First we copy the sample boot scripts `zebra` and `ripngd` from `/usr/pkg/share/examples/rc.d/` to `/etc/rc.d/`. Then we add the lines

`/etc/rc.conf`

```
ip6mode=router
zebra=YES
zebra_flags="--daemon -A ::1"
ripngd=YES
ripngd_flags="--daemon -A ::1"
```

to `/etc/rc.conf` to enable packet forwarding and the `zebra` and `ripngd` daemons as a daemon with their administrative interface bound to the IPv6 loopback address. To make the services more conveniently accessible through their administrative interface using their symbolic names we should also add the lines shown on page 238 to `/etc/services` and change the owner of the directory `/usr/pkg/etc/zebra` to `quagga`. To bring up all interfaces create the usual file `/etc/ifconfig.wm0` for every interface `wm0` with the IPv6 address configuration in it. Finally we either reboot or, if the `ip6mode` variable was already correctly set during the last boot, start the daemons by manually running the boot scripts.

110 p. 248

**NetBSD 3.1** To start the `ospf6d` daemon we have to copy the boot script `ospf6d` from `/usr/pkg/share/examples/rc.d/` to `/etc/rc.d/` and add the lines

`/etc/rc.conf`

```
ospf6d=YES
ospf6d_flags="--daemon -A ::1"
```

to `/etc/rc.conf`. If we don't want to use RIPng anymore we can remove the according boot file and configuration variables again.

**NetBSD 3.1** The `-g` option of `netstat` doesn't work as with FreeBSD 6.1. It only displays multicast routing data. The workaround option combination `-i -a` basically works but truncates the addresses. The alternative `ifmcstat` also mentioned with FreeBSD 6.1 works, however:

```
# ifmcstat
wm0:
    inet6 2001:db8:fedc:abcd::1
    inet6 fe80::20c:29ff:fe2e:eaea%wm0
    inet6 2001:db8:fedc:cdef::1
        group ff02::1:ff2e:eaea%wm0 refcnt 1
        group ff02::2:a917:663%wm0 refcnt 3
        group ff02::1%wm0 refcnt 3
        group ff02::1:ff00:1%wm0 refcnt 2
    enaddr 00:0c:29:2e:ea:ea multicnt 4
        33:33:ff:2e:ea:ea -- 33:33:ff:2e:ea:ea 1
        33:33:a9:17:06:63 -- 33:33:a9:17:06:63 1
        33:33:00:00:00:01 -- 33:33:00:00:00:01 1
        33:33:ff:00:00:01 -- 33:33:ff:00:00:01 1
lo0:
    inet6 ::1
    inet6 fe80::1%lo0
        group ff01::1 refcnt 2
        group ff02::2:a917:663%lo0 refcnt 2
        group ff02::1%lo0 refcnt 2
        group ff02::1:ff00:1%lo0 refcnt 2
```

It lists each interface, each address assigned to that interface, and the multicast groups that a socket from that address is listening to.

**OpenBSD 4.1** The `-g` option of `netstat` doesn't work as with FreeBSD 6.1. It only displays multicast routing data. The workaround option combination `-i -a` basically works but truncates the addresses.

**NetBSD 3.1, OpenBSD 4.1** Like FreeBSD 6.1 and Solaris 10 the `ping6` program uses the standard hop limit even on multicast addresses.

## 115 p. 267

**NetBSD 3.1** uses MLDv1 packets.

**OpenBSD 4.1** apparently has a broken MLD implementation.

According to tests both in a VMware Server 1.0.3 environment and a physical machine, OpenBSD 4.1 doesn't send MLD packets at all. Neither is `mcjoin` able to join multicast groups with a scope larger than `link-local`.



For this reason, multicast routing with OpenBSD 4.1 should be considered non-functional.

## 116 p. 271/272

**NetBSD 3.1** As with FreeBSD 6.1 we first build and install a kernel with the `MROUTING` and `PIM` options. Then we install the `pim6dd` package from the `ports/packages` collection; it contains a slightly earlier version of the `pim6dd` daemon presented with FreeBSD 6.1.

To enable the daemon we start it either from `/etc/rc.local` or manually again as usual.

**OpenBSD 4.1** There is no multicast routing daemon available with OpenBSD 4.1.

## 117 p. 278

**NetBSD 3.1** Again we need the kernel built with the `MROUTING` and `PIM` options. From the `ports/packages` collection we need the `pim6sd` package which contains the `pim6sd` daemon presented with FreeBSD 6.1.

To enable the daemon we start it either from `/etc/rc.local` or manually again.

## 118 p. 279

**NetBSD 3.1** We can use the same configuration as with FreeBSD 6.1, only on NetBSD 3.1 the configuration is expected in `/etc/pim6sd.conf`.

119 p. 279/280

**NetBSD 3.1** Again we can use the same configuration as for FreeBSD 6.1 except for the path to the configuration file.

*At least in a VMware Server 1.0.3 test environment this configuration causes reproducible kernel panics on the rendezvous point.*



120 p. 280/281

**NetBSD 3.1** The same reasoning as for FreeBSD 6.1 applies here, too. But with the problem mentioned above it won’t work at least in the test environment I have at hand.

121 p. 281

**NetBSD 3.1** The same reasoning as for FreeBSD 6.1 applies here, too. But with the problem mentioned above it won’t work at least in the test environment I have at hand.

122 p. 281

**NetBSD 3.1** The same `pim6stat` command described for FreeBSD 6.1 also ships with the `pim6sd` port/package on NetBSD 3.1.

123 p. 289–291

**NetBSD 3.1** The `wide-dhcpv6` package contains the same DHCPv6 implementation as FreeBSD 6.1. After installing the package or port we must copy the necessary boot script for a client, server or relay from `/usr/pkg/share/examples/rc.d/dhcp6*` to `/etc/rc.d`. To enable the client we must add the lines

```
/etc/rc.conf
```

```
dhcp6c=YES
dhcp6c_flags=wm0
```

to `/etc/rc.conf` if we want to use `wm0` as the DHCPv6 interface. For a server the equivalent settings are

`/etc/rc.conf`

```
dhcp6s=YES
dhcp6s_flags=wm0
```

Finally, just like on a FreeBSD 6.1 system we should create the shared secrets mentioned there. Just remember that the files go to `/usr/pkg/etc` instead of `/usr/local/etc`.

**OpenBSD 4.1** There is no DHCPv6 implementation available as a port or package.

124 p. 291/292

**NetBSD 3.1** We can copy the configuration file from the FreeBSD 6.1 example verbatim to NetBSD 3.1. Only the file location differs again: it goes to `/usr/local/etc/dhcp6s.conf`.

125 p. 292

**NetBSD 3.1** The `dhcp6c -i` command should work exactly the same as on FreeBSD 6.1.

126 p. 293

**NetBSD 3.1** Except for the interface and path names we can copy the configuration from the FreeBSD 6.1 example.

127 p. 294

**NetBSD 3.1** As with FreeBSD 6.1 the NTP support is based on a pre-standard draft and by default not compiled in.

128 p. 294

**NetBSD 3.1** *According to the documentation, NIS/NIS+ support is available.*

129 p. 294

**NetBSD 3.1** *As with FreeBSD 6.1, only static address assignments are available.*

130 p. 296

**NetBSD 3.1** *After copying the sample `dhcp6relay` boot script from `/usr/pkg/share/examples/rc.d/` to `/etc/rc.d/`, we just have to set the configuration variables*

```
/etc/rc.conf
```

```
dhcp6relay=YES
dhcp6relay_flags="-s 2001:db8:fedc::2 wm0
```

*to enable the relay and specify the DHCPv6 server and the interfaces served.*

131 p. 297

**NetBSD 3.1** *The `-r` option makes the `dhcp6relay` daemon use multicasts instead of unicasts to reach the DHCPv6 server like it does on FreeBSD 6.1.*

133 p. 304

**NetBSD 3.1, OpenBSD 4.1** *The programs are part of the core system.*

134 p. 317

**NetBSD 3.1, OpenBSD 4.1** As with FreeBSD 6.1 there is no documented way to filter by IPsec headers except with the `proto` keyword.

135 p. 324

**NetBSD 3.1** The KAME implementation mentioned with FreeBSD 6.1 should also support NetBSD 3.1.

**OpenBSD 4.1** Due to the reimplemention of IPsec, OpenBSD 4.1 doesn't seem to support the KAME implementation.

136 p. 351

**NetBSD 3.1, OpenBSD 4.1** Like FreeBSD, these BSDs also ship with a reasonably up-to-date BIND version in the core installation.

137 p. 352

**NetBSD 3.1** The `named.conf` file resides in `/etc`.

**OpenBSD 4.1** All the configuration files reside in a chroot environment in `/var/named`. The configuration file `/named.conf` is located in `/var/named/etc`.

138 p. 352/353

**NetBSD 3.1** To enable the `named` daemon we must add the line

```
/etc/rc.conf
```

```
named=YES
```

to `/etc/rc.conf`. This will run the `named` daemon without a chroot environment. Alternatively, the `/etc/defaults/rc.conf` file has the details on how to start `named` in a chroot environment.

Before we start the server for the first time we must run the command

```
# rndc-confgen -a
```

to create a file `/etc/rndc.key`.

After that to start the daemon we can either reboot the machine or run the command `/etc/rc.d/named start` manually.

**OpenBSD 4.1** Setting the variable `named_flags` to

```
/etc/rc.conf.local
```

```
named_flags=""
```

in `/etc/rc.conf.local` enables the name server. To start it the easiest way is a quick reboot because OpenBSD 4.1 doesn't come with individual start scripts for its services.

139 p. 354

**NetBSD 3.1** The configuration shipped with the distribution assumes all zone files to be stored in `/etc/namedb`.

**OpenBSD 4.1** The configuration as shipped with the distribution assumes all zone files to be rooted in `/var/named`. The subdirectories `master`, `slave` and `standard` contain the primary, secondary and standard zone files, respectively. The configuration file must specify the location of the zone files relative to `/var/named`.

140 p. 355

**NetBSD 3.1** As on FreeBSD 6.1, the command `/etc/rc.d/named reload` reloads the configuration.

**OpenBSD 4.1** Without a start script the command `rndc reload` is the preferred way to reload the configuration.