29.3 Wireless Networking

Loader, Marc Fonvieille, and Murray Stokely.

29.3.1 Wireless Networking Basics

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

802.11 networks are organized in two ways: in *infrastructure mode* one station acts as a master with all the other stations associating to it; the network is known as a BSS and the master station is termed an access point (AP). In a BSS all communication passes through the AP; even when one station wants to communicate with another wireless station messages must go through the AP. In the second form of network there is no master and stations communicate directly. This form of network is termed an IBSS and is commonly known as an *ad-hoc network*.

802.11 networks were first deployed in the 2.4GHz band using protocols defined by the IEEE 802.11 and 802.11b standard. These specifications include the operating frequencies, MAC layer characteristics including framing and transmission rates (communication can be done at various rates). Later the 802.11a standard defined operation in the 5GHz band, including different signalling mechanisms and higher transmission rates. Still later the 802.11g standard was defined to enable use of 802.11a signalling and transmission mechanisms in the 2.4GHz band in such a way as to be backwards compatible with 802.11b networks.

Separate from the underlying transmission techniques 802.11 networks have a variety of security mechanisms. The original 802.11 specifications defined a simple security protocol called WEP. This protocol uses a fixed pre-shared key and the RC4 cryptographic cipher to encode data transmitted on a network. Stations must all agree on the fixed key in order to communicate. This scheme was shown to be easily broken and is now rarely used except to discourage transient users from joining networks. Current security practice is given by the IEEE 802.11i specification that defines new cryptographic ciphers and an additional protocol to authenticate stations to an access point and exchange keys for doing data communication. Further, cryptographic keys are periodically refreshed and there are mechanisms for detecting intrusion attempts (and for countering intrusion attempts). Another security protocol specification commonly used in wireless networks is termed WPA. This was a precursor to 802.11i defined by an industry group as an interim measure while waiting for 802.11i to be ratified. WPA specifies a subset of the requirements found in 802.11i and is designed for implementation on legacy hardware. Specifically WPA requires only the TKIP cipher that is derived from the original WEP cipher. 802.11i permits use of TKIP but also requires support for a stronger cipher, AES-CCM, for encrypting data. (The AES cipher was not required in WPA because it was deemed too computationally costly to be implemented on legacy hardware.)

Other than the above protocol standards the other important standard to be aware of is 802.11e. This defines protocols for deploying multi-media applications such as streaming video and voice over IP (VoIP) in an 802.11 network. Like 802.11i, 802.11e also has a precursor specification termed WME (later renamed WMM) that has been defined by an industry group as a subset of 802.11e that can be deployed now to enable multi-media applications while waiting for the final ratification of 802.11e. The most important thing to know about 802.11e and WME/WMM is that it enables prioritized traffic use of a wireless network through Quality of Service (QoS) protocols and enhanced media access protocols. Proper implementation of these protocols enable high speed bursting of data and prioritized traffic flow.

Since the 6.0 version, FreeBSD supports networks that operate using 802.11a, 802.11b, and 802.11g. The WPA and 802.11i security protocols are likewise supported (in conjunction with any of 11a, 11b, and 11g) and QoS and traffic prioritization required by the WME/WMM protocols are supported for a limited set of wireless devices.

29.3.2 Basic Setup

29.3.2.1 Kernel Configuration

To use wireless networking you need a wireless networking card and to configure the kernel with the appropriate wireless networking support. The latter is separated into multiple modules so that you only need to configure the software you are actually going to use.

The first thing you need is a wireless device. The most commonly used devices are those that use parts made by Atheros. These devices are supported by the ath(4) driver and require the following line to be added to the `/boot/loader.conf` file:

```
if_ath_load="YES"
```

The Atheros driver is split up into three separate pieces: the driver proper (ath(4)), the hardware support layer that handles chip-specific functions (ath_hal(4)), and an algorithm for selecting which of several possible rates for transmitting frames (ath_rate_sample here). When you load this support as modules these dependencies are automatically handled for you. If instead of an Atheros device you had another device you would select the module for that device; e.g.:

```
if_wi_load="YES"
```

for devices based on the Intersil Prism parts (wi(4) driver).

Note: In the rest of this document, we will use an ath(4) device, the device name in the examples must be changed according to your configuration. A list of available wireless drivers can be found at the beginning of the wlan(4) manual page. If a native FreeBSD driver for your wireless device does not exist, it may be possible to directly use the Windows® driver with the help of the NDIS driver wrapper.

With a device driver configured you need to also bring in the 802.11 networking support required by the driver. For the ath(4) driver this is at least the wlan(4) module; this module is automatically loaded with the wireless device driver. With that you will need the modules that implement cryptographic support for the security protocols you intend to use. These are intended to be dynamically loaded on demand by the wlan(4) module but for now they must be manually configured. The following modules are available: wlan_wep(4), wlan_ccmp(4) and wlan_tkip(4). Both wlan_ccmp(4) and wlan_tkip(4) drivers are only needed if you intend to use the WPA and/or 802.11i security protocols. If your network is to run totally open (i.e., with no encryption) then you do not even need the wlan_wep(4) support. To load these modules at boot time, add the following lines to `/boot/loader.conf`:

```
wlan_wep_load="YES"

wlan_ccmp_load="YES"

wlan_tkip_load="YES"
```

With this information in the system bootstrap configuration file (i.e., `/boot/loader.conf`), you have to reboot your FreeBSD box. If you do not want to reboot your machine for the moment, you can just load the modules by hand using kldload(8).

Note: If you do not want to use modules, it is possible to compile these drivers into the kernel by adding the following lines to your kernel configuration file:

```
device ath              # Atheros IEEE 802.11 wireless network
driver

device ath_hal           # Atheros Hardware Access Layer

device ath_rate_sample   # John Bicket's SampleRate control
algorithm.

device wlan             # 802.11 support (Required)

device wlan_wep          # WEP crypto support for 802.11 devices

device wlan_ccmp         # AES-CCMP crypto support for 802.11
devices

device wlan_tkip         # TKIP and Michael crypto support for
802.11 devices
```

With this information in the kernel configuration file, recompile the kernel and reboot your FreeBSD machine.

When the system is up, we could find some information about the wireless device in the boot messages, like this:

```
ath0: <Atheros 5212> mem 0xff9f0000-0xff9fffff irq 17 at
device 2.0 on pci2

ath0: Ethernet address: 00:11:95:d5:43:62

ath0: mac 7.9 phy 4.5 radio 5.6
```

29.3.3 Infrastructure Mode

The infrastructure mode or BSS mode is the mode that is typically used. In this mode, a number of wireless access points are connected to a wired network. Each wireless network has its own name, this name is called the SSID of the network. Wireless clients connect to the wireless access points.

29.3.3.1 FreeBSD Clients

29.3.3.1.1 How to Find Access Points

To scan for networks, use the `ifconfig` command. This request may take a few moments to complete as it requires that the system switches to each available wireless frequency and probes for available access points. Only the super-user can initiate such a scan:

```
# ifconfig ath0 up scan
SSID            BSSID              CHAN RATE  S:N   INT CAPS
dlinkap          00:13:46:49:41:76    6   54M 29:0   100 EPS  WPA
WME
freebsdap        00:11:95:c3:0d:ac    1   54M 22:0   100 EPS  WPA
```

Note: You must mark the interface `up` before you can scan. Subsequent scan requests do not require you to mark the interface up again.

The output of a scan request lists each BSS/IBSS network found. Beside the name of the network, `SSID`, we find the `BSSID` which is the MAC address of the access point. The `CAPS` field identifies the type of each network and the capabilities of the stations operating there:

E

Extended Service Set (ESS). Indicates that the station is part of an infrastructure network (in contrast to an IBSS/ad-hoc network).

I

IBSS/ad-hoc network. Indicates that the station is part of an ad-hoc network (in contrast to an ESS network).

P

Privacy. Data confidentiality is required for all data frames exchanged within the BSS. This means that this BSS requires the station to use cryptographic means such as WEP, TKIP or AES-CCMP to encrypt/decrypt data frames being exchanged with others.

S

Short Preamble. Indicates that the network is using short preambles (defined in 802.11b High Rate/DSSS PHY, short preamble utilizes a 56 bit sync field in contrast to a 128 bit field used in long preamble mode).

s

Short slot time. Indicates that the 802.11g network is using a short slot time because there are no legacy (802.11b) stations present.

One can also display the current list of known networks with:

```
# ifconfig ath0 list scan
```

This information may be updated automatically by the adapter or manually with a `scan` request. Old data is automatically removed from the cache, so over time this list may shrink unless more scans are done.

29.3.3.1.2 Basic Settings

This section provides a simple example of how to make the wireless network adapter work in FreeBSD without encryption. After you are familiar with these concepts, we strongly recommend using WPA to set up your wireless network.

There are three basic steps to configure a wireless network: selecting an access point, authenticating your station, and configuring an IP address. The following sections discuss each step.

29.3.3.1.2.1 Selecting an Access Point

Most of time it is sufficient to let the system choose an access point using the builtin heuristics. This is the default behaviour when you mark an interface up or otherwise configure an interface by listing it in `/etc/rc.conf`, e.g.:

```
ifconfig_ath0="DHCP"
```

If there are multiple access points and you want to select a specific one, you can select it by its SSID:

```
ifconfig_ath0="ssid your_ssid_here DHCP"
```

In an environment where there are multiple access points with the same SSID (often done to simplify roaming) it may be necessary to associate to one specific device. In this case you can also specify the BSSID of the access point (you can also leave off the SSID):

```
ifconfig_ath0="ssid your_ssid_here bssid xx:xx:xx:xx:xx:xx DHCP"
```

There are other ways to constrain the choice of an access point such as limiting the set of frequencies the system will scan on. This may be useful if you have a multi-band wireless card as scanning all the possible channels can be time-consuming. To limit operation to a specific band you can use the `mode` parameter; e.g.:

```
ifconfig_ath0="mode 11g ssid your_ssid_here DHCP"
```

will force the card to operate in 802.11g which is defined only for 2.4GHz frequencies so any 5GHz channels will not be considered. Other ways to do this are the `channel` parameter, to lock operation to one specific frequency, and the `chanlist` parameter, to specify a list of channels for scanning. More information about these parameters can be found in the ifconfig(8) manual page.

29.3.3.1.2.2 Authentication

Once you have selected an access point your station needs to authenticate before it can pass data. Authentication can happen in several ways. The most common scheme used is termed open authentication and allows any station to join the network and communicate. This is the authentication you should use for test purpose the first time you set up a wireless network. Other schemes require cryptographic handshakes 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. Most users will use open authentication which is the default setting. Next most common setup is WPA-PSK, also known as WPA Personal, which is described below.

Note: If you have an Apple® AirPort® Extreme base station for an access point you may need to configure shared-key authentication together with a WEP key. This can be done in the `/etc/rc.conf` file or using the wpa_supplicant(8) program. If you have a single AirPort base station you can setup access with something like:

```
ifconfig_ath0="authmode shared wepmode on weptxkey 1 wepkey
01234567 DHCP"
```

In general shared key authentication is to be avoided because it uses the WEP key material in a highly-constrained manner making it even easier to crack the key. If WEP must be used (e.g., for compatibility with legacy devices) it is better to use WEP with `open` authentication. More information regarding WEP can be found in the Section 29.3.3.1.4.

29.3.3.1.2.3 Getting an IP Address with DHCP

Once you have selected an access point and set the authentication parameters, you will have to get an IP address to communicate. Most of time you will obtain your wireless IP address via DHCP. To achieve that, simply edit `/etc/rc.conf` and add `DHCP` to the configuration for your device as shown in various examples above:

```
ifconfig_ath0="DHCP"
```

At this point, you are ready to bring up the wireless interface:

```
# /etc/rc.d/netif start
```

Once the interface is running, use `ifconfig` to see the status of the interface `ath0`:

```
# ifconfig ath0

ath0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu
1500

       inet6 fe80::211:95ff:fed5:4362%ath0 prefixlen 64 scopeid
0x1

       inet 192.168.1.100 netmask 0xffffff00 broadcast
192.168.1.255
```

```
	ether 00:11:95:d5:43:62
	media: IEEE 802.11 Wireless Ethernet autoselect
(OFDM/54Mbps)
	status: associated
	ssid dlinkap channel 6 bssid 00:13:46:49:41:76
	authmode OPEN privacy OFF txpowmax 36 protmode CTS
bintval 100
```

The `status: associated` means you are connected to the wireless network (to the `dlinkap` network in our case). The `bssid 00:13:46:49:41:76` part is the MAC address of your access point; the `authmode` line informs you that the communication is not encrypted (`OPEN`).

29.3.3.1.2.4 Static IP Address

In the case you cannot obtain an IP address from a DHCP server, you can set a fixed IP address. Replace the `DHCP` keyword shown above with the address information. Be sure to retain any other parameters you have set up for selecting an access point:

```
ifconfig_ath0="inet 192.168.1.100 netmask 255.255.255.0 ssid
your_ssid_here"
```

29.3.3.1.3 WPA

WPA (Wi-Fi Protected Access) is a security protocol used together with 802.11 networks to address the lack of proper authentication and the weakness of WEP. WPA leverages the 802.1X authentication protocol and uses one of several ciphers instead of WEP for data integrity. The only cipher required by WPA is TKIP (Temporary Key Integrity Protocol) which is a cipher that extends the basic RC4 cipher used by WEP by adding integrity checking, tamper detection, and measures for responding to any detected intrusions. TKIP is designed to work on legacy hardware with only software modification; it represents a compromise that improves security but is still not entirely immune to attack. WPA also specifies the AES-CCMP cipher as an alternative to TKIP and that is preferred when possible; for this specification the term WPA2 (or RSN) is commonly used.

WPA defines authentication and encryption protocols. Authentication is most commonly done using one of two techniques: by 802.1X and a backend authentication service such as RADIUS, or by a minimal handshake between the station and the access point using a pre-shared secret. The former is commonly termed WPA Enterprise with the latter known as WPA Personal. Since most people will not set up a RADIUS backend server for wireless network, WPA-PSK is by far the most commonly encountered configuration for WPA.

The control of the wireless connection and the authentication (key negotiation or authentication with a server) is done with the wpa_supplicant(8) utility. This program requires a configuration file, `/etc/wpa_supplicant.conf`, to run. More information regarding this file can be found in the wpa_supplicant.conf(5) manual page.

29.3.3.1.3.1 WPA-PSK

WPA-PSK also known as WPA-Personal is based on a pre-shared key (PSK) generated from a given password and that will be used as the master key in the wireless network. This means every wireless user will share the same key. WPA-PSK is intended for small networks where the use of an authentication server is not possible or desired.

Warning: Always use strong passwords that are sufficiently long and made from a rich alphabet so they will not be guessed and/or attacked.

The first step is the configuration of the /etc/wpa_supplicant.conf file with the SSID and the pre-shared key of your network:

```
network={
  ssid="freebsdap"
  psk="freebsdmall"
}
```

Then, in /etc/rc.conf, we indicate that the wireless device configuration will be done with WPA and the IP address will be obtained with DHCP:

```
ifconfig_ath0="WPA DHCP"
```

Then, we can bring up the interface:

```
# /etc/rc.d/netif start
Starting wpa_supplicant.
DHCPDISCOVER on ath0 to 255.255.255.255 port 67 interval 5
DHCPDISCOVER on ath0 to 255.255.255.255 port 67 interval 6
DHCPOFFER from 192.168.0.1
DHCPREQUEST on ath0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.1
bound to 192.168.0.254 -- renewal in 300 seconds.
ath0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu
1500
     inet6 fe80::211:95ff:fed5:4362%ath0 prefixlen 64 scopeid
0x1
     inet 192.168.0.254 netmask 0xffffff00 broadcast
192.168.0.255
     ether 00:11:95:d5:43:62
     media: IEEE 802.11 Wireless Ethernet autoselect
(OFDM/36Mbps)
     status: associated
     ssid freebsdap channel 1 bssid 00:11:95:c3:0d:ac
     authmode WPA privacy ON deftxkey UNDEF TKIP 2:128-bit
txpowmax 36

     protmode CTS roaming MANUAL bintval 100
```

Or you can try to configure it manually using the same /etc/wpa_supplicant.conf above, and run:

```
# wpa_supplicant -i ath0 -c /etc/wpa_supplicant.conf

Trying to associate with 00:11:95:c3:0d:ac (SSID='freebsdap'
freq=2412 MHz)

Associated with 00:11:95:c3:0d:ac

WPA: Key negotiation completed with 00:11:95:c3:0d:ac
[PTK=TKIP GTK=TKIP]
```

The next operation is the launch of the dhclient command to get the IP address from the DHCP server:

```
# dhclient ath0

DHCPREQUEST on ath0 to 255.255.255.255 port 67

DHCPACK from 192.168.0.1

bound to 192.168.0.254 -- renewal in 300 seconds.

# ifconfig ath0

ath0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu
1500

     inet6 fe80::211:95ff:fed5:4362%ath0 prefixlen 64 scopeid
0x1

     inet 192.168.0.254 netmask 0xffffff00 broadcast
192.168.0.255

     ether 00:11:95:d5:43:62

     media: IEEE 802.11 Wireless Ethernet autoselect
(OFDM/48Mbps)

     status: associated

     ssid freebsdap channel 1 bssid 00:11:95:c3:0d:ac

     authmode WPA privacy ON deftxkey UNDEF TKIP 2:128-bit
txpowmax 36
```

```
     protmode CTS roaming MANUAL bintval 100
```

Note: If the /etc/rc.conf is set up with the line ifconfig_ath0="DHCP" then it is no need to run the dhclient command manually, dhclient will be launched after wpa_supplicant plumbs the keys.

In the case where the use of DHCP is not possible, you can set a static IP address after wpa_supplicant has authenticated the station:

```
# ifconfig ath0 inet 192.168.0.100 netmask 255.255.255.0

# ifconfig ath0

ath0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu
1500

     inet6 fe80::211:95ff:fed5:4362%ath0 prefixlen 64 scopeid
0x1

     inet 192.168.0.100 netmask 0xffffff00 broadcast
192.168.0.255

     ether 00:11:95:d5:43:62

     media: IEEE 802.11 Wireless Ethernet autoselect
(OFDM/36Mbps)

     status: associated

     ssid freebsdap channel 1 bssid 00:11:95:c3:0d:ac

     authmode WPA privacy ON deftxkey UNDEF TKIP 2:128-bit
txpowmax 36

     protmode CTS roaming MANUAL bintval 100
```

When DHCP is not used, you also have to manually set up the default gateway and the nameserver:

```
# route add default your_default_router

# echo "nameserver your_DNS_server" >> /etc/resolv.conf
```

29.3.3.1.3.2 WPA with EAP-TLS

The second way to use WPA is with an 802.1X backend authentication server, in this case WPA is called WPA-Enterprise to make difference with the less secure WPA-Personal with its pre-shared key. The authentication in WPA-Enterprise is based on EAP (Extensible Authentication Protocol).

EAP does not come with an encryption method, it was decided to embed EAP inside an encrypted tunnel. Many types of EAP authentication methods have been designed, the most common methods are EAP-TLS, EAP-TTLS and EAP-PEAP.

EAP-TLS (EAP with Transport Layer Security) is a very well-supported authentication protocol in the wireless world since it was the first EAP method to be certified by the Wi-Fi alliance. EAP-TLS will require three certificates to run: the CA certificate (installed on all machines), the server certificate for your authentication server, and one client certificate for each wireless client. In this EAP method, both authentication server and wireless client authenticate each other in presenting their respective certificates, and they verify that these certificates were signed by your organization's certificate authority (CA).

As previously, the configuration is done via `/etc/wpa_supplicant.conf`:

```
network={
  ssid="freebsdap" ❶
  proto=RSN  ❷
  key_mgmt=WPA-EAP ❸
  eap=TLS ❹
  identity="loader" ❺
  ca_cert="/etc/certs/cacert.pem" ❻
  client_cert="/etc/certs/clientcert.pem" ❼
  private_key="/etc/certs/clientkey.pem" ❽
  private_key_passwd="freebsdmallclient" ❾
}
```

❶

This field indicates the network name (SSID).

❷

Here, we use RSN (IEEE 802.11i) protocol, i.e., WPA2.

❸

The `key_mgmt` line refers to the key management protocol we use. In our case it is WPA using EAP authentication: `WPA-EAP`.

❹

In this field, we mention the EAP method for our connection.

❺

The `identity` field contains the identity string for EAP.

❻

The `ca_cert` field indicates the pathname of the CA certificate file. This file is needed to verify the server certificat.

❼

The `client_cert` line gives the pathname to the client certificate file. This certificate is unique to each wireless client of the network.

❽

The `private_key` field is the pathname to the client certificate private key file.

❾

The `private_key_passwd` field contains the passphrase for the private key.

Then add the following line to `/etc/rc.conf`:

```
ifconfig_ath0="WPA DHCP"
```

The next step is to bring up the interface with the help of the `rc.d` facility:

```
# /etc/rc.d/netif start
Starting wpa_supplicant.
DHCPREQUEST on ath0 to 255.255.255.255 port 67
DHCPREQUEST on ath0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.20
bound to 192.168.0.254 -- renewal in 300 seconds.
ath0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu
1500
     inet6 fe80::211:95ff:fed5:4362%ath0 prefixlen 64 scopeid
0x1
     inet 192.168.0.254 netmask 0xffffff00 broadcast
192.168.0.255
     ether 00:11:95:d5:43:62
     media: IEEE 802.11 Wireless Ethernet autoselect
(DS/11Mbps)
     status: associated
     ssid freebsdap channel 1 bssid 00:11:95:c3:0d:ac
     authmode WPA2/802.11i privacy ON deftxkey UNDEF TKIP
2:128-bit
     txpowmax 36 protmode CTS roaming MANUAL bintval 100
```

As previously shown, it is also possible to bring up the interface manually with both `wpa_supplicant` and `ifconfig` commands.

29.3.3.1.3.3 WPA with EAP-TTLS

With EAP-TLS both the authentication server and the client need a certificate, with EAP-TTLS (EAP-Tunneled Transport Layer Security) a client certificate is optional. This method is close to what some secure web sites do , where the web server can create a secure SSL tunnel even if the visitors do not have client-side certificates. EAP-TTLS will use the encrypted TLS tunnel for safe transport of the authentication data.

The configuration is done via the `/etc/wpa_supplicant.conf` file:

```
network={
  ssid="freebsdap"
  proto=RSN
  key_mgmt=WPA-EAP
  eap=TTLS ❶
  identity="test" ❷
  password="test" ❸
  ca_cert="/etc/certs/cacert.pem" ❹
  phase2="auth=MD5" ❺
}
```

❶

In this field, we mention the EAP method for our connection.

❷

The `identity` field contains the identity string for EAP authentication inside the encrypted TLS tunnel.

❸

The `password` field contains the passphrase for the EAP authentication.

❹

The `ca_cert` field indicates the pathname of the CA certificate file. This file is needed to verify the server certificat.

❺

In this field, we mention the authentication method used in the encrypted TLS tunnel. In our case, EAP with MD5-Challenge has been used. The “inner authentication” phase is often called “phase2”.

You also have to add the following line to /etc/rc.conf:

```
ifconfig_ath0="WPA DHCP"
```

The next step is to bring up the interface:

```
# /etc/rc.d/netif start
Starting wpa_supplicant.
DHCPREQUEST on ath0 to 255.255.255.255 port 67
DHCPREQUEST on ath0 to 255.255.255.255 port 67
DHCPREQUEST on ath0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.20
bound to 192.168.0.254 -- renewal in 300 seconds.
ath0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu
1500
     inet6 fe80::211:95ff:fed5:4362%ath0 prefixlen 64 scopeid
0x1
     inet 192.168.0.254 netmask 0xffffff00 broadcast
192.168.0.255
     ether 00:11:95:d5:43:62
     media: IEEE 802.11 Wireless Ethernet autoselect
(DS/11Mbps)
     status: associated
     ssid freebsdap channel 1 bssid 00:11:95:c3:0d:ac
     authmode WPA2/802.11i privacy ON deftxkey UNDEF TKIP
2:128-bit
     txpowmax 36 protmode CTS roaming MANUAL bintval 100
```

29.3.3.1.3.4 WPA with EAP-PEAP

PEAP (Protected EAP) has been designed as an alternative to EAP-TTLS. There are two types of PEAP methods, the most common one is

PEAPv0/EAP-MSCHAPv2. In the rest of this document, we will use the PEAP term to refer to that EAP method. PEAP is the most used EAP standard after EAP-TLS, in other words if you have a network with mixed OSes, PEAP should be the most supported standard after EAP-TLS.

PEAP is similar to EAP-TTLS: it uses a server-side certificate to authenticate clients by creating an encrypted TLS tunnel between the client and the authentication server, which protects the ensuing exchange of authentication information. In term of security the difference between EAP-TTLS and PEAP is that PEAP authentication broadcasts the username in clear, only the password is sent in the encrypted TLS tunnel. EAP-TTLS will use the TLS tunnel for both username and password.

We have to edit the `/etc/wpa_supplicant.conf` file and add the EAP-PEAP related settings:

```
network={
  ssid="freebsdap"
  proto=RSN
  key_mgmt=WPA-EAP
  eap=PEAP ❶
  identity="test" ❷
  password="test" ❸
  ca_cert="/etc/certs/cacert.pem" ❹
  phase1="peaplabel=0" ❺
  phase2="auth=MSCHAPV2" ❻
}
```

❶

In this field, we mention the EAP method for our connection.

❷

The `identity` field contains the identity string for EAP authentication inside the encrypted TLS tunnel.

❸

The `password` field contains the passphrase for the EAP authentication.

❹

The `ca_cert` field indicates the pathname of the CA certificate file. This file is needed to verify the server certificat.

❺

This field contains the parameters for the first phase of the authentication (the TLS tunnel). According to the authentication server used, you will have to specify a specific label for the authentication. Most of time, the label will be "client EAP encryption" which is set by using `peaplabel=0`. More information can be found in the wpa_supplicant.conf(5) manual page.

❻

In this field, we mention the authentication protocol used in the encrypted TLS tunnel. In the case of PEAP, it is `auth=MSCHAPV2`.

The following must be added to `/etc/rc.conf`:

```
ifconfig_ath0="WPA DHCP"
```

Then, we can bring up the interface:

```
# /etc/rc.d/netif start
Starting wpa_supplicant.
DHCPREQUEST on ath0 to 255.255.255.255 port 67
DHCPREQUEST on ath0 to 255.255.255.255 port 67
DHCPREQUEST on ath0 to 255.255.255.255 port 67
```

```
DHCPACK from 192.168.0.20
bound to 192.168.0.254 -- renewal in 300 seconds.
ath0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu
1500
     inet6 fe80::211:95ff:fed5:4362%ath0 prefixlen 64 scopeid
0x1
     inet 192.168.0.254 netmask 0xffffff00 broadcast
192.168.0.255
     ether 00:11:95:d5:43:62
     media: IEEE 802.11 Wireless Ethernet autoselect
(DS/11Mbps)
     status: associated
     ssid freebsdap channel 1 bssid 00:11:95:c3:0d:ac
     authmode WPA2/802.11i privacy ON deftxkey UNDEF TKIP
2:128-bit
     txpowmax 36 protmode CTS roaming MANUAL bintval 100
```

29.3.3.1.4 WEP

WEP (Wired Equivalent Privacy) is part of the original 802.11 standard. There is no authentication mechanism, only a weak form of access control, and it is easily to be cracked.

WEP can be set up with `ifconfig`:

```
# ifconfig ath0 inet 192.168.1.100 netmask 255.255.255.0 ssid my_net ₩
       wepmode on weptxkey 3 wepkey 3:0x3456789012
```

- The `weptxkey` means which WEP key will be used in the transmission. Here we used the third key. This must match the setting in the access point.
- The `wepkey` means setting the selected WEP key. It should in the format *index:key*, if the index is not given, key 1 is set. That

is to say we need to set the index if we use keys other than the first key.

> **Note:** You must replace the `0x3456789012` with the key configured for use on the access point.

You are encouraged to read ifconfig(8) manual page for further information.

The `wpa_supplicant` facility also can be used to configure your wireless interface with WEP. The example above can be set up by adding the following lines to `/etc/wpa_supplicant.conf`:

```
network={
  ssid="my_net"
  key_mgmt=NONE
  wep_key3=3456789012
  wep_tx_keyidx=3
}
```

Then:

```
# wpa_supplicant -i ath0 -c /etc/wpa_supplicant.conf
Trying to associate with 00:13:46:49:41:76 (SSID='dlinkap'
freq=2437 MHz)
Associated with 00:13:46:49:41:76
```

29.3.4 Ad-hoc Mode

IBSS mode, also called ad-hoc mode, is designed for point to point connections. For example, to establish an ad-hoc network between the machine `A` and the machine `B` we will just need to choose two IP adresses and a SSID.

On the box A:

```
# ifconfig ath0 inet 192.168.0.1 netmask 255.255.255.0 ssid freebsdap
mediaopt adhoc

# ifconfig ath0

  ath0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu
1500

     inet 192.168.0.1 netmask 0xffffff00 broadcast
192.168.0.255

     inet6 fe80::211:95ff:fec3:dac%ath0 prefixlen 64 scopeid
0x4

     ether 00:11:95:c3:0d:ac

     media: IEEE 802.11 Wireless Ethernet autoselect <adhoc>
(autoselect <adhoc>)

     status: associated

     ssid freebsdap channel 2 bssid 02:11:95:c3:0d:ac

     authmode OPEN privacy OFF txpowmax 36 protmode CTS
bintval 100
```

The adhoc parameter indicates the interface is running in the IBSS mode.

On B, we should be able to detect A:

```
# ifconfig ath0 up scan

  SSID           BSSID              CHAN RATE  S:N   INT CAPS

  freebsdap       02:11:95:c3:0d:ac    2   54M 19:0   100 IS
```

The I in the output confirms the machine A is in ad-hoc mode. We just have to configure B with a different IP address:

```
# ifconfig ath0 inet 192.168.0.2 netmask 255.255.255.0 ssid freebsdap
mediaopt adhoc

# ifconfig ath0

  ath0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu
1500
```

```
     inet6 fe80::211:95ff:fed5:4362%ath0 prefixlen 64 scopeid
0x1
     inet 192.168.0.2 netmask 0xffffff00 broadcast
192.168.0.255
     ether 00:11:95:d5:43:62
     media: IEEE 802.11 Wireless Ethernet autoselect <adhoc>
(autoselect <adhoc>)
     status: associated
     ssid freebsdap channel 2 bssid 02:11:95:c3:0d:ac
     authmode OPEN privacy OFF txpowmax 36 protmode CTS
bintval 100
```

Both `A` and `B` are now ready to exchange informations.

29.3.5 Troubleshooting

If you are having trouble with wireless networking, there are a number of steps you can take to help troubleshoot the problem.

- If you do not see the access point listed when scanning be sure you have not configured your wireless device to a limited set of channels.
- If you cannot associate to an access point verify the configuration of your station matches the one of the access point. This includes the authentication scheme and any security protocols. Simplify your configuration as much as possible. If you are using a security protocol such as WPA or WEP configure the access point for open authentication and no security to see if you can get traffic to pass.
- Once you can associate to the access point diagnose any security configuration using simple tools like ping(8).

The wpa_supplicant has much debugging support; try running it manually with the -dd option and look at the system logs.

- There are also many lower-level debugging tools. You can enable debugging messages in the 802.11 protocol support layer using the wlandebug program found in /usr/src/tools/tools/net80211. For example:

```
# wlandebug -i ath0 +scan+auth+debug+assoc
  net.wlan.0.debug: 0 => 0xc80000<assoc,auth,scan>
```

  can be used to enable console messages related to scanning for access points and doing the 802.11 protocol handshakes required to arrange communication.

  There are also many useful statistics maintained by the 802.11 layer; the wlanstats tool will dump these informations. These statistics should identify all errors identified by the 802.11 layer. Beware however that some errors are identified in the device drivers that lie below the 802.11 layer so they may not show up. To diagnose device-specific problems you need to refer to the drivers' documentation.

If the above information does not help to clarify the problem, please submit a problem report and include output from the above tools.