Chapter 7. Network

Table of Contents

7.1. Synopsis

This chapter delves into the topic of network configuration and performance, showcasing the robust networking capabilities of the FreeBSD operating system. Whether working with wired or wireless networks, this chapter provides a comprehensive guide to configuring and optimizing network connectivity in FreeBSD.

Before diving into the details, it is beneficial for readers to have a basic understanding of networking concepts such as protocols, network interfaces, and addressing.

This chapter covers:

- The ability to configure wired networks in FreeBSD, including network interface setup, addressing, and customization options.
- The skills to configure wireless networks in FreeBSD, encompassing wireless network interface setup, security protocols, and troubleshooting techniques.
- FreeBSD's networking capabilities and its reputation for excellent network performance.

- An understanding of various network services and protocols supported by FreeBSD, with configuration instructions for DNS, DHCP and more.

More information about how to make advanced network configurations in Advanced Networking.

7.2. Setting up the Network

Setting up a wired or wireless connection is a common task for a FreeBSD user. This section will show how to identify the wired and wireless network adapters and how to configure them.

Before starting with the configuration it is necessary to know the following network data:

- If the network has DHCP
- If the network does not have DHCP, the static IP to be used
- The netmask
- The IP address of the default gateway

> **Tip**
> The network connection may have been configured at installation time by bsdinstall(8).

7.2.1. Identify Network Adapters

FreeBSD supports a wide variety of network adapters for both wired and wireless networks. Check the Hardware Compatibility List for the used FreeBSD release to see if the network adapter is supported.

To get the network adapters used by our system execute the following command:

```
% pciconf -lv | grep -A1 -B3 network
```

The output should be similar to the following:

```
em0@pci0:0:25:0:        class=0x020000 rev=0x03 hdr=0x00 vendor=0x80
    vendor     = 'Intel Corporation'  1
    device     = '82567LM Gigabit Network Connection'  2
    class      = network
    subclass   = ethernet
--
iwn0@pci0:3:0:0:        class=0x028000 rev=0x00 hdr=0x00 vendor=0x80
    vendor     = 'Intel Corporation'  1
    device     = 'PRO/Wireless 5100 AGN [Shiloh] Network Connection'
    class      = network
```

The text before the '@' symbol is the name of the driver controlling the device. In this case these are em(4) and iwn(4).

1 Shows the name of the vendor

2 Shows the name of the device

> **Note**
>
> It is only necessary to load the network interface card module if FreeBSD has not detected it correctly.
>
> For example, to load the alc(4) module, execute the following command:
>
> ```
> # kldload if_alc
> ```
>
> Alternatively, to load the driver as a module at boot time, place the following line in **/boot/loader.conf**:
>
> ```
> if_alc_load="YES"
> ```

7.3. Wired Networks

Once the right driver is loaded the network adapter needs to be configured. FreeBSD uses the driver name followed by a unit number to name the network interface adapter. The unit number represents the order in which the adapter is detected at boot time, or is later discovered.

For example, `em0` is the first network interface card (NIC) on the system using the em(4) driver.

To display the network interface configuration, enter the following command:

```
% ifconfig
```

The output should be similar to the following:

```
em0: flags=8863<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu
        options=481249b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_H
        ether 00:1f:16:0f:27:5a
        inet6 fe80::21f:16ff:fe0f:275a%em0 prefixlen 64 scopeid 0x1
        inet 192.168.1.19 netmask 0xffffff00 broadcast 192.168.1.255
        media: Ethernet autoselect (1000baseT <full-duplex>)
        status: active
        nd6 options=23<PERFORMNUD,ACCEPT_RTADV,AUTO_LINKLOCAL>
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> metric 0 mtu 16384
        options=680003<RXCSUM,TXCSUM,LINKSTATE,RXCSUM_IPV6,TXCSUM_IP
        inet6 ::1 prefixlen 128
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x2
        inet 127.0.0.1 netmask 0xff000000
        groups: lo
        nd6 options=21<PERFORMNUD,AUTO_LINKLOCAL>
```

In this example, the following devices were displayed:

- `em0` : The Ethernet interface.
- `lo0` : The loop interface is a software loopback mechanism which may be used for performance analysis, software testing, and/or local communication. More information in lo(4).

The example shows that `em0` is up and running.

The key indicators are:

1. `UP` means that the interface is configured and ready.
2. The interface has an IPv4 Internet (`inet`) address, `192.168.1.19`.
3. The interface has an IPv6 Internet (`inet6`) address, `fe80::21f:16ff:fe0f:275a%em0`.
4. It has a valid subnet mask (`netmask`), where `0xffffff00` is the same as `255.255.255.0`.
5. It has a valid broadcast address, `192.168.1.255`.
6. The MAC address of the interface (`ether`) is `00:1f:16:0f:27:5a`.
7. The physical media selection is on autoselection mode (`media: Ethernet autoselect (1000baseT <full-duplex>)`).
8. The status of the link (`status`) is `active`, indicating that the carrier signal is detected. For `em0`, the `status: no carrier` status is normal when an Ethernet cable is not plugged into the interface.

If the ifconfig(8) output had shown something similar to the next output it would indicate the interface has not been configured:

```
em0: flags=8822<BROADCAST,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=481249b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_H
        ether 00:1f:16:0f:27:5a
        media: Ethernet autoselect
        status: no carrier
        nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
```

7.3.1. Configuring Static IPv4 Address

This section provides a guide to configuring a static IPv4 address on a FreeBSD system.

The network interface card configuration can be performed from the command line with ifconfig(8) but will not persist after a reboot unless the configuration is also added to **/etc/rc.conf**.

> **Note**
>
> If the network was configured during installation by bsdinstall(8), some entries for

> the network interface card (NICs) may be already present. Double check **/etc/rc.conf** before executing sysrc(8).

The IP address can be set executing the following command:

```
# ifconfig em0 inet 192.168.1.150/24
```

To make the change persist across reboots execute the following command:

```
# sysrc ifconfig_em0="inet 192.168.1.150 netmask 255.255.255.0"
```

Add the default router executing the following command:

```
# sysrc defaultrouter="192.168.1.1"
```

Add the DNS records to **/etc/resolv.conf**:

```
nameserver 8.8.8.8
nameserver 8.8.4.4
```

Then restart `netif` and `routing` executing the following command:

```
# service netif restart && service routing restart
```

The connection can be tested using ping(8):

```
% ping -c2 www.FreeBSD.org
```

The output should be similar to the following:

```
PING web.geo.FreeBSD.org (147.28.184.45): 56 data bytes
```

```
64 bytes from 147.28.184.45: icmp_seq=0 ttl=51 time=55.173 ms
64 bytes from 147.28.184.45: icmp_seq=1 ttl=51 time=53.093 ms

--- web.geo.FreeBSD.org ping statistics ---
2 packets transmitted, 2 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 53.093/54.133/55.173/1.040 ms
```

7.3.2. Configuring Dynamic IPv4 Address

If the network has a DHCP server, it is very easy to configure the network interface to use DHCP. FreeBSD uses dhclient(8) as the DHCP client. dhclient(8) will automatically provide the IP, the netmask and the default router.

To make the interface work with DHCP execute the following command:

```
# sysrc ifconfig_em0="DHCP"
```

dhclient(8) can be used manually by running the following command:

```
# dhclient em0
```

The output should be similar to the following:

```
DHCPREQUEST on em0 to 255.255.255.255 port 67
DHCPACK from 192.168.1.1
unknown dhcp option value 0x7d
bound to 192.168.1.19 -- renewal in 43200 seconds.
```

In this way it can be verified that the address assignment using DHCP works correctly.

> **Tip**
>
> dhclient(8) client can be started in background. This can cause trouble with applications depending on a working network, but it will provide a faster startup in many cases.

To execute dhclient(8) in background execute the following command:

```
# sysrc background_dhclient="YES"
```

Then restart `netif` executing the following command:

```
# service netif restart
```

The connection can be tested using ping(8):

```
% ping -c2 www.FreeBSD.org
```

The output should be similar to the following:

```
PING web.geo.FreeBSD.org (147.28.184.45): 56 data bytes
64 bytes from 147.28.184.45: icmp_seq=0 ttl=51 time=55.173 ms
64 bytes from 147.28.184.45: icmp_seq=1 ttl=51 time=53.093 ms

--- web.geo.FreeBSD.org ping statistics ---
2 packets transmitted, 2 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 53.093/54.133/55.173/1.040 ms
```

7.3.3. IPv6

IPv6 is the new version of the well-known IP protocol, also known as IPv4.

IPv6 provides several advantages over IPv4 as well as many new features:

- Its 128-bit address space allows for 340,282,366,920,938,463,463,374,607,431,768,211,456 addresses. This addresses the IPv4 address shortage and eventual IPv4 address exhaustion.
- Routers only store network aggregation addresses in their routing tables, thus

reducing the average space of a routing table to 8192 entries. This addresses the scalability issues associated with IPv4, which required every allocated block of IPv4 addresses to be exchanged between Internet routers, causing their routing tables to become too large to allow efficient routing.

- Address autoconfiguration (RFC4862).
- Mandatory multicast addresses.
- Built-in IPsec (IP security).
- Simplified header structure.
- Support for mobile IP.
- IPv6-to-IPv4 transition mechanisms.

FreeBSD includes the KAME project IPv6 reference implementation and comes with everything needed to use IPv6.

This section focuses on getting IPv6 configured and running.

There are three different types of IPv6 addresses:

Unicast

A packet sent to a unicast address arrives at the interface belonging to the address.

Anycast

These addresses are syntactically indistinguishable from unicast addresses but they address a group of interfaces. The packet destined for an anycast address will arrive at the nearest interface.

Multicast

These addresses identify a group of interfaces. A packet destined for a multicast address will arrive at all interfaces belonging to the multicast group. The IPv4 broadcast address, usually `xxx.xxx.xxx.255`, is expressed by multicast addresses in IPv6.

When reading an IPv6 address, the canonical form is represented as `x:x:x:x:x:x:x:x`, where each `x` represents a 16 bit hex value. An example is `FEBC:A574:382B:23C1:AA49:4592:4EFE:9982`.

Often, an address will have long substrings of all zeros. A `::` (double colon) can be used to replace one substring per address. Also, up to three leading `0`s per hex value can be omitted. For example, `fe80::1` corresponds to the canonical form `fe80:0000:0000:0000:0000:0000:0000:0001`.

A third form is to write the last 32 bits using the well known IPv4 notation. For example, `2002::10.0.0.1` corresponds to the hexadecimal canonical representation `2002:0000:0000:0000:0000:0000:0a00:0001`, which in turn is equivalent to `2002::a00:1`.

To view a FreeBSD system's IPv6 address execute the following command:

```
# ifconfig
```

The output should be similar to the following:

```
em0: flags=8863<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu
        options=481249b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_H
        ether 00:1f:16:0f:27:5a
        inet 192.168.1.150 netmask 0xffffff00 broadcast 192.168.1.25
        inet6 fe80::21f:16ff:fe0f:275a%em0 prefixlen 64 scopeid 0x1
        media: Ethernet autoselect (1000baseT <full-duplex>)
        status: active
        nd6 options=23<PERFORMNUD,ACCEPT_RTADV,AUTO_LINKLOCAL>
```

In this example, the `em0` interface is using `fe80::21f:16ff:fe0f:275a%em0`, an auto-configured link-local address which was automatically generated from the MAC address.

Some IPv6 addresses are reserved. A list of reserved addresses can be checked in the following table:

Table 1. Example IPv6 Reserved Addresses

IPv6 address	Description	Notes
`::/128`	unspecified	Equivalent to `0.0.0.0` in IPv4.

IPv6 address	Description	Notes
`::1/128`	loopback address	Equivalent to `127.0.0.1` in IPv4.
`::ffff:0.0.0.0/96`	IPv4 mapped IPv6 address	The lower 32 bits are the IPv4 address for compatibility with IPv4 hosts and routers.
`fe80::/10`	link-local unicast	Equivalent to 169.254.0.0/16 in IPv4.
`fc00::/7`	unique-local	Unique local addresses are intended for local communication and are only routable within a set of cooperating sites.
`ff00::/8`	multicast	
`2000::/3`	global unicast	All global unicast addresses are assigned from this pool. The first 3 bits are `001`.
`2001:db8::/32`, `3fff::/20`	documentation	IPv6 address prefix for use in documentation.

For further information on the structure of IPv6 addresses, refer to RFC4291.

7.3.4. Configuring Static IPv6 Address

To configure a FreeBSD system as an IPv6 client with a static IPv6 address it is necessary to set the IPv6 address.

Execute the following commands to meet the requirements:

```
# sysrc ifconfig_em0_ipv6="inet6 2001:db8:4672:6565:2026:5043:2d42:5344 prefixlen 64"
```

To assign a default router, specify its address executing the following command:

```
# sysrc ipv6_defaultrouter="2001:db8:4672:6565::1"
```

To configure an additional IPv6 anycast address, specify the anycast address as an `_aliasN`, as specified in rc.conf(5), followed by the `anycast` option:

```
# sysrc ifconfig_em0_alias0="inet6 2001:db8:4672:6565::a anycast"
```

Keep in mind that the applications can't bind to anycast addresses; in that case you need to use an alias address instead.

7.3.5. Configuring Dynamic IPv6 Address

To dynamically configure the IPv6 address of the interface using SLAAC, execute the following commands:

```
# sysrc ifconfig_em0_ipv6="inet6 accept_rtadv"
# sysrc rtsold_enable="YES"
```

Note that when IPv6 packet forwarding is enabled (i.e., `ipv6_gateway_enable=YES`), the system will not configure a SLAAC address unless the `net.inet6.ip6.rfc6204w3` sysctl(8) variable is set to 1.

7.3.6. Router Advertisement and Host Auto Configuration

This section demonstrates how to setup rtadvd(8) on an IPv6 router to advertise the IPv6 network prefix and default route.

To enable rtadvd(8), execute the following command:

```
# sysrc rtadvd_enable="YES"
```

It is important to specify the interface on which to do IPv6 router advertisement. For example, to tell rtadvd(8) to use `em0`:

```
# sysrc rtadvd_interfaces="em0"
```

Next, create the configuration file, **/etc/rtadvd.conf** as seen in this example:

```
em0:\
        :addrs#1:addr="2001:db8:1f11:246::":prefixlen#64:tc=ether:
```

Replace `em0` with the interface to be used and `2001:db8:1f11:246::` with the prefix of the allocation.

For a dedicated `/64` subnet, nothing else needs to be changed. Otherwise, change the `prefixlen#` to the correct value.

7.3.7. IPv6 and IPv4 Address mapping

When IPv6 is enabled on a server, there may be a need to enable IPv4 mapped IPv6 address communication. This compatibility option allows for IPv4 addresses to be represented as IPv6 addresses. Permitting IPv6 applications to communicate with IPv4 and vice versa may be a security issue.

This option may not be required in most cases and is available only for compatibility. This option will allow IPv6-only applications to work with IPv4 in a dual stack environment. This is most useful for third party applications which may not support an IPv6-only

environment.

To enable this feature execute the following command:

```
# sysrc ipv6_ipv4mapping="YES"
```

7.4. Wireless Networks

Most wireless networks are based on the IEEE® 802.11 standards.

FreeBSD supports networks that operate using 802.11a, 802.11b, 802.11g and 802.11n.

> **Note**
> 802.11ac support on FreeBSD is currently under development.

A basic wireless network consists of multiple stations communicating with radios that broadcast in either the 2.4GHz or 5GHz band, though this varies according to the locale and is also changing to enable communication in the 2.3GHz and 4.9GHz ranges.

There are three basic steps to configure a wireless network:

1. Scan and select an access point
2. Authenticate the station
3. Configure an IP address or use DHCP.

The following sections discuss each step.

7.4.1. Quick Start to Connect to a Wireless Network

Connecting FreeBSD to an existing wireless network is a very common situation.

This procedure shows the steps required:

- The first step will be to obtain the SSID (Service Set Identifier) and PSK (Pre-Shared Key) for the wireless network from the network administrator.
- The second step will be to add default configuration paramaters and an entry for

this network to **/etc/wpa_supplicant.conf**. If the file does not exist, create it:

```
ctrl_interface=/var/run/wpa_supplicant
eapol_version=1
ap_scan=1
fast_reauth=1

network={
 ssid="myssid"  1
 psk="mypsk"  2
}
```

1 Is the SSID of the wireless network. Replace it with the name of the wireless network.

2 Is the PSK of the wireless network. Replace it with the password of the wireless network.

- The third step will be to add the network entry to configure the network on startup:

```
# sysrc wlans_iwn0="wlan0"
# sysrc ifconfig_wlan0="WPA DHCP"
```

- And the last step will be the restart `netif` service executing the following command:

```
# service netif restart
```

7.4.2. Basic Wireless Configuration

The first step will be to configure the wireless network card to an interface. To find out what wireless network cards are in the system check the section Identify Network Adapters.

```
# ifconfig wlan0 create wlandev iwm0
```

To make the change persist across reboots execute the following command:

```
# sysrc wlans_iwm0="wlan0"
```

> **Note**
>
> Since the regulatory situation is different in various parts of the world, it is necessary to correctly set the domains that apply to your location to have the correct information about what channels can be used.
>
> The available region definitions can be found in **/etc/regdomain.xml**. To set the data at runtime, use `ifconfig`:
>
> ```
> # ifconfig wlan0 regdomain etsi2 country AT
> ```
>
> To persist the settings, add it to **/etc/rc.conf**:
>
> ```
> # sysrc create_args_wlan0="country AT regdomain etsi2"
> ```

7.4.3. Scan Wireless Networks

Available wireless networks can be scanned using ifconfig(8).

To list the wireless networks execute the following command:

```
# ifconfig wlan0 up list scan
```

The output should be similar to the following:

```
SSID/MESH ID                    BSSID              CHAN RATE    S:
FreeBSD                         e8:d1:1b:1b:58:ae     1   54M  -47:
NetBSD                          d4:b9:2f:35:fe:08     1   54M  -80:
OpenBSD                         fc:40:09:c6:31:bd    36   54M  -94:
GNU-Linux                       dc:f8:b9:a0:a8:e0    44   54M  -95:
Windows                         44:48:b9:b3:c3:ff    44   54M  -84:
MacOS                           46:48:b9:b3:c3:ff    44   54M  -84:
```

1. SSID/MESH ID identifies the name of the network.
2. BSSID identifies the MAC address of the access point.
3. CAPS field identifies the type of each network and the capabilities of the stations operating there (see the definition of `list scan` in ifconfig(8) for more details).

7.4.4. Connection and Authentication to a Wireless Network

Once a wireless network has been selected from the list of scanned networks, it is necessary to perform the connection and the authentication. In the vast majority of wireless networks, authentication is done with a password configured in the router. Other schemes require cryptographic handshakes to be completed before data traffic can flow, either using pre-shared keys or secrets, or more complex schemes that involve backend services such as RADIUS.

7.4.4.1. Authenticate with WPA2/WPA/Personal

The authentication process in a wireless network is managed by wpa_supplicant(8).

The wpa_supplicant(8) configuration will be made in the **/etc/wpa_supplicant.conf** file. For more information, see wpa_supplicant.conf(5).

Once the scanning of the wireless networks has been carried out, a network has been chosen and have the password (PSK), that information will be added to the file **/etc/wpa_supplicant.conf** as in the following example:

```
network={
        scan_ssid=1  1
```

```
        ssid="FreeBSD"  2
        psk="12345678"  3
}
```

1 SSID scan technique. Only need to use this option if the network is hidden.

2 Network name.

3 Password of the wireless network.

The next step will be to configure the wireless connection in the file **/etc/rc.conf**.

To use a static address it will be necessary to execute the following command:

```
# sysrc ifconfig_wlan0="inet 192.168.1.20 netmask 255.255.255.0"
```

To use a dynamic address it will be necessary to execute the following command:

```
# sysrc ifconfig_wlan0="WPA DHCP"
```

Then restart the network executing the following command:

```
# service netif restart
```

> **Note**
> More information on how to perform more advanced methods of authentication can be obtained at Wireless Advanced Authentication.

7.4.4.2. Authenticate with Open Networks

> **Tip**
>
> It is important that the user is **very** careful when connecting to open networks without any kind of authentication.

Once the wireless network scan is done and the SSID of the wireless network is selected, execute the following command:

```
# ifconfig wlan0 ssid SSID
```

And then execute dhclient(8) to get the address configured:

```
# dhclient wlan0
```

7.4.5. Using Both Wired and Wireless Connections

A wired connection provides better performance and reliability, while a wireless connection provides flexibility and mobility. Laptop users typically want to roam seamlessly between the two types of connections.

On FreeBSD, it is possible to combine two or even more network interfaces together in a "failover" fashion. This type of configuration uses the most preferred and available connection from a group of network interfaces, and the operating system switches automatically when the link state changes.

Link aggregation and failover is covered in Link Aggregation and Failover and an example for using both wired and wireless connections is provided at Failover Mode Between Ethernet and Wireless Interfaces.

7.5. Hostname

The hostname represents the fully qualified domain name (FQDN) of the host on the network.

> **Tip**
> If no hostname is set for the host, FreeBSD will call itself `Amnesiac`.

7.5.1. Check The Current Hostname

hostname(1) can be used to check the current hostname:

```
$ hostname
```

The output should be similar to the following:

```
freebsdhostname.example.com
```

7.5.2. Change Hostname

To change the hostname of the host and persist it across reboots execute the following command:

```
# sysrc hostname="freebsdhostname.example.com"
```

7.6. DNS

The DNS could be understood as a telephone directory in which an IP is identified to a hostname and vice versa.

There are three files that handle how a FreeBSD system interact with the DNS. These three files are hosts(5), resolv.conf(5) and nsswitch.conf(5)

Unless otherwise stated in the **/etc/nsswitch.conf** file, FreeBSD will look at the addresses in the **/etc/hosts** file and then the DNS information in the **/etc/resolv.conf** file.

> **Note**
>
> The nsswitch.conf(5) file specifies how the nsdispatch (name-service switch dispatcher) should operate.
>
> By default, the hosts section of the **/etc/nsswitch.conf** file will be as follows:
>
> ```
> hosts: files dns
> ```
>
> For example, in case of using the nscd(8) service. The order of preference could be changed by leaving the line as follows:
>
> ```
> hosts: files cache dns
> ```

7.6.1. Local addresses

The **/etc/hosts** file is a simple text database who provide host name to IP address mappings. Entries for local computers connected via a LAN can be added to this file for simplistic naming purposes instead of setting up a DNS server. Additionally, **/etc/hosts** can be used to provide a local record of Internet names, reducing the need to query external DNS servers for commonly accessed names.

For example, in the case of having a local instance of www/gitlab-ce in a local environment, it could be added as follows to the file **/etc/hosts**:

```
192.168.1.150 git.example.com git
```

7.6.2. Configuring the Nameserver

How a FreeBSD system accesses the Internet Domain Name System (DNS) is controlled by resolv.conf(5).

The most common entries to **/etc/resolv.conf** are:

`nameserver`	The IP address of a name server the resolver should query. The servers are queried in the order listed with a maximum of three.
`search`	Search list for hostname lookup. This is normally determined by the domain of the local hostname.
`domain`	The local domain name.

A typical **/etc/resolv.conf** looks like this:

```
search example.com
nameserver 147.11.1.11
nameserver 147.11.100.30
```

> **Note**
> Only one of the `search` and `domain` options should be used.

When using DHCP, dhclient(8) usually rewrites **/etc/resolv.conf** with information received from the DHCP server.

> **Tip**
> If the machine in which the configuration is being made is **not** a DNS server, local-unbound(8) can be used to improve DNS lookup performance.
>
> To enable it at boot time execute the following command:

```
# sysrc local_unbound_enable="YES"
```

To start the local-unbound(8) service execute the following command:

```
# service local_unbound start
```

7.7. Troubleshooting

When troubleshooting hardware and software configurations, check the simple things first.

- Is the network cable plugged in?
- Are the network services properly configured?
- Is the firewall configured correctly?
- Is the NIC supported by FreeBSD?
- Is the router working correctly?

Tip

Before sending a bug report, always check the Hardware Notes in the FreeBSD release page, update the version of FreeBSD to the latest STABLE version, check the mailing list archives, and search the Internet.

7.7.1. Troubleshooting in Wired Networks

If the card works, yet performance is poor, read through tuning(7). Also, check the network configuration as incorrect network settings can cause slow connections.

`No route to host` messages occur if the system is unable to route a packet to the

destination host. This can happen if no default route is specified or if a cable is unplugged. Check the output of `netstat -rn` and make sure there is a valid route to the host. If there is not, read Gateways and Routes.

`ping: sendto: Permission denied` error messages are often caused by a misconfigured firewall. If a firewall is enabled on FreeBSD but no rules have been defined, the default policy is to deny all traffic, even ping(8). Refer to Firewalls for more information.

7.7.2. Troubleshooting in Wireless Networks

This section describes a number of steps to help troubleshoot common wireless networking problems.

- If the access point is not listed when scanning, check that the configuration has not limited the wireless device to a limited set of channels.
- If the device cannot associate with an access point, verify that the configuration matches the settings on the access point. This includes the authentication scheme and any security protocols. Simplify the configuration as much as possible. If using a security protocol such as WPA2 or WPA, configure the access point for open authentication and no security to see if traffic will pass.
- Once the system can associate with the access point, diagnose the network configuration using tools like ping(8).
- There are many lower-level debugging tools. Debugging messages can be enabled in the 802.11 protocol support layer using wlandebug(8).

Last modified on: July 27, 2025 by kraytonian

About

FreeBSD

FreeBSD Foundation

Get FreeBSD

Code of Conduct

Security Advisories

Documentation

Documentation portal

Manual pages

Presentations and papers

Previous versions

4.4BSD Documents

Wiki

Community

Get involved

Community forum

Mailing lists

IRC Channels

Bug Tracker

Legal

Donations

Licensing

Privacy Policy

Legal notices

Made with ♥ by the FreeBSD Community