WLAN (IEEE 802.11) capture setup

The following will explain capturing on 802.11 wireless networks (WLAN).

If you are only trying to capture network traffic between the machine running Wireshark or TShark and other machines on the network, are only interested in regular network data, rather than 802.11 management or control packets, and are not interested in radio-layer information about packets such as signal strength and data rates, you should be able to do this by capturing on the network interface through which the packets will be transmitted and received; no special setup should be necessary. (If you're trying to capture network traffic between processes running on the machine running Wireshark or TShark, i.e. network traffic from that machine to itself, you will need to capture on a loopback interface, if that's possible; see CaptureSetup/Loopback.)

If you're trying to capture network traffic that's *not* being sent to or from the machine running Wireshark or TShark, i.e. traffic between two or more other machines on an Ethernet segment, or are interested in 802.11 management or control packets, or are interested in radio-layer information about packets, you will probably have to capture in "monitor mode". This is discussed below.

Without any interaction, capturing on WLAN's may capture only *user data* packets with "fake" Ethernet headers. In this case, you won't see any 802.11 management or control packets at all, and the 802.11 packet headers are "translated" by the network driver to "fake" Ethernet packet headers.

A 802.11 LAN uses a "broadcast medium", much like (the mostly obsolete shared) Ethernet. Compared to Ethernet, the 802.11 network is even "broader", as the transmitted packets are not limited by the cable medium. That's one of the reasons why the 802.11 network adapters have two additional mechanisms to ignore unwanted packets at the receiving side: channels and SSID's.

Conclusion: the packets you'll be capturing with default settings might be modified, and only a limited number of the packets transmitted through the WLAN.

The following will provide some 802.11 network details, and will describe how to disable the translation/filtering and see what's "really" going on inside your WLAN.

Unfortunately, changing the 802.11 capture modes is very platform/network adapter/driver/libpcap dependent, and might not be possible at all (Windows is very limited here).

Table of contents

Contents

1. WLAN (IEEE 802.11) capture setup
    1. Table of contents
    2. Packet Types
        1. Link-Layer (Radio) packet headers
        2. Data Packets
        3. Non-data packets
            1. Management Packets
            2. Low-level Control Packets
    3. 802.11 Filter (Modes)
        1. Channels (Frequencies)
        2. SSID/ESSID (Network Name)
            1. Monitor mode
        3. MAC Addresses
            1. Promiscuous mode
    4. Turning on monitor mode
        1. *BSD
            1. DragonFly BSD
        2. Linux
        3. Mac OS X
        4. Windows
            1. Starting from Windows Vista: Npcap
            2. WinPcap
            3. AirPcap
            4. Intel Centrino adapters
        5. Channel Hopping
2. Discussion
    1. See Also

Packet Types

802.11 traffic includes data packets, which are the packets used for normal network protocols; it also includes management packets and low-level control packets.

The 802.11 hardware on the network adapter filters all packets received, and delivers to the host

- all Unicast packets that are being sent to one of the addresses for that adapter, i.e. packets sent to that host on that network;
- all Multicast packets that are being sent to a Multicast address for that adapter, or all Multicast packets regardless of the address to which they're being sent (some network adapters can be configured to accept packets for specific Multicast addresses, others deliver all multicast packets to the host for it to filter);
- all Broadcast packets.

The driver for the adapter will also send copies of transmitted packets to the packet capture mechanism, so that they will be seen by a capture program as well.

In order to capture 802.11 traffic other than Unicast traffic to and from the host on which you're running Wireshark, Multicast traffic, and Broadcast traffic, the adapter will have to be put into monitor mode, so that the filter mentioned above is switched off and all packets received are delivered to the host. Promiscuous mode is, in theory, possible on many 802.11 adapters, but often does not work in practice; if you specify promiscuous mode, the attempt to enable promiscuous mode may fail, the adapter might only capture traffic to and from your machine, or the adapter might not capture *any* packets.

When not in monitor mode, the adapter might only capture data packets; you may have to put the adapter into monitor mode to capture management and control packets. In addition, when not in monitor mode, the adapter might supply packets with fake Ethernet headers, rather than 802.11 headers, and might not supply additional radio-layer information such as data rates and signal strength. You may have to perform operating-system-dependent and adapter-type-dependent operations to enable monitor mode; information on how to do so is given below.

On some platforms, such as FreeBSD, you may be able to capture non-data packets, and see 802.11 headers rather than fake Ethernet headers, without going into monitor mode, by selecting an 802.11 link-layer header type, rather than Ethernet, when capturing; however, that might not show both incoming and outgoing traffic.

Link-Layer (Radio) packet headers

802.11 adapters often transform 802.11 data packets into fake Ethernet packets before supplying them to the host, and, even if they don't, the drivers for the adapters often do so before supplying the packets to the operating system's networking stack and packet capture mechanism.

This means that if you capture on an 802.11 network, the packets will look like Ethernet packets, and you won't be able to see all the fields in the 802.11 header.

On some platforms, you can request that 802.11 headers be supplied when capturing, at least with some 802.11 adapters, regardless of whether you capture in monitor mode, sometimes called "rfmon mode" (see below); on some other platforms, you will get 802.11 headers in monitor mode, and only in monitor mode.

In addition, on some platforms, at least with some 802.11 adapters, you can get radio headers, supplying information such as signal strength, in addition to 802.11 headers. On some of those platforms, the radio headers are available whether you are capturing in monitor mode or not; on other platforms, they are only available in monitor mode. In Wireshark 1.4 and later, when built with libpcap 1.0 or later, there may be a "Monitor mode" check box in the "Capture Options" dialog to capture in monitor mode, and the command-line option `-I` to dumpcap, TShark, and Wireshark may be used to capture in monitor mode. However, due to problems with libpcap 1.0.x and libpcap 1.1.x, and due to the way libpcap 1.1.x is built on some Linux distributions, the check box and `-I` flag might not work on those distributions; see the "Turning on monitor mode" section below for information on how to capture in monitor mode if the check box and `-I` flag are either not available or don't work.

In FreeBSD 5.2 and later, NetBSD 2.0 and later, OpenBSD 3.7 and later, and DragonFly BSD 1.2 and later, you do not have to capture in monitor mode to get 802.11 headers, except when capturing on a Cisco Aironet adapter in FreeBSD. For earlier releases of those BSDs, 802.11 headers are not supported, except perhaps when capturing on a Cisco Aironet adapter in FreeBSD.

On Linux and Mac OS X, you can only get 802.11 headers in monitor mode.

To see 802.11 headers for frames, without radio information, you should:

- in Wireshark, if you're starting the capture from the GUI, select "802.11" as the "Link-layer header type" in the "Capture Options" dialog;
- in dumpcap or TShark, or in Wireshark if you're starting the capture from the command line, add the argument `-y IEEE802_11` to the command.

If 802.11 headers are not available for your 802.11 adapter on your platform at all, "802.11" will not be offered as a link-layer header type, and attempts to use `-y IEEE802_11` even if the "Monitor mode" checkbox, if present, is checked, or if `-I` is specified on the command line. If they are only available in monitor mode, "802.11" will only be offered if the "Monitor mode" checkbox is checked or `-I` is specified on the command line.

For Wireshark 1.4 and later, when built with libpcap 1.0 or later, to determine from the command line what

link-layer header types are available for an interface in monitor mode, run one of

```
dumpcap -i interface -I -L
tshark -i interface -I -L
wireshark -i interface -I -L
```

Omit the `-I` to see what link-layer header types are available when not in monitor mode. For earlier versions of Wireshark, or versions of Wireshark built with earlier versions of libpcap, the `-I` flag is not specified; on Linux, you will have to put the adapter into monitor mode yourself (see below) to see what link-layer header types are available in monitor mode, and, in Mac OS X Leopard and later, selecting 802.11 headers will put the adapter in monitor mode.

To see 802.11 headers for frames, with radio information, you should:

- in Wireshark, if you're starting the capture from the GUI, select one of "802.11 plus BSD radio information header", "802.11 plus AVS radio information", or "802.11 plus Prism header" as the "Link-layer header type", if one or more of them are available (they won't necessarily be available for all interfaces supporting monitor mode);
- in dumpcap or TShark, or in Wireshark if you're starting the capture from the command line, add the argument `-y IEEE802_11_RADIO`, `-y IEEE802_11_RADIO_AVS`, or `-y PRISM` to the command - to see which of those are supported, run to see which are supported.

If 802.11+radio headers are not available for your 802.11 adapter on your platform at all, "802.11" will not be offered as a link-layer header type, and attempts to use `-y IEEE802_11` even if the "Monitor mode" checkbox, if present, is checked, or if `-I` is specified on the command line. If they are only available in monitor mode, "802.11" will only be offered if the "Monitor mode" checkbox is checked or `-I` is specified on the command line.

Data Packets

Data packets are often supplied to the packet capture mechanism, by default, as "fake" Ethernet packets, synthesized from the 802.11 header; you don't see the real 802.11 link-layer header.

The driver for the adapter will also send copies of transmitted packets to the packet capture mechanism, so that they will be seen by a capture program as well.

Non-data packets

You might have to capture in monitor mode to capture non-data packets. If not, you should capture with 802.11 headers, as no "fake" Ethernet headers can be constructed for non-data frames.

Management Packets

Management packets are used by peer WLAN controllers to maintain a WLAN network, and as such is seldom of importance above OSI layer 2. They are discarded by most drivers, and hence they do not reach the packet capture mechanism. However, if adapter/driver supports this, you may capture such packets in "monitor mode" as discussed below.

Low-level Control Packets

Control packets are used by peer WLAN controllers to synchronize channel access within contending WLAN hardware, as well as to synchronize packet exchange between peers. It is seldom of importance above OSI layer 2. They are discarded by most drivers, and hence they do not reach the packet capture mechanism. However, if adapter/driver supports this, you may capture such packets in "monitor mode" as discussed below.

802.11 Filter (Modes)

802.11 adapters (or their drivers) will filter packets on the receiving side in several ways. This section will give an overview which mechanisms are used and if/how these filters can be disabled.

Channels (Frequencies)

802.11 uses radio frequencies in the range of 2412-2484 MHz; please note that not all frequencies are allowed to be used in all countries. 802.11 splits the available frequencies in 14 network channels, numbered 1-14 (-> 14 "wireless cables"). The frequency range of a channel partially overlaps with the next one, so the channels are therefore not independent. Channels 1, 6 and 11 have no overlap with each other; those three are the unofficial "standard" for wireless channel independence.

Since the frequency range that's unlicensed varies in each country some places may not have 14 channels. For example, Japan has #1-#14, Europe #1-#13 and the FCC in the US allows #1-#11.

The user has to choose which channel to use for the network adapter/access point. Traffic will only be sent to (or received from) that channel.

This filtering can't be disabled. However, special measuring network adapters *might* be available to capture on multiple channels at once.

SSID/ESSID (Network Name)

In normal operation the user sets the SSID (Service Set Identifier) at the access point and the network adapter. If multiple access points use the same SSID it's called an ESSID (Extended SSID). A network adapter will then filter based on this SSID and hand over packets to the host only of the same SSID as it's currently set itself to.

Monitor mode

In monitor mode the SSID filter mentioned above is disabled and *all* packets of *all* SSID's from the currently selected channel are captured.

Even in promiscuous mode, an 802.11 adapter will only supply to the host packets of the SSID the adapter has joined, assuming promiscuous mode works at all; even if it "works", it might only supply to the host the same packets that would be seen in non-promiscuous mode. Although it can receive, at the radio level, packets on other SSID's, it will not forward them to the host.

Therefore, in order to capture all traffic that the adapter can receive, the adapter must be put into "monitor mode", sometimes called "rfmon mode". In this mode, the driver will put the adapter in a mode where it will supply to the host packets from *all* service sets. Depending on the adapter and the driver, this might disassociate the adapter from the SSID, so that the machine will not be able to use that adapter for network traffic, or it might leave the adapter associated, so that it can still be used for network traffic. If it disassociates the adapter from the SSID, and the host doesn't have any other network adapters, it will not be able to:

- resolve addresses to host names using a network protocol such as DNS;
- save packets to a file on a network file server;

etc..

Monitor mode is not supported by WinPcap, and thus not by Wireshark or TShark, on Windows. It is supported, for at least some interfaces, on some versions of Linux, FreeBSD, NetBSD, OpenBSD, DragonFly BSD, and Mac OS X.

You might have to perform operating-system-dependent and adapter-type-dependent operations to enable monitor mode, described below in the "Turning on monitor mode" section.

MAC Addresses

The 802.11 hardware on the network adapter filters all packets received by the destination MAC address (just as in traditional Ethernet), and delivers to the host:

- all Unicast packets that are being sent to one of the addresses for that adapter, i.e. packets sent to that host on that network;
- all Multicast packets that are being sent to a Multicast address for that adapter, or all Multicast packets regardless of the address to which they're being sent (some network adapters can be configured to accept packets for specific Multicast addresses, others deliver all multicast packets to the host for it to filter);
- all Broadcast packets.

Promiscuous mode

In promiscuous mode the MAC address filter mentioned above is disabled and *all* packets of the currently joined 802.11 network (with a specific SSID and channel) are captured, just as in traditional Ethernet. However, on a "protected" network, packets from or to other hosts will not be able to be decrypted by the adapter, and will not be captured, so that promiscuous mode works the same as non-promiscuous mode.

This seems to work on Linux and various BSDs, including Mac OS X. On Windows, putting 802.11 adapters into promiscuous mode is usually crippled, see the Windows section below.

Promiscuous mode can be enabled in the Wireshark Capture Options.

Turning on monitor mode

If you are running Wireshark 1.4 or later on a *BSD, Linux, or Mac OS X system, and it's built with libpcap 1.0 or later, for interfaces that support monitor mode, there will be a "Monitor mode" checkbox in the Capture Options window in Wireshark, and a command line `-I` to dumpcap, TShark, and Wireshark.

In Wireshark, if the "Monitor mode" checkbox is not grayed out, check that check box to capture in monitor mode. If it is grayed out, libpcap does not think the adapter supports monitor mode. If it is not an 802.11 adapter, it cannot support monitor mode; if it is an 802.11 adapter, either the adapter does not support monitor mode, the adapter's driver does not support monitor mode, or there's a bug in libpcap causing it not to think the adapter and driver support monitor mode.

In dumpcap and TShark, and in Wireshark if you're starting a capture from the command line, specify the `-I` command-line option to capture in monitor mode.

FreeBSD 8.0 and later, newer versions of some Linux distributions, and Mac OS X 10.6 (Snow Leopard) and later, come with libpcap 1.x, so versions of Wireshark built on and for those OSes should have the "Monitor mode" checkbox and the `-I` command-line flag. On other OSes, you would have to build and install a newer version of libpcap, and build Wireshark using that version of libpcap.

If that checkbox is not displayed, or if the `-I` command-line option isn't supported, you will have to put the interface into monitor mode yourself, if that's possible. Whether that is possible, and, if it is possible, the way that it's done is dependent on the OS you're using, and may be dependent on the adapter you're using; see the section below for your operating system.

In Linux distributions, for some or all network adapters that support monitor mode, with libpcap 1.0.x and the version of libpcap 1.1.x in some versions of some of those distributions, the `-I` command-line option will cause an error to be reported, and the "Monitor mode" checkbox will be automatically un-checked, either with or without an error dialog. See the "Linux" section below for information on how to manually put the interface into monitor mode in that case.

*BSD

In:

- FreeBSD 5.2 and later;
- NetBSD 2.0 and later;
- OpenBSD 3.7 and later;
- DragonFly BSD 1.2 and later;

you should be able to capture in monitor mode, and see raw 802.11 headers for packets, on at least some 802.11 adapters, if Wireshark is built with and using libpcap 0.8.1 or later. Which adapters support this is dependent on the adapter and the version of the OS; see CaptureSetup/WLAN/FreeBSD for information on FreeBSD, CaptureSetup/WLAN/NetBSD for information on NetBSD, and CaptureSetup/WLAN/OpenBSD for information on OpenBSD.

For most adapters that support monitor mode, to capture in monitor mode, you should:

1. Put the card into monitor mode with the command `ifconfig` *interface* `monitor`. You can also set the channel to monitor by adding the argument `channel` *channel_number* to that command.
2. Request 802.11 headers, as per the above - fake Ethernet headers can be supplied for data frames, but that's impossible for management and control frames.

When a monitor mode capture completes, turn off monitor mode with the command `ifconfig` *interface* `-monitor`, so that the machine can again perform regular network operations with the 802.11 adapter.

DragonFly BSD

From a quick look at the DragonFly BSD CVS source, it appears that the wireless capture support in DragonFly BSD 1.0 and 1.1 was like FreeBSD 4.x, with support only for Cisco/Aironet cards in the old style, and the support in 1.2 is more like FreeBSD 5.x, with the old-style Cisco/Aironet support and with new-style support for some interfaces supported by the wi driver (Prism II and Orinoco, but not Spectrum24). (XXX - is this the case? I need to look into this more; don't create CaptureSetup/WLAN/DragonFly_BSD until I get a chance to check this. -*Guy Harris*)

Linux

Whether you will be able to capture in monitor mode depends on the card and driver you're using. Newer Linux kernels support the mac80211 framework for 802.11 adapter drivers, which most if not all newer drivers, and some older drivers, supports. See the linuxwireless.org list of 802.11 adapter drivers for some information on what 802.11 drivers are available and whether they support monitor mode; drivers listed as supporting cfg80211 and monitor mode should support enough of the mac80211 framework to allow monitor mode to be controlled in a standard fashion. For additional information, see:

- the seattlewireless.net Linux Drivers page;
- this page of Linux 802.11b information for details on 802.11b wireless cards, including information on the chips they use;
- this page of Linux 802.11b+/a/g/n information for details on 802.11b+, 802.11a, 802.11g, and 802.11n wireless cards, including information on the chips they use;
- the aircrack-ng "What is the best wireless card to buy?" page;
- the aircrack-ng tutorial "Is My Wireless Card Compatible?";
- the aircrack-ng driver compatibility page;
- the LinuxWireless Drivers page and Devices pages.

In order to see 802.11 headers, you will have to capture in monitor mode. (XXX - true for all drivers?)

The easiest way to turn manually turn monitor mode on or off for an interface is with the airmon-ng script in aircrack-ng; your distribution may already have a package for aircrack-ng.

Note that the behavior of airmon-ng will differ between drivers that support the new mac80211 framework and drivers that don't. For drivers that support it, a command such as `sudo airmon-ng start wlan0` will produce output such as

```
Interface   Chipset      Driver

 wlan0      Intel 4965 a/b/g/n   iwl4965 - [phy0]
          (monitor mode enabled on mon0)
```

The "monitor mode enabled on mon0" means that you must then capture on the "mon0" interface, *not* on the "wlan0" interface, to capture in monitor mode. To turn monitor mode off, you would use a command such as `sudo airmon-ng stop mon0`, not `sudo airmon-ng stop wlan0`.

For drivers that don't support the mac80211 framework, a command such as `sudo airmon-ng start wlan0` will not report anything about a "mon0" device, and you will capture on the device you specified in the command. To turn monitor mode off, you would use a command such as `sudo airmon-ng stop wlan0`.

If you can't install airmon-ng, you will have to perform a more complicated set of commands, duplicating what airmon-ng would do. For adapters whose drivers support the new mac80211 framework, to capture in monitor mode create a monitor-mode interface for the adapter and capture on that; delete the monitor-mode interface afterwards. To do this in newer Linux distributions with the `iw` command, first run the command `ifconfig -a` to find out what interfaces already exist with names beginning with `mon` followed by a number. Then choose a number greater than all of the numbers for `mon`*N* devices; choose 0 if there are no `mon`*N* devices. Then run the command `iw dev` *interface* `interface add mon`*num* `type monitor`, where *interface* is the ifconfig name for the adapter and *num* is the number you chose. If that succeeds, bring up the interface with the command `ifconfig mon`*num* `up`, and capture on the `mon`*num* interface. When you are finished capturing, delete the monitor mode interface with the command `iw dev mon`*num* `interface del`

On Ubuntu 15.10 (and probably on earlier versions also), I find the easiest way to configure a monitor interface is to plug in your hardware and then run "ifconfig -a". Look at the output and then put entries in /etc/network /interfaces for any interfaces that are related to the hardware you are using and any entries for the monitor interfaces you or wireshark are going to create. You must put two entries in for each interface one for IPV4 and one for IV6 e.g.

```
iface mywificard0 inet manual
iface wywificard0 inet6 manual
```

This stops NetworkManager interfering with then. Here is an example of my interfaces file.

```
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback

# Disable network manager on interface names I want to use for
monitoring
```

```
# related purposes
iface eth1 inet manual
iface mon0 inet manual
iface eth1 inet6 manual
iface mon0 inet6 manual
# Make sure Wireshark generated wifi interfaces are excluded as
well
iface phy0.mon inet manual
iface phy1.mon inet manual
iface phy2.mon inet manual
iface phy3.mon inet manual
iface phy4.mon inet manual
iface phy5.mon inet manual
iface phy6.mon inet manual
iface phy7.mon inet manual
iface phy8.mon inet manual
iface phy0.mon inet6 manual
iface phy1.mon inet6 manual
iface phy2.mon inet6 manual
iface phy3.mon inet6 manual
iface phy4.mon inet6 manual
iface phy5.mon inet6 manual
iface phy6.mon inet6 manual
iface phy7.mon inet6 manual
iface phy8.mon inet6 manual
# Make sure that non monitor wifi interfaces on shared wiphys are
ignored
# as well or I will miss some monitor packets
iface BigTenda inet manual
iface LittleBelkin inet manual
iface LittleTenda inet manual
iface Sinmax inet manual
iface Alfa inet manual
iface WNDA3200 inet manual
iface wlp6s0 inet manual
iface BigTenda inet6 manual
iface LittleBelkin inet6 manual
iface LittleTenda inet6 manual
iface Sinmax inet6 manual
iface Alfa inet6 manual
iface WNDA3200 inet manual
iface wlp6s0 inet6 manual
```

I use entries in /etc/udev/rules.d/70-persistent-net.rules to give my networking hardware friendly names. This is optional. Here is an example.

```
# This file was automatically generated by the /lib/udev
/write_net_rules
# program, run by the persistent-net-generator.rules rules file.
#
# You can modify it, as long as you keep each rule on a single
```

```
# line, and change only the value of the NAME= key.

# PCI device 0x10ec:0x8168 (r8169)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
ATTR{address}=="00:26:18:7f:d1:20", ATTR{dev_id}=="0x0",
ATTR{type}=="1", KERNEL=="eth*", NAME="eth0"

# PCI device 0x10ec:0x8169 (r8169)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
ATTR{address}=="00:02:44:b9:0f:7f", ATTR{dev_id}=="0x0",
ATTR{type}=="1", KERNEL=="eth*", NAME="eth1"

# USB device 0x:0x (rt2800usb)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
ATTR{address}=="c8:3a:35:c4:1c:76", ATTR{dev_id}=="0x0",
ATTR{type}=="1", KERNEL=="wlan*", NAME="BigTenda"

# USB device 0x:0x (rtl8192cu)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
ATTR{address}=="ec:1a:59:0e:51:c3", ATTR{dev_id}=="0x0",
ATTR{type}=="1", KERNEL=="wlan*", NAME="LittleBelkin"

# USB device 0x:0x (rt2800usb)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
ATTR{address}=="c8:3a:35:cc:bd:12", ATTR{dev_id}=="0x0",
ATTR{type}=="1", KERNEL=="wlan*", NAME="LittleTenda"

# USB device 0x:0x (rtl8187)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
ATTR{address}=="00:0f:11:92:06:b2", ATTR{dev_id}=="0x0",
ATTR{type}=="1", KERNEL=="wlan*", NAME="Sinmax"
```

Here is an exmaple script that uses iw to set up a monitor interface. However wireshark will set up a monitor interface for you.

```
DEFAULT_WIPHY=phy0
WIPHY=${1:-$DEFAULT_WIPHY}
DEFAULT_MONIF=mon0
MONIF=${2:-$DEFAULT_MONIF}
DEFAULT_CHANNEL=11
CHANNEL=${3:-$DEFAULT_CHANNEL}

echo "Setting up wifi monitor interface on" $WIPHY
sudo iw phy $WIPHY interface add mon0 type monitor flags none
control otherbss
echo "Bringing up $MONIF"
sudo ifconfig mon0 up promisc
echo "Setting wifi channel to" $CHANNEL
sudo iw dev mon0 set channel 11
```

The default setting of monitor flags on a newly created monitor interface is control|otherbss, Just to ensure that

the flags are in a known state the above script clears all flags then set these two flags itself. At this time (April 2016) there is no way to read monitor flags back out the kernel.

The monitor interface should now be visible in ifconfig and in Wireshark.

Because the new kernel wifi architecture allows multiple virtual interfaces (vif) to share of physical interface (wiphy) it is essential to ensure that any other vif's sharing a wiphy with your monitor vif do not retune the radio to a different channel or initiate a scan. If this happens you will silently miss packets! "NetworkManager" is a major culprit in this respect.

The golden rule is if the radio is not tuned to the channel you will miss stuff!

For adapters whose drivers don't support the new mac80211 framework, see CaptureSetup/WLAN /Linux_non_mac80211.

Note that some adapters might be supported using the NdisWrapper mechanism. Unfortunately, if you use NdisWrapper, you have the same limitations as Windows for 802.11 capture, which usually means "no monitor mode and no 802.11 headers".

Mac OS X

Using Apple's own AirPort Extreme 802.11 wireless cards:

In Mac OS X releases prior to 10.4.0 (Panther and earlier), neither monitor mode, nor seeing 802.11 headers when capturing data, nor capturing non-data frames are supported - although promiscuous mode is supported.

In Mac OS X 10.4.x (Tiger) (at least in later updates), monitor mode is supported; 802.11 headers are provided, and non-data frames are captured, only in monitor mode. To capture in monitor mode on an AirPort Extreme device named en*n*, capture on a device named wlt*n* instead - for example, if your AirPort Extreme device is named en1, capture on wlt1. On PowerPC Macs, you will have to enable that device by changing the !APMonitormode property in the /System/Library/Extensions/AppleAirport2.kext/Contents/Info.plist property list file to have the value "true" (<true/>) and rebooting; on Intel Macs, that device is enabled by default.

In Mac OS X 10.5.x (Leopard), monitor mode is supported; 802.11 headers are provided, and non-data frames are captured, only in monitor mode. To capture in monitor mode on an AirPort Extreme device, select a "Link-layer header type" other than "Ethernet" from the Capture -> Options dialog box in Wireshark or by selecting a link-layer header type other than "EN10MB" with the "-y" flag in TShark or from the command line in Wireshark (the available link-layer types are printed if you use the "-L" flag).

In Mac OS X 10.6.x (Snow Leopard) and later versions, monitor mode is supported; 802.11 headers are provided, and non-data frames are captured, only in monitor mode. With Wireshark 1.4 or later, to capture in monitor mode on an AirPort Extreme device, check the "Monitor mode" checkbox in the "Capture Options" dialog (in Wirehark before 1.8) or in the "Edit Interface Settings" dialog for the interface in Wireshark 1.8 and later. With versions earlier than 1.4, see the description of how to enable monitor mode on 10.5.x.

It's possible to capture in monitor mode on an AirPort Extreme while it's associated, but this necessarily limits the captures to the channel in use. You can use the undocumented "airport" command to disassociate from a network, if necessary, and set the channel. As the command is not in the standard path, you might find it convenient to set up a link, as shown in http://osxdaily.com/2007/01/18/airport-the-little-known-command-line-wireless-utility/:

```
sudo ln -s /System/Library/PrivateFrameworks/Apple80211.framework
/Versions/Current/Resources/airport /usr/sbin/airport
```

Then "airport -I" shows the current channel, among other things, "airport -z" disassociates from any network,

and "network -c<chan>" sets the channel. Enter just "airport" for more details. The command can also scan and sniff.

If you use a Prism II chipset PCMCIA card in a Powerbook, or use another wireless card which is supported appropriately by the wireless sourceforge drivers, you may be able to use software such as KisMAC to dump to file full frames captured in passive mode. Since Wireshark allows review of dumps you could then run them through the Wireshark analyzer. I don't have enough knowledge to tell how/if it is possible to point Wireshark to such a PCMCIA card, or to get it to watch a growing dump file, to allow live analysis but I think it's a plausible project. (if you manage it then remember to write it up here!)

Windows

Starting from Windows Vista: Npcap

Npcap is an update of WinPcap using NDIS 6 Light-Weight Filter (LWF), done by Yang Luo for Nmap project during Google Summer of Code 2013 and 2015. Npcap has added many features compared to the legacy WinPcap.

```
1) NDIS 6 Support
2) "Admin-only Mode" Support
3) "WinPcap Compatible Mode" Support
4) Loopback Packets Capture and Send Support (either as fake
Ethernet or Null/Loopback frames)
5) Raw 802.11 packets Capture Support (in "monitor mode")
```

When installed on Windows Vista or later (including Win7, Win8 and Win10) with option "Support raw 802.11 traffic (and monitor mode) for wireless adapters" selected, all the wireless adapters can be selected in Wireshark so as to capture raw 802.11 traffic. In "monitor mode", raw 802.11 packets (data + management + control) with radiotap header can be see. Otherwise, only 802.11 data packets can be see. You can enter "monitor mode" via Wireshark or WlanHelper.exe tool shipped with Npcap.

When installed on Windows XP or earlier, Npcap will install the legacy WinPcap driver and won't have any raw 802.11 support.

While waiting for an official download page, the current latest installer can be found here: https://github.com/nmap/npcap/releases, the source code can be found here: https://github.com/nmap/npcap

Starting from Wireshark 1.12.8 and 1.99.9, the Windows installer will detect Npcap presence (when installed in WinPcap compatible mode) and will not try to install WinPcap 4.1.3.

WinPcap

Capturing WLAN traffic on Windows depends on WinPcap and on the underlying network adapters and drivers. Unfortunately, WinPcap doesn't support monitor mode and, on Windows, you can see 802.11 headers when capturing, and capture non-data frames, and capture traffic other than traffic to or from your own machine, only in monitor mode.

Promiscuous mode can be set; unfortunately, it's often crippled. In this mode many drivers don't supply packets at all, or don't supply packets sent by the host.

If you experience any problems capturing packets on WLANs, try to switch promiscuous mode off. In this case you will have to capture traffic on the host you're interested in.

If anybody finds an adapter and driver that *do* support promiscuous mode, they should mention it at the bottom

of this page, for the benefit of other users.

See the archived MicroLogix's list of wireless adapters, with indications of how well they work with WinPcap (Wireshark uses WinPcap to capture traffic on Windows), for information about particular adapters.

Useful video to set up packet capture on wireless using Windows bridging: http://www.micro-logix.com/WinPcap/howtonetworkbridge.avi

AirPcap

The AirPcap adapters from Riverbed Technology allow full raw 802.11 captures under Windows, including radiotap information. Note that the AirPcap adaptors are no longer being sold by Riverbed, as announced in their End-of-Availability (EOA) Notice on October 2, 2017.

Intel Centrino adapters

You might have some success capturing non-data frames in promiscuous mode with at least some Centrino interfaces. As these interfaces encapsulate the 802.11 header in a fake Ethernet packet in a non-standard fashion, you will need Wireshark 0.10.6 or later in order to have the non-data packets recognized and properly dissected.

Channel Hopping

When capturing traffic in monitor mode, you can capture on a single, fixed channel, or capture while hopping through multiple channels (channel hopping). Channel hopping will inevitably cause you to lose traffic in your packet capture, since a wireless card in monitor mode can only capture on a single channel at any given time. However, it may be desirable to perform channel hopping initially as part of your analysis to idenitfy all the networks within range of your wireless card, and then select the channel that is most appropriate for analysis.

If you are capturing traffic to troubleshoot a wireless connectivity problem, or to analyze traffic for a single AP or station, it's best to capture on a single, fixed channel.

In order to implement channel hopping for a wireless packet capture, users have a few options. Wireshark does not have a built-in facility to perform channel hopping during a packet capture, but you can have multiple processes controlling a single wireless card simultaneously; one to perform the channel hopping, and a second process to capture the traffic (Wireshark, in this case).

One tool that is particularly effective and flexible for performing channel hopping is Kismet (http://www.kismetwireless.net). The user can control the desired channels, frequencies (e.g. 802.11b, 802.11g, 802.11a) and hopping rate by editing the kismet.conf file. see the Kismet README file at http://www.kismetwireless.net/documentation.shtml#readme.

If you are looking for a simpler channel hopping solution, you can use the following shell script; modify it to suit your needs.

http://802.11ninja.net/code/chanhop.sh

Save this script to a file (e.g. /usr/local/bin/chanhop.sh) and run:

```
# chmod 700 /usr/local/bin/chanhop.sh
```

As root, to make the script executable. Running the script with no arguments displays the following usage instructions:

```
chanhop.sh: Usage:
```

```
./chanhop.sh [-i|--interface] [-b|--band] [-d|--dwelltime]
-i or --interface specifies the interface name to hop on
[mandatory]
-b or --band specifies the bands to use for channel hopping, one
of
        IEEE80211B      Channels 1-11 [default]
        IEEE80211BINTL  Channels 1-13
        IEEE80211BJP    Channels 1-14
        IEEE80211A      Channels 36-161
    Use multiple -b arguments for multiple channels
-d or --dwelltime amount of time to spend on each channel
[default .25 seconds]
e.x. ./chanhop.sh -i ath0 -b IEEE80211BINTL -b IEEE80211A -d .10
Exiting.
```

To use the script, specify the interface name that is monitor mode as the only mandatory arugment:

```
# ./chanhop.sh -i ath0
Starting channel hopping, press CTRL/C to exit.
```

By default, this will cause the specified interface to cycle through the eleven IEEE 802.11b channels with a dwell time of .25 seconds. Optionally, you can specify additional channels with a different dwell time for each channel. For example, if you wish to channel hop between the IEEE 802.11b and IEEE 802.11a channels with a .10 second dwell time, you can specify the following arguments:

```
# ./chanhop.sh -i ath0 -b IEEE80211B -b IEEE80211A -d .10
Starting channel hopping, press CTRL/C to exit.
```

The chanhop.sh script requires the Wireless Tools utility "iwconfig" and standard Linux shell script tools (whoami, sleep).

Constructing similar scripts, using "ifconfig" rather than "iwconfig", for versions of {Free,Net,Open,DragonFly}BSD with the 802.11 framework and adapters whose drivers support the standard 802.11 framework ioctls is left as an exercise for the reader.

Discussion

As this page is becoming very long, split into several subpages? Keeping the platform independant part here and creating platform dependent subpages?

- /BSD (incl. MAC OS X?)
- /Linux
- /Windows

See Also

- Capturing on Ethernet Networks
- Capturing on Token Ring Networks
- Capturing on VLAN Protected Networks
- Capturing on PPP Networks
- Capturing on the Loopback Device

- Capturing on Frame Relay Networks
- Capturing DOCSIS Traffic
- Capturing Bluetooth Traffic
- Capturing on ATM Networks
- Capturing USB Traffic
- Capturing IrDA Traffic
- Capturing on Cisco HDLC Networks
- Capturing SS7 Traffic

CategoryHowTo

CaptureSetup/WLAN (last edited 2019-01-25 19:36:56 by ChristopherMaynard)