802.11 a/b/g Wi-Fi Support#

Wi-Fi capability is built into the two "-hc" variants of the stack (io-pkt-v4-hc and io-pkt-v6-hc). The NetBSD stack includes it's own separate 802.11 MAC layer that is independent of the driver. Many other implementations pull the 802.11 MAC inside the driver and this results in separate interfaces and configuration utilities required for every driver. By writing a driver that conforms to the stack's 802.11 layer, the same set of configuration and control utilities can be used for all wireless drivers.

The networking Wi-Fi solution allows a user to join or host WLAN (Wireless LAN) networks based on IEEE 802.11 specifications. Using io-pkt, you can connect using a peer to peer mode (ad-hoc mode, also referred to as Independent Basic Service Set, IBSS configuration) or you can either act as a client for a Wireless Access Point (WAP, also known as a base station) or configure Neutrino to act as a Wireless Access Point. This second mode is referred to as Infrastructure Mode or BSS (Basic Service Set).

add-hoc mode allows you create a wireless network quickly by allowing wireless nodes within range (for example, the wireless devices in a room) to communicate directly with each other without the need for a wireless access point. While being easy to construct, it may not be appropriate for a large number of nodes because of performance degradation, limited range, non-central administration and weak encryption.

Infrastructure mode is the more common network configuration where all wireless hosts (clients) connect to the wireless network via a WAP (Wireless Access Point). The WAP centrally controls access and authentication to the wireless network and provides access to rest of your network. More than one WAP can exist on a wireless network to services large numbers of wireless clients.

io-pkt supports WEP, WPA, WPA2 or no security for authentication and encryption when acting as the WAP or client. WPA/WPA2 is the recommended encryption protocol for use with your wireless network. WEP is not as secure as WPA/WPA2 and is known to be breakable. It is available for backward compatibility with already deployed wireless networks.

For those of you that just want to jump in, go here for the general case on how to get your client to connect in.

NetBSD 80211 Layer#

(This section taken from the NetBSD Documentation Internal Documents)

The net80211 layer provides functionality required by wireless cards. It is located under sys/net80211. The code is meant to be shared between FreeBSD and NetBSD and NetBSD-specific bits should be attempted to be kept in the source file ieee80211_netbsd.c (likewise, there is ieee80211_freebsd.c in FreeBSD).

The ieee80211 interfaces are documented in Chapter 9 of the NetBSD manual pages. This document does not attempt to duplicate information already available there.

The responsibilities of the net80211 layer are the following:

- MAC address based access control
- crypto
- input and output frame handling
- node management
- radiotap framework for bpf/tcpdump
- rate adaption
- supplementary routines such as conversion functions and resource management

The ieee80211 layer positions itself logically between the device driver and the ethernet module, although for transmission it is called indirectly by the device driver instead of control passing straight through it. For input,

the ieee80211 layer receives packets from the device driver, strips any information useful only to wireless devices and in case of data payload proceeds to hand the Ethernet frame up to ether_input.

Device Management#

The way to describe an ieee80211 device to the ieee80211 layer is by using a struct ieee80211com, declared in sys/net80211/ieee80211_var.h. It is used to register a device to the ieee80211 from the device driver by calling ieee80211_ifattach. Fields to be filled out by the caller include the underlying struct ifnet pointer, function callbacks and device capability flags. If a device is detached, the ieee80211 layer can be notified with the call ieee80211_ifdetach.

Nodes#

A node represents another entity in the wireless network. It is usually a base station when operating in BSS mode, but can also represent entities in an ad-hoc network. A node is described by struct ieee80211_node, declared in sys/net80211/ieee80211_node.h. Examples of fields contained in the structure include the node unicast encryption key, current transmit power, the negotiated rate set and various statistics.

A list of all the nodes seen by a certain device is kept in the struct ieee80211com instance in the field ic_sta and can be manipulated with the helper functions provided in sys/net80211/ieee80211_node.c. The functions include, for example, methods to scan for nodes, iterate through the nodelist and functionality for maintaining the network structure.

Crypto Support#

Crypto support enables the encryption and decryption of the network frames. It provides a framework for multiple encryption methods such as WEP and null crypto. Crypto keys are mostly managed through the ioctl interface and inside the ieee80211 layer, and the only time drivers need to worry about them is in the send routine when they must test for an encapsulation requirement and call ieee80211_crypto_encap if necessary.

Utilities and Libraries#

We are in the process of putting together a generic configuration library that interfaces to the supplicant and provides a programmatic interface for configuring your wireless connection (this is called "wlconfig" and can be found in the source tree under trunk/lib/wlconfig). We'll be using this library to create a ~Wi-Fi GUI for Photon but it is, of course, also available for anyone else to develop a configuration tool.

The main utilities required for Wi-Fi usage are:

- **wpa_supplicant**: Client side authentication / key management / network persistence daemon
  - **wpa_cli**: Command line utility for interacting with the supplicant
  - **wpa_supplicant** also requires the following:
    - **libcrypto.so** : Crypto Library (created from OpenSSL)
    - **libssl.so** : Secure Socket Library (created from OpenSSL)
    - **random** : Executable which creates /dev/urandom for random number generation
    - **libm.so** : Math library required by **random**
    - **libz.so** : Compression library required by **random**
  - **wpa_supplicant** also needs a read / write file system for creation of a "ctrl_interface" directory (see the sample wpa_supplicant.conf configuration file). /dev/shmem can't be used since it is not possible to create a directory there.
- **hostapd**: Server side (Access Point) authentication / key management daemon

There are also subsidiary utilities which are available but likely you won't need to use.

- **wiconfig**: Configuration utility for wi / awi drivers. **ifconfig** can handle the device configuration required without needing this utility.
- **wlanctl** : examine IEEE 802.11 wireless LAN client/peer table

How do I use Wi-Fi with the new stack?#

When connecting to a Wireless Network in Neutrino, the first step that needs to be done is to start the stack process. The stack needs to be started with the appropriate driver for the installed hardware. Information on available drivers can be found in the drivers wiki. For this sample, we will use the driver for network adapters using the RAL chipset "devnp-ral.so". After a default installation, all driver binaries are installed under the staging directory /<cpu>/lib/dll. Note that the io-pkt-v4 stack variant does NOT have the 802.11 layer built in and therefore can't be used with WiFi drivers. If you attempt to load a WiFi driver into io-pkt-v4, you will see a number of unresolved symbol errors produced and the driver will not work.

- io-pkt-v4-hc -d /lib/dll/devnp-ral.so

or

- io-pkt-v6-hc -d ral

If the network driver is installed to a location other than /lib/dll, you will need to specify the fullpath and filename of the driver on the command line.

Once the stack and appropriate driver has been started, you need to determine what wireless networks are available. If you already have the network name (SSID or Service Set Identifier), you can skip this step as you already know what network you want to join. These steps can also be used to determine if the network you wish to joint is within range and active.

To determine what wireless networks are available to join, you must first set the interface status to up. This is done with the ifconfig command

- ifconfig ral0 up

After this step, you can then check to see what wireless networks have advertised themselves using the command

- wlanctl ral0

This command will provide a listing of available networks and their configuration. You can use this information to determine the network name (SSID), its mode of operation (ad-hoc or infrastructure mode), and radio channel for example.

You can also force a manual scan of the network with the command

- ifconfig ral0 scan

This will cause the wireless adapter to scan for WAP stations or ad-hoc nodes within range of the wireless adapter. The available networks will be listed in the utility output along with their configuration. Scan information can also be retrieved from the wpa_supplicant utility (detailed later in this document).

Once the appropriate driver is started and the wireless network has been located, you will need to choose the network mode that will be used (ad-hoc vs Infrastructure mode), the authentication method to attach to the wireless network, and the encryption protocol being used, if encryption is being used at all.

NOTE: It is recommended that you implement encryption on your wireless network if there are no physical security solutions being used.

By default, most network drivers will implement Infrastructure mode (BSS) as most wireless networks are configured to allow network access via a WAP. If you wish to implement an ad-hoc network, you can change the network mode via the ifconfig command.

- ifconfig ral0 mediaopt adhoc (Create or join ad-hoc networks)

If you wish to switch back to infrastructure mode, you can use the command

- ifconfig ral0 -mediaopt adhoc (Connect to WAP on Infrastructure networks)

The media options available for your driver are listed in the manual entry for the driver. When you are in ad-hoc mode you advertise your presence to other peers which are within physical range. This means that other 802.11 devices can discover you and connect to your network.

Whether you are a client in infrastructure mode, or are using ad-hoc mode, the steps to implement encryption are the same. You will need to make sure that you are using the authentication method and encryption key that has been chosen for the network. If you wish to connect with your peers using an ad-hoc wireless network, all peers must be using the same authentication method and encryption key. If you are a client connecting to a WAP, you must use the same authentication method and encryption key as has been configured on the WAP.

Connecting to a wireless network#

For the general case of connecting to a wi-fi network, we recommend using the ***wpa_supplicant*** daemon. It handles unsecure, WEP, WPA and WPA2 networks and provides a mechanism for saving network information in a configuration file that is scanned on start up, thereby removing the need for the user to constantly re-enter network parameters upon a re-boot or moving from one network domain to another. The information following covers the more specific cases if you don't want to run the supplicant, but for the casual user, that's the way to go.

The following Wi-Fi connectivity capabilities are supported:

- Connecting to a Wireless Network Using WPA/WPA2 for Authentication and Encryption
- Connecting to a Wireless Network using WEP (Wired Equivalent Privacy) for Authentication and Encryption
- Connecting to an Unsecure Wireless Network
- Using wpa_supplicant to manage all of your Wi-Fi network connections

We will also be rolling out a new Wi-Fi configuration GUI as part of Photon that interfaces into the wpa_supplicant directly.

Once connected, the interface needs to be configured in the standard way

- TCP/IP Configuration in A Wireless Network (Client in Infrastructure Mode, or Ad-hoc Mode)

Creating a Wireless Access Point#

Creating A WAP (Wireless Acess Point)#

A Wireless Access Point (WAP) is a system which allows wireless clients to access the rest of the network or the internet. Your WAP will operate in BSS mode. A WAP will have at least one wireless network interface, to provide a connection point for your wireless clients, and one wired network interface which will connect to the rest of your network. Your WAP will act as a bridge or gateway between the wireless clients, and the wired intra-net or internet.

To setup your wireless access point, you will first need to start the appropriate driver for your network adapters. Note that not all network adapter hardware will support operating as an access point. Please refer to the documentation for your specific hardware for further information. For the wireless access point samples, we will use the driver for the RAL wireless chipsets "devnp-ral.so", the driver "devnp-i82544.so" for the wired interface. After a default installation, all driver binaries are installed under the directory $QNX_TARGET/<cpu>/lib/dll (or in the same location in your staging directory if you've built the source yourself).

- io-pkt-v4-hc -d ral -d i82544

or

- io-pkt-v4-hc -d /lib/dll/devnp-ral.so -d /lib/dll/devnp-i82544.so

or

- io-pkt-v6-hc -d ral -d i82544

If the driver is installed to a location other than /lib/dll, you will need to specify the fullpath and filename of the driver on the command line.

The next step to configure your WAP is to determine whether it will be acting as a gateway, or a bridge to the rest of the network.

- **Acting as a gateway**

When your WAP acts as a gateway, it is forwarding traffic between two subnets (your wireless network, and the wired network). For TCP/IP, this means that the wireless TCP/IP clients cannot directly reach the wired TCP/IP clients without first sending their packets to the gateway (your WAP). Your WAP network interfaces will also each be assigned an IP address. This type of configuration is common for SOHO (small office, home office) or home use where the WAP is directly connected to your internet service provider. Using this type of configuration allows you to keep all of your network hosts behind a firewall/NAT, and allows you to define and administer your own TCP/IP network. The TCP/IP configuration of a gateway, and firewall is the same whether your network interfaces are wired or wireless. Please see the documentation for more information on TCP/IP configuration of gateways, firewalls and NAT.

Once your network is active, you will assign each interface of your WAP an IP address, enable forwarding of IP packets between interfaces, and apply the appropriate firewall and NAT configuration.

- TCP/IP Configuration in A Wireless Network (DHCP Server On WAP Acting As A Gateway)

- **Acting as a bridge**

When your WAP acts as a bridge, it is connecting your wireless and wired network as if it was one physically connected network (broadcast domain, layer 2). In this case, all the wired and wireless hosts are on the same TCP/IP subnet and can directly exchange TCP/IP packets without the need for the WAP to act as a gateway. In this case, you do not need to assign your WAP network interfaces an IP address to be able to exchange packets between the wireless and wired network. A bridged WAP could be used to allow wireless clients onto your corporate or home network and have them configured in the same manner as the wireless hosts. You will not

need to add more services (such as DHCP) or manipulate routing tables. The wireless clients will make use of the same network resources that the wired network hosts use.

Note: While it is not necessary to assign your WAP network interfaces an IP address for TCP/IP connectivity between the wireless clients and wired hosts, you probably will want to assign at least one of your WAP interfaces and IP address so that the device can be addressed for the purposes of managing the device or statistics gathering.

To enable your WAP to act as a bridge, you first need to create a bridge interface.

- ifconfig bridge0 create

In this case, "bridge" is the specific interface type, while '0' is a unique instance of the interface type. There can be no space between bridge and 0. "bridge0" becomes the new interface name.

- brconfig bridge0 add ral0 add wm0 up

The brconfig command create a logical link between the interfaces added to the bridge (In this case bridge0). This command adds the interfaces ral0 (our wireless interface) and wm0 (our wired interface). The up option is required to activate the bridge.

Note: Remember to mark your bridge as "up' or else it will not be activated

To see the status of your defined bridge interface, you can use the command

- brconfig bridge0

```
bridge0: flags=41<UP,RUNNING>
   Configuration:
      priority 32768 hellotime 2 fwddelay 15 maxage 20
   Interfaces:
      en0 flags=3<LEARNING, DISCOVER>
         port 3 priority 128
      ral0 flags=3<LEARNING,DISCOVER>
         port 2 priority 128
   Address cache (max cache: 100, timeout: 1200):
```

WEP Access Point#

NOTE: If you are creating a new wireless network, we recommend you use WPA or WPA2 (RSN) rather than WEP as it supports a much more secure network. WEP should only be used if there are one or more devices on your network which do not support WPA or WPA2.

Enabling WEP network authentication and data encryption is similar to configuring a wireless client as both the WAP and client require the same configuration parameters.

To use your network adapter as a wireless access point, you must first put the network adapter in host access point mode.

- ifconfig ral0 mediaopt hostap

You will also likely need to adjust the media type (link speed) for your wireless adapter as the auto-selected default may not be suitable. You can view all the available media types with the command "ifconfig -m". They will be listed in the supported combinations of media type and media options. For example, if the combination of

- media OFDM54 mode 11g mediaopt hostap

was listed, you could use the command

- ifconfig ral0 media OFDM54 mediaopt hostap

to set the wireless adapter to use 54Mbit/s.

The next parameter to specify is the network name or SSID. This can be up to 32 characters long.

- ifconfig ral0 ssid "my lan"

The final configuration parameter is the WEP key. The WEP key must be either 40 bits long or 104 bits long. You can either enter 5 or 13 characters for the key, or a 10 to 26 digit long hexidecimal value.

- ifconfig ral0 nwkey corpseckey456

You must also set your network interface as up to activate it.

- ifconfig ral0 up

You can also combine all of these commands on one command line

- ifconfig ral0 ssid "my lan" nwkey corpseckey456 mediaopt hostap up

You network should now be marked as up.

- ifconfig ral0

```
ral0: flags=8943<UP,BROADCAST, RUNNING, PROMISC, SIMPLEX, MULTICAST> mtu 1500
   ssid "my lan" apbridge nwkey corpseckey456
   powersave off
   bssid 11:22:33:44:55:66 chan 2
   address: 11:22:33:44:55:66
   media: IEEE802.11 autoselect hostap (autoselect mode 11b hostap)
   status: active
```

WPA Access Point#

WPA/WPA2 support in Neutrino is provided by the hostapd daemon. This daemon is the access point counterpart to the client side wpa_supplicant daemon. This daemon manages your wireless network adapter when in access point mode. The hostapd configuration is defined in the /etc/hostapd.conf configuration file.

Before you start the hostapd process, you must put the network adapter in host access point mode.

- ifconfig ral0 mediaopt hostap

You will also likely need to adjust the media type (link speed) for your wireless adapter as the auto-selected default may not be suitable. You can view all the available media types with the command "ifconfig -m". They will be listed in the supported combinations of media type and media options. For example, if the combination of

- media OFDM54 mode 11g mediaopt hostap

was listed, you could use the command

- ifconfig ral0 media OFDM54 mediaopt hostap

to set the wireless adapter to use 54Mbit/s.

The remainder of the configuration is handled with the hostapd daemon. The daemon will automatically set your network interface as up, so you do not need to do this step with the ifconfig utility. A simple hostapd configuration file is shown below. This configuration uses WPA-PSK for authentication, and AES for data encryption.

- /etc/hostapd.conf

```
interface=ral0
ssid=my home lan
macaddr_acl=0
auth_algs=1
wpa=1
wpa_passphrase=myhomelanpass23456
wpa_key_mgmt=WPA-PSK
wpa_pairwise=CCMP
```

Note that auth_algs and wpa are bitfields, and not values, when you review the hostapd.conf documentation

You can now start the hostapd utility specifying the configuration file

- hostapd -B /etc/hostapd.conf

Using the command ifconfig should show that the network interface is active

- ifconfig ral0

```
ral0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 2290
        ssid "my home lan" apbridge nwkey 2:"",0x49e2a9908872e76b3e5e0c32d09b0b52,0x00000000dc710408c04b32b07c9735b
        powersave off
        bssid 00:15:e9:31:f2:5e chan 4
        address: 00:15:e9:31:f2:5e
        media: IEEE802.11 OFDM54 hostap (OFDM54 mode 11g hostap)
        status: active
```

Your WAP should now be available to your clients.