Linux Wireless - Linux Kernel Networking (4)- advanced topics

Rami Rosen
ramirose@gmail.com
Haifux, March 2009
www.haifux.org

Haifux
The Haifa Linux Club

Linux Kernel Networking (4)- advanced topics

- Note:
- This lecture is a sequel to the following 3 lectures I gave:

1) Linux Kernel Networking lecture

- http://www.haifux.org/lectures/172/
- **slides**:http://www.haifux.org/lectures/172/netLec.pdf

2) Advanced Linux Kernel Networking - Neighboring Subsystem and IPSec lecture

- http://www.haifux.org/lectures/180/
- **slides**:http://www.haifux.org/lectures/180/netLec2.pdf

Linux Kernel Networking (4)- advanced topics

3) Advanced Linux Kernel Networking -

IPv6 in the Linux Kernel lecture

- http://www.haifux.org/lectures/187/
  - **Slides**: http://www.haifux.org/lectures/187/netLec3.pdf

Contents:

- General.
  - IEEE80211 specs.
  - SoftMAC and FullMAC; mac80211.
- Modes: (802.11 Topologies)
  - Infrastructure mode.
    - Association.
    - Scanning.
    - Hostapd
    - Power save in Infrastructure mode.
  - IBSS (Ad Hoc mode).
  - Mesh mode (80211s).

- 802.11 Physical Modes.
- Appendix: mac80211- implementation details.
- Tips.
- Glossary.
- Links.
- Images
  - Beacon filter – Wireshark sniff.
  - edimax router user manual page (BR-6504N).

- **Note:** we will not deal with security/encryption, regulation, fragmentation in the linux wireless stack and not deal with tools (NetworkManager, kwifimanager,etc). and not with billing (Radius, etc).
- You might find help on these topics in two Haifux lectures:
- Wireless management (WiFi (802.11) in GNU/Linux by Ohad Lutzky):
  - http://www.haifux.org/lectures/138/
- Wireless security (Firewall Piercing, by Alon Altman):
  - http://www.haifux.org/lectures/124/
- Note: We will not delve into hardware features.

General

- Wireless networks market grows constantly
- Two items from recent month newspaper: (ynet.co.il)
  - Over 12,000 wireless room hotels in Israel.
  - Over 50,000 wireless networks in Europe.
- In the late nineties there were discussions in IEEE committees regarding the 802.11 protocol.
- **1999** : The first spec (about 500 pages).
  - (see no 1 in the list of links below).
- **2007**: A second spec (1232 pages) ; and there were some amendments since then.

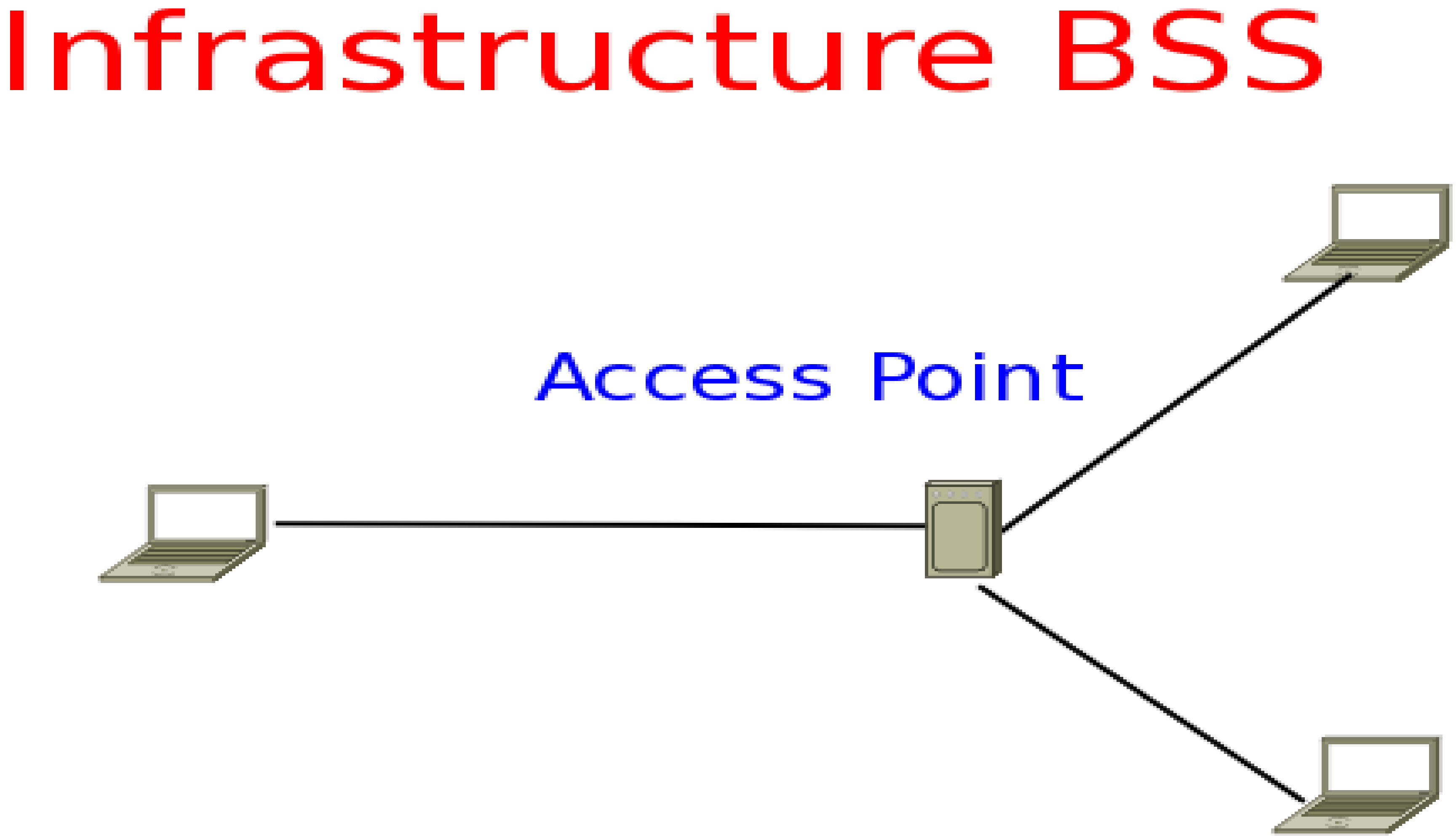
SoftMAC and FullMAC

- In 2000-2001, the market became abound with laptops with wireless nics.
- It was important to produce wireless driver and wireless stack Linux solutions in time.
- The goal was then, as Jeff Garzik (the previous wireless Maintainer) put it: “They just want their hardware to work...".
- **mac80211** - new Linux softmac layer.
  - formerly called d80211 of Devicescape)
- Current mac80211 maintainer: Johannes Berg from sipsolutions.

- Mac80211 merged into Kernel mainstream (upstream) starting 2.6.22, July 2007.
- Drivers were adjusted to use mac80211 afterwards.
- Devicescape is a wireless networking company.
  - http://devicescape.com/pub/home.do
- Location in the kernel tree: net/mac80211.
- A kernel module named mac80211.ko.

- Most wireless drivers were ported to use mac80211.
  - There is a little number of exceptions though.
- Libertas (Marvell) does not work with mac80211.
- libertas_tf (Marvell) uses thin firmware ; so it does use mac80211.
  - libertas_tf supports Access Point and Mesh Point.
  - Both are in OLPC project.
- When starting development of a new driver, most chances are that it will use mac80211 API.

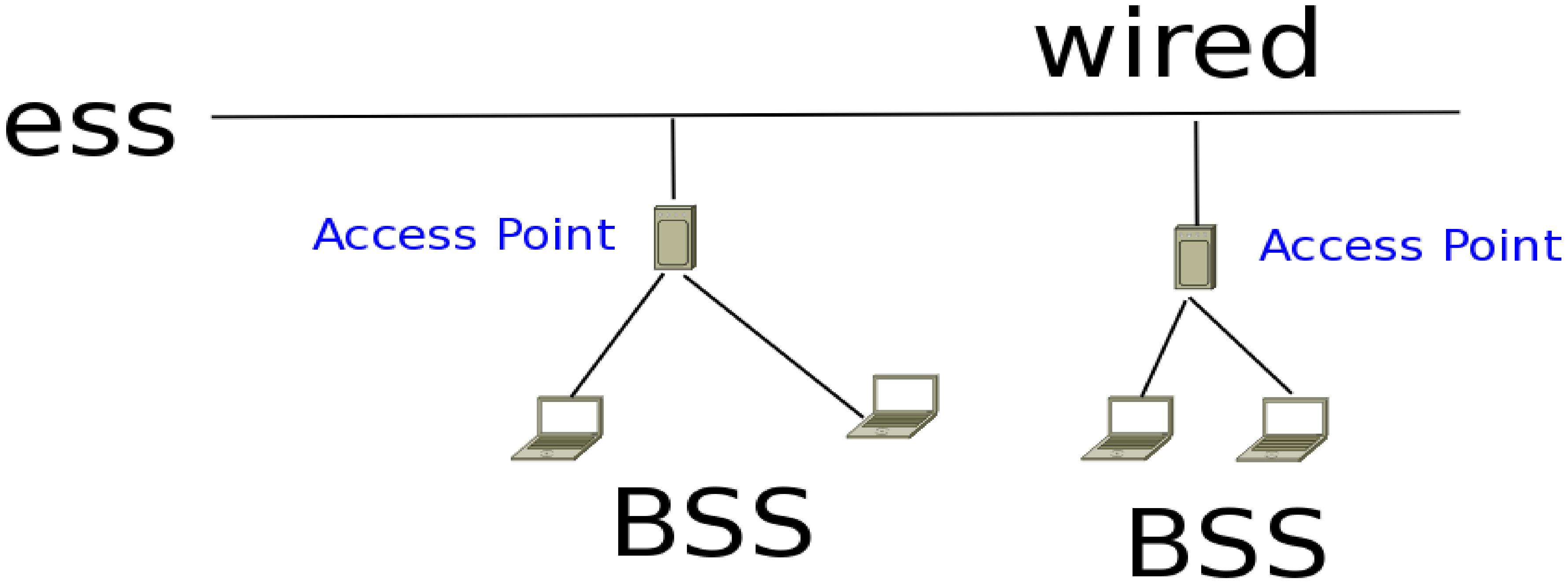
Modes: Infrastructure BSS



Classic ESS (Extended Service Set)

ESS = two or more BSSs.

Infrastructure BSS



wired

ess

Access Point

Access Point

BSS

BSS

- What is an Access Point ?
- Edimax MIMO nMax BR-6504n Router

- Linksys WRT54GL 54Mbps Route

- NOTE: **Infrastructure BSS != IBSS**
  - **IBSS = Independent BSS. (Ad-Hoc mode)**
- **Access Point**: A wireless device acting in master mode with some hw enhancements and a management software (like hostapd).
  - A wireless device in master mode cannot scan

    (as opposed to other modes).
    - Also a wireless device in monitor mode cannot scan.
- Master Mode is one of 7 modes in which a wireless card can be configured.

- All stations must authenticate and associate and with the Access Point prior to communicating.
- Stations sometimes perform scanning prior to authentication and association in order to get details about the Access Point (like mac address, essid, and more).

Scanning

- Scanning can be:
  - **Active** (send broadcast Probe request) scanning.
  - **Passive** (Listening for beacons) scanning.
  - Some drivers support passive scanning. (see the IEEE80211_CHAN_PASSIVE_SCAN flag).
  - Passive scanning is needed in some higher 802.11A frequency bands,as you're not allowed to transmit anything at all until you've heard an AP beacon.
- scanning with "*iwlist wlan0 scan*" is in fact sending an IOCTL (SIOCSIWSCAN).

Scanning-contd.

- It is handled by *ieee80211_ioctl_siwscan().*
- This is part of the Wireless-Extensions mechanism. (aka WE).
- Also other operations like setting the mode to Ad-Hoc or Managed can be done via IOCTLs.
- The Wireless Extensions module; see: net/mac80211/wext.c
- Eventually, the scanning starts by calling *ieee80211_sta_start_scan()* method ,in net/mac80211/mlme.c.
- MLME = MAC Layer Management Entity.

Scanning-contd.

- Active Scanning is performed by sending Probe Requests on all the channels which are supported by the station.
  - One station in each BSS will respond to a Probe Request.
  - That station is the one **which transmitted the last beacon in that BSS.**
    - In infrastructure BSS, this stations is the Access Point.
    - Simply because there are no other stations in BSS which send beacons.
    - In IBSS, the station which sent the last beacon can change during time.

Scanning-contd.

- You can also sometimes scan for a specific BSS:
  - *iwlist wlan1 scan essid homeNet.*
  - Also in this case, a broadcast is sent.
  - (sometimes, this will return homeNet1 also and homeNet2).

Example of scan results

```
iwlist wlan2 scan
```

```
wlan2     Scan completed :
          Cell 01 - Address: 00:16:E3:F0:FB:39
                    ESSID:"SIEMENS-F0FB39"
                    Mode:Master
                    Channel:6
                    Frequency:2.437 GHz (Channel 6)
                    Quality=5/100  Signal level:25/100
                    Encryption key:on
                    IE: Unknown: 000E5349454D454E532D463046423339
                    IE: Unknown: 010882848B962430486C
                    IE: Unknown: 030106
                    IE: Unknown: 2A0100
          IE: Unknown: 32040C121860
          IE: Unknown: DD06001018020000
          Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 18 Mb/s
                    24 Mb/s; 36 Mb/s; 54 Mb/s; 6 Mb/s; 9 Mb/s
                    12 Mb/s; 48 Mb/s
          Extra:tsf=00000063cbf32479
          Extra: Last beacon: 470ms ago
Cell 02 - Address: 00:13:46:73:D4:F1
          ESSID:"D-Link"
          Mode:Master
          Channel:6
          Frequency:2.437 GHz (Channel 6)
```

Authentication

- Open-system authentication (WLAN_AUTH_OPEN) is the only mandatory authentication method required by 802.11.
- The AP does not check the identity of the station.
- Authentication Algorithm Identification = 0.
- Authentication frames are management frames.

Association

- At a given moment, a station may be associated with no more than one AP.
- A Station (“STA”) can select a BSS and authenticate and associate to it.
- (In Ad-Hoc : authentication is not defined).

Association-contd.

- Trying this:
  - *iwconfig wlan0 essid AP1 ap macAddress1*
  - *iwconfig wlan0 essid AP2 ap macAddress2*
- Will cause first associating to AP1, and then disassociating from AP1 and associating to AP2.
- AP will not receive any data frames from a station before it it is associated with the AP.

Association-contd.

- An Access Point which receive an association request will check whether the mobile station parameters match the Access point parameters.
  - These parameters are SSID, Supported Rates and capability information. The Access Point also define a Listen Interval.

- When a station associates to an Access Point, it gets an ASSOCIATION ID (**AID**) in the range 1-2007.

Association-contd.

- Trying unsuccessfully to associate more than 3 times results with this message in the kernel log:
- “apDeviceName: association with AP apMacAddress timed out” and ths state is changed to **IEEE80211_STA_MLME_DISABLED**.
- Also if does not match securiy requirement, will return

  IEEE80211_STA_MLME_DISABLED.

Hostapd

- hostapd is a user space daemon implementing access point functionality (and authentication servers). It supports Linux and FreeBSD.
- *http://hostap.epitest.fi/hostapd/*
- Developed by Jouni Malinen.
- hostapd.conf is the configuration file.
    - Example of a very simple hostapd.conf file:

```
interface=wlan0
driver=nl80211
hw_mode=g
channel=1
ssid=homeNet
```

Hostapd-cont.

- Launching hostapd:
  - *./hostapd hostapd.conf*
  - *(add -dd for getting more verbose debug messages)*
- Certain devices, which support Master Mode, can be operated as Access Points by running the hostapd daemon.
  - Hostapd implements part of the MLME AP code which is not in the kernel
    - and probably will not be in the near future.
    - For example: handling association requests which are received from wireless clients.

Hostapd-cont.

- Hostapd uses the nl80211 API (netlink socket based , as opposed to ioctl based).

Hostapd-cont.

- The hostapd starts the device in monitor mode:

```
drv->monitor_ifidx =
nl80211_create_iface(drv, buf, NL80211_IFTYPE_MONITOR, NULL);
```

The hostapd opens a raw socket with this device:

```
drv->monitor_sock = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL));
```

(hostapd/driver_nl80211.c)

The packets which arrive at this socket are handled by the AP.

- Receiving in monitor mode means that a special header (RADIOTAP) is added to the received packet.
- The hostapd changes management and control packets.
- The packet is sent by the sendmsg() system call:
- *sendmsg(drv->monitor_sock, &msg, flags);*

Hostapd-cont.

- This means sending directly from the raw socket (PF_PACKET) and putting on the transmit queue (by *dev_queue_xmit()*), without going through the 80211 stack and without the driver).
- When the packet is transmitted, an “INJECTED” flags is added. This tells the other side, which will receive the packet, to remove the radiotap header. (IEEE80211_TX_CTL_INJECTED)

Hostapd-cont.

- Hostapd manages:
  - Association/Disassociation requests.
  - Authentication/deauthentication requests.
- The Hostapd keeps an array of stations; When an association request of a new station arrives at the AP, a new station is added to this array.

Hostapd-cont.

- There are three types of IEEE80211 packets:
- The type and subtype of the packet are represented by the **frame control** field in the 802.11 header.
  - **Management** (IEEE80211_FTYPE_MGMT)
  - Each management frame contains information elements (IEs). For example, beacons has the ssid (network name) ,ESS/IBSS bits (10=AP,01=IBSS), and more.
  - (WLAN_CAPABILITY_ESS/WLAN_CAPABILITY_IBSS in ieee80211.h.)
  - There are 47 types of information elements (IEs) in current implementation
  - All in /include/linux/ieee80211.h.

- Association and Authentication are management packets.
  - Beacons are also management frames.
  - IEEE80211_STYPE_BEACON

Hostapd-cont.

- **Control (IEEE80211_FTYPE_CTL)**
- For example, PSPOLL IEEE80211_STYPE_PSPOLL
  - Also ACK, RTS/CTS.
- **Data** (IEEE80211_FTYPE_DATA)
  - See: include/linux/ieee80211.h
- The hostapd daemon sends special management packets called **beacons** (Access Points send usually 10 beacons in a second; this can be configured (see the router manual page at the bottom)).

- The area in which these beacons appear define the basic service area.

From /net/mac80211/rx.c (with remarks)

* IEEE 802.11 address fields:

ToDS	FromDS	Addr1	Addr2	Addr3	Addr4	
0	0	DA	SA	BSSID	n/a	AdHoc
0	1	DA	BSSID	SA	n/a	Infra (From AP)
1	0	BSSID	SA	DA	n/a	To AP (Infra)
1	1	RA	TA	DA	SA	WDS (Bridge)

My laptop as an access point

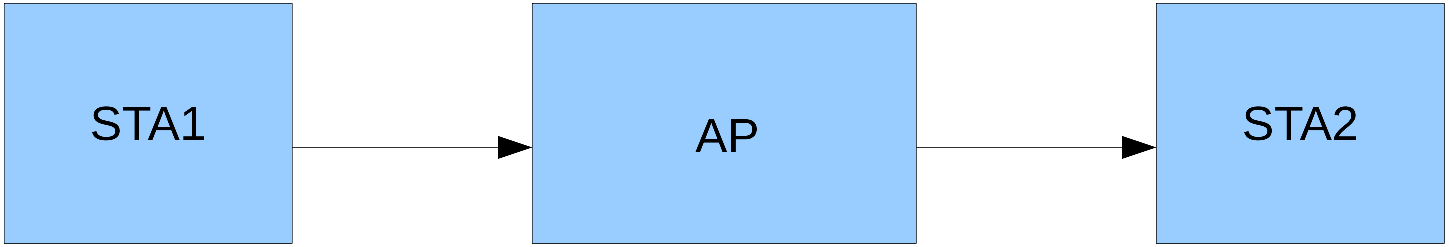
- My laptop as an access point: There is an Israeli Start Up company which develops free access point Windows sw which enables your laptop to be an access point.
- http://www.bzeek.com/static/index.html
- Currently it is for Intel PRO/Wireless 3945.
- In the future: Intel PRO/Wireless 4965.

Power Save in Infrastructure Mode

- Power Save it a hot subject.
- Intel linux Power Save site:
  - http://www.lesswatts.org/
  - PowerTOP util:
    - PowerTOP is a tool that helps you find which software is using the most power.

Power Save in Infrastructure Mode-cont

- Usual case (Infrastructure BSS).



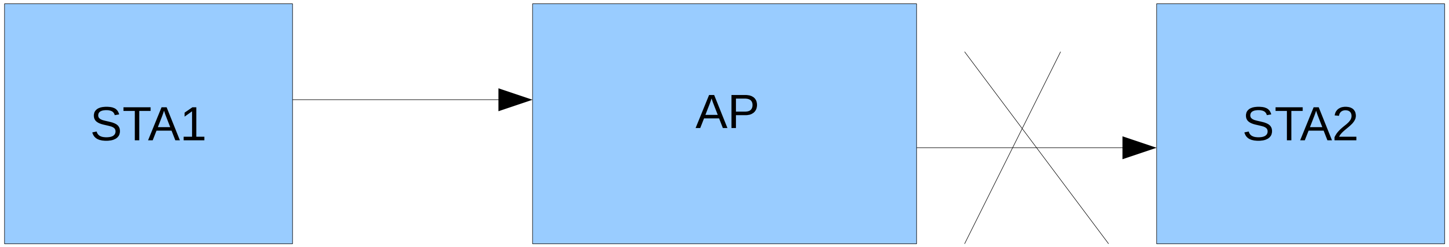
- Mobile devices are usually battery powered most of the time.
- A station may be in one of two different modes:
  - Awake (fully powered)
  - Asleep (also termed “dozed” in the specs)
- Access points never enters power save mode and does not transmit Null packets.
- In power save mode, the station is not able to transmit or receive and consumes very low power.

- Until recently, power management worked only with devices which handled power save in firmware.
- From time to time, a station enters power save mode.
- This is done by:
  - firmware, or
  - by using mac80211 API
    - Dynamic power management patches that were recently sent by Kalle Valo (Nokia).

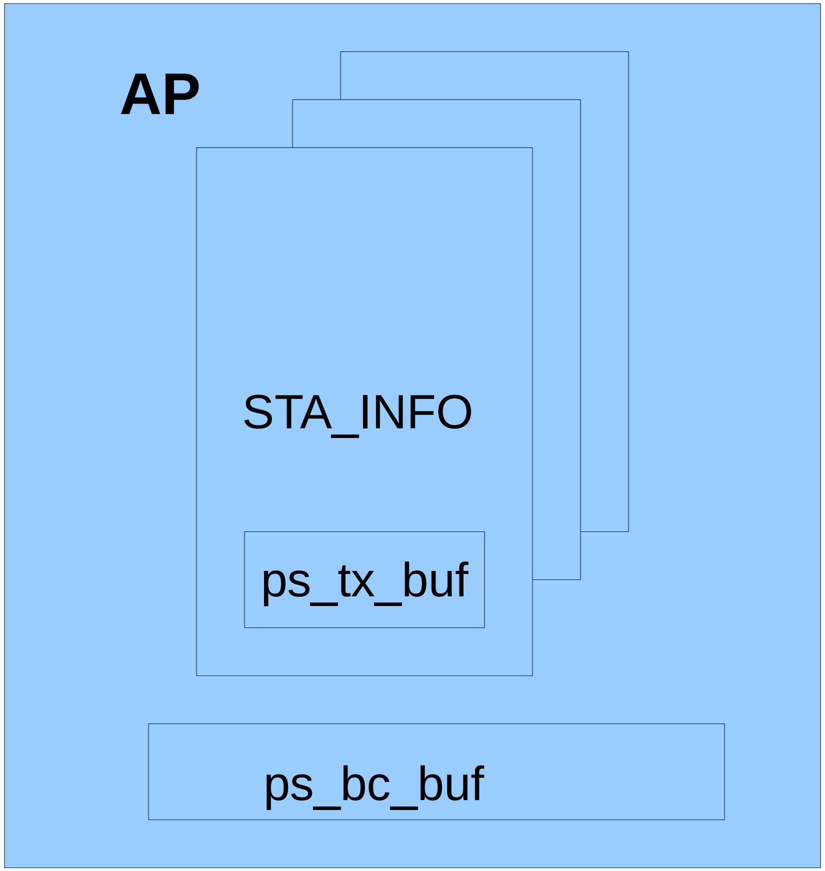
- How do we initiate power save?
- *iwconfig wlan0 power timeout 5*
  - *Sets the timeout to 5 seconds.*
- *Note: this can be done only with the beta version of Wireless Tools (version 30-pre7 (beta)):*
- *http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html*

- In case the firmware has support for power save, drivers can disable this feature by setting **IEEE80211_HW_NO_STACK_DYNAMIC_PS** flag in the driver configuration.
- The Access Point is notified about it by a **null frame** which is sent from the client (which calls *ieee80211_send_nullfunc()*). The **PM** bit is set in this packet (Power Management).

- When STA2 is in power saving mode:
- AP has two buffers: (a doubly linked list of sk_buff structures, sk_buff_head).
  - For unicast frames (ps_tx_buf in sta; one queue for each station).
  - For multicast/broadcast frames. (ps_bc_buf ,one for AP).



- Each AP has an array of its associated stations inside (sta_info objects). Each one has **ps_tx_buf** queue inside, (for unicasts), and **ps_bc_buf** (for multicast/broadcasts)



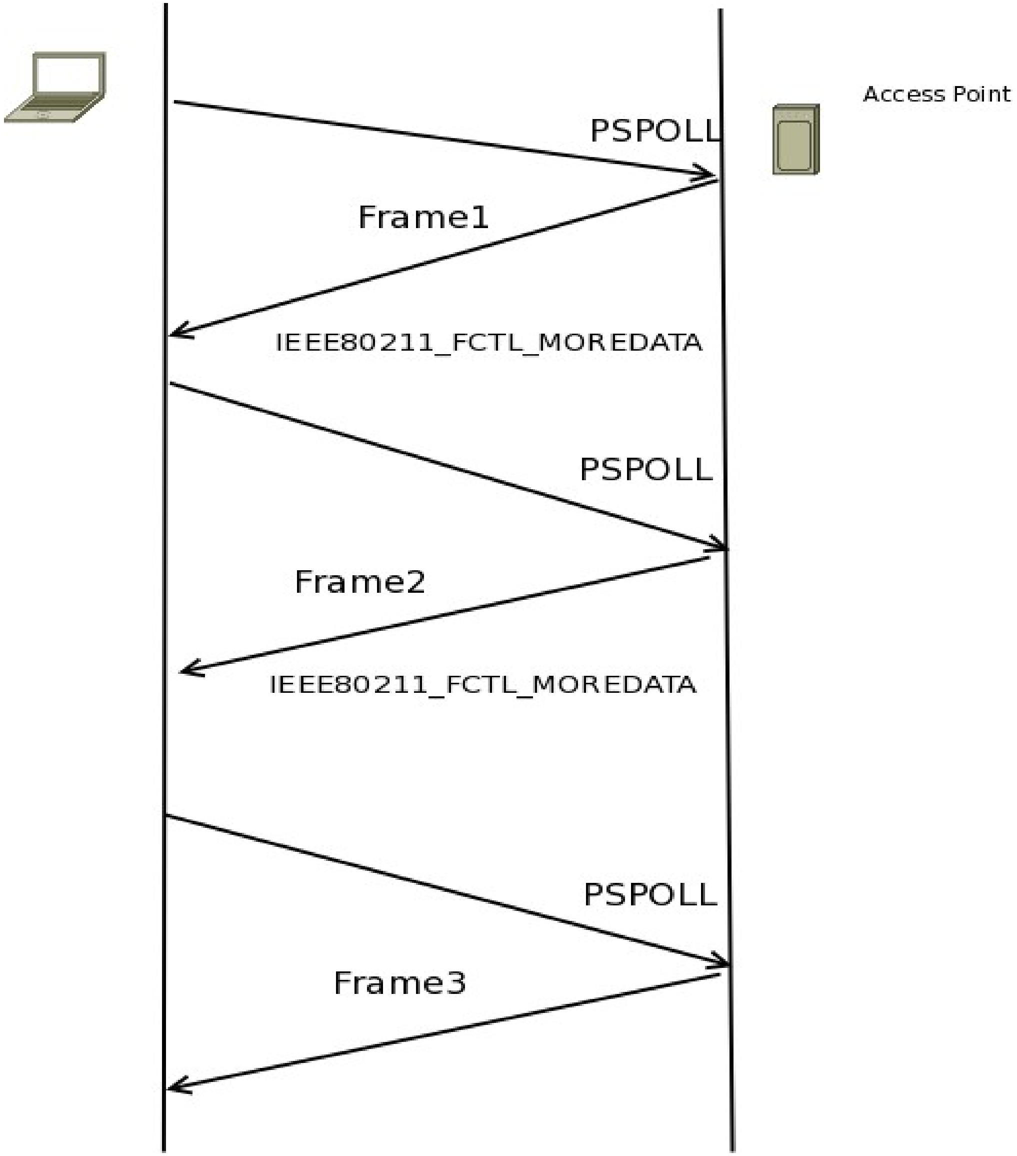
- **The size of ps_tx_buf and of ps_bc_buf is 128 packets**
- #define STA_MAX_TX_BUFFER 128 in net/mac80211/sta_info.h
- #define AP_MAX_BC_BUFFER 128 in net/mac80211/ieee80211_i.h
- Adding to the queue: done by *skb_queue_tail().*
- There is however, a common counter (*total_ps_buffered*) which sums both buffered unicasts and multicasts.
- When a station enters PS mode it turns off its RF. From time to time it turns the RF on, but **only for receiving beacons.**

- When buffering in AP, every packet (unicast and multicast) is saved in the corresponding key.
- The only exception is when strict ordering between unicast and multicast is enforced. This is a service which MAC layer supply. However, it is rarely in use.
- From net/mac80211/tx.c:
  ieee80211_tx_h_multicast_ps_buf() {

```
...
/* no buffering for ordered frames */
if (ieee80211_has_order(hdr->frame_control))
  return TX_CONTINUE;
```

- The AP sends a **TIM** (Traffic Indication Map) with each beacon.
- Beacons are sent periodically from the AP.
- TIM[i]=1 => The AP has buffered traffic for a station with Association ID=i.
  - In fact, a partial virtual bitmap is sent – which is a smaller data structure in most cases.
- The STA sends a PS-POLL packet (Power Saving Poll) to tell the AP that it is awake.
- AP sends the buffered frame.

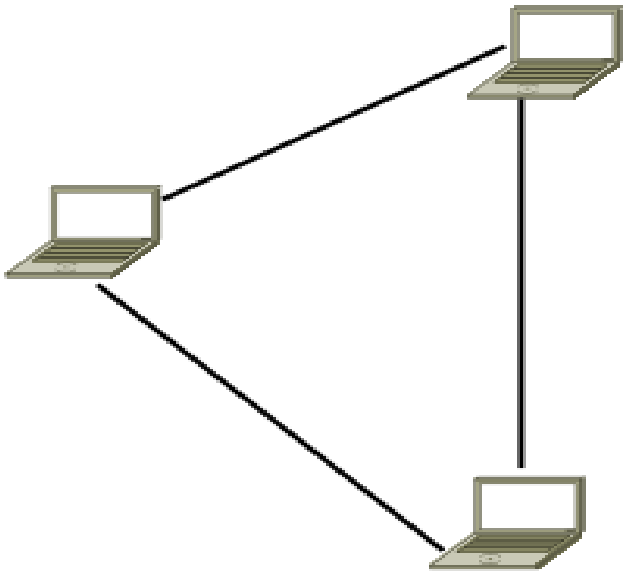
pspoll diagram



IBSS Mode

- IBSS – **without** an access point.

IBSS (Independent BSS)



IBSS Mode - contd

- IBSS network is often formed without pre-planning, for only as long as the LAN is needed.
- This type of operation is often referred to as an Ad Hoc network.
  - Also sometimes called “Peer To Peer” network.

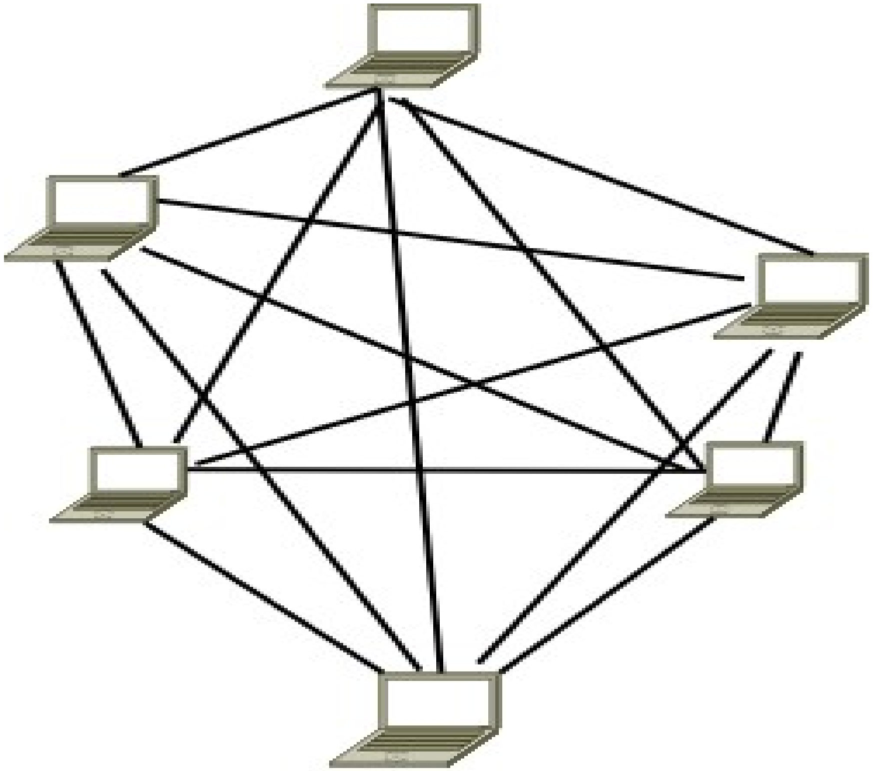
- Creating Ad-Hoc network:
  - iwconfig wlan0 mode ad-hoc
  - (note: if the nic is running, you should run before this: *ifconfig wlan0 down*)
  - iwconfig wlan0 essid myEssid
  - The essid has to be distributed manually (or otherwise) to everyone who wishes to connect to the Ad-Hoc network.
- The BSSID is a random MAC address.
  - (in fact, 46 bits of it are random).

- “iwconfig wlan0 essid myEssid” triggers ibss creation by calling *ieee80211_sta_create_ibss()*
  - *net/mac80211/mlme.c*

- Joining an IBSS:
  - All members of the IBSS participate in beacon generation.
  - The members are synchronized (TSF).
  - The beacon interval within an IBSS is established by the STA that instantiates the IBSS.
  - *ieee80211_sta_create_ibss()* (mlme.c)
  - The bssid of the ibss is a random address (based on mixing get_random_bytes() and MAC address).

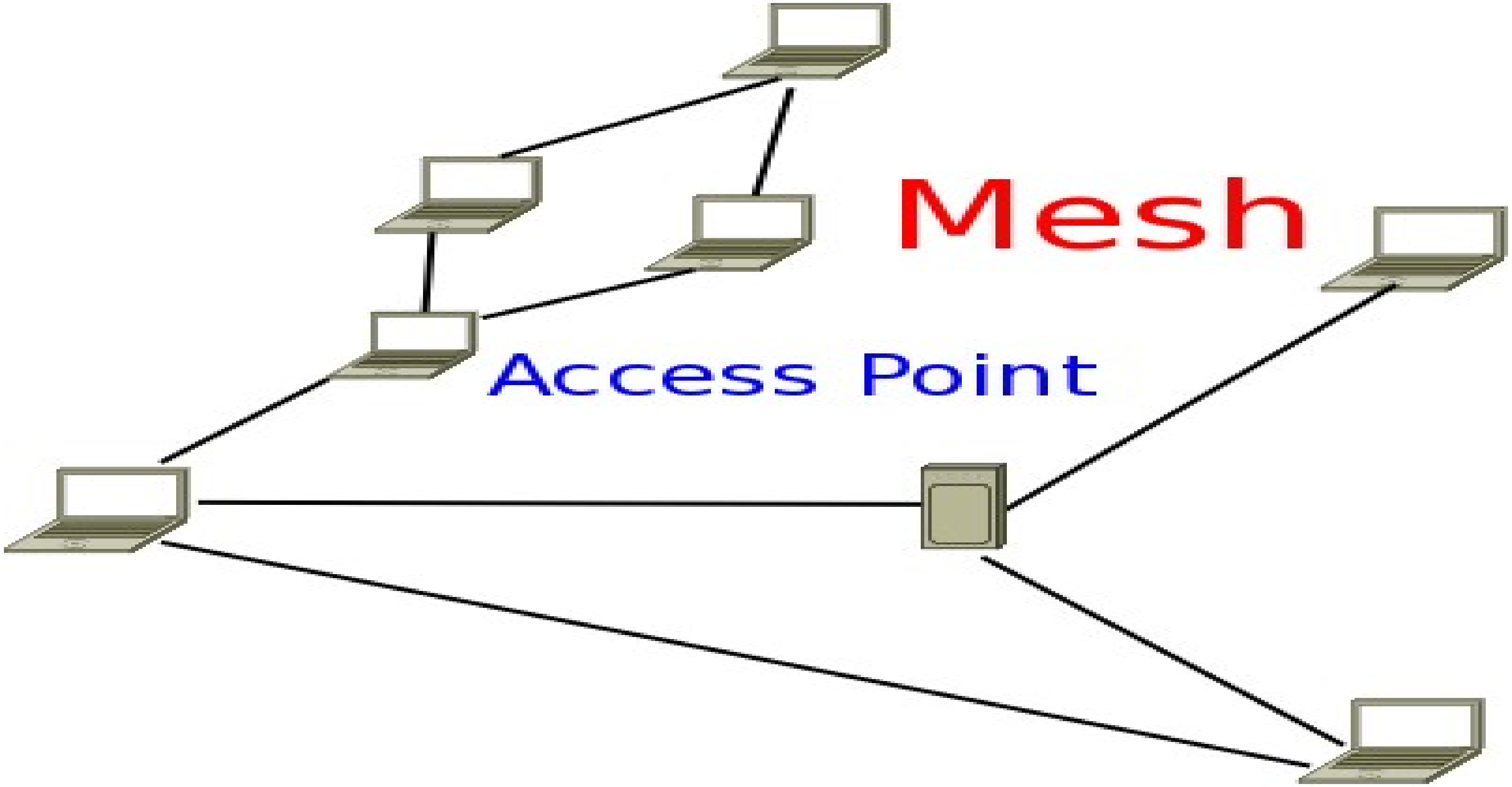
Mesh Mode (80211s)

Full Mesh:In the full mesh topology, each node is connected directly to each of the others.



Mesh Mode (80211s)

Partial Mesh:nodes are connected to only some, not all.



802.11s (Mesh)

- 802.11s started as a Study Group of IEEE 802.11 in September 2003, and became a TG (Task Group) in 2004. (name: TGs)
- In 2006, two proposals, out of 15, (the "SEE-Mesh" and "Wi-Mesh" proposals) were merged into one. This is draft D0.01.
- Wireless Mesh Networks are also called WMN.
- Wireless mesh networks forward data packets over multiple wireless hops. Each mesh node acts as relay point/router for other mesh nodes.

- In 2.6.26, the network stack added support for the draft of wireless mesh networking (802.11s), thanks to the open80211s project (http://www.open80211s.org/).
  - There is still no final spec.
  - There are currently **five** drivers in linux with support to mesh networking (ath5k,b43,libertas_tf,p54, zd1211rw), and **one** is under development (rt2x00).

- Open80211.s
- Goal: To create the first open implementation of 802.11s.
  - Sponsors:
    - OLPC project.

    - Cozybit (http://www.cozybit.com/), the company that developed the mesh software on the OLPC Laptop.
      - Luis Carlos Cobo and Javier Cardona (both from Cozybit) developed the Linux mac80211 mesh code.
    - Nortel

- 80211.s defines a default routing protocol called **HWMP** (Hybrid Wireless Mesh Protocol)
- Based on: Ad Hoc Demand Distance Vector (AODV) routing (C. Perkins); rfc3561.
- The HWMP protocol works with layer 2 (Mac addresses).
- The 80211 header was extended:
  - A ttl field was added to avoid loops.

- The current implementation uses **on demand** path selection.
- The draft also talks about proactive path selection.
  - This is not implemented yet in the Linux Kernel.
  - Uses Root Announcement (RANN) messages and Mesh Portal as a root.

- As with IPV4 static routes, you can force a specific next hop for a mesh station (MESH_PATH_FIXED flag)
  - *(mesh_path_fix_nexthop() in mesh_pathtbl.c)*
- Every station is called an **MP**. (Mesh Point)
- MPP is a Mesh Portal. (For example, when an MP is used to connect to external network, like the Internet).
- Each station holds a routing table (*struct mesh_table*) – helps to decide which route to take.

- In the initial state, when a packet is sent to another station, there is first a lookup in the mesh table; there is no hit, so a **PREQ (Path Request)** is sent as a broadcast.
  - When the **PREQ** is received on all stations except the final destination, it is forwarded.
  - When the **PREQ** is received on the final station, a PREP is sent **(Path Reply).**
  - If there is some failure on the way, a **PERR** is sent.**(Path Error).**
    - Handled by *mesh_path_error_tx()*, mesh_hwmp.c
- The route take into consideration an airtime metric
  - Calculated in *airtime_link_metric_get()* (based on rate and other hw parameters).
- POWER SAVING in the MESH spec is optional.

- **Advantage:**
  - Rapid deployment.
  - Minimal configuration; inexpensive.
  - Easy to deploy in hard-to-wire environments.
- **Disadvantage**:
  - Many broadcasts limit network performance
- You can set a wireless device to work in mesh mode only with the iw command (You cannot perform this with the wireless tools).
- Example: setting a wireless nic to work in mesh mode:
  - *iw dev wlan1 interface add mesh type mp mesh_id 1*
  - *(type = mp => Mesh Point)*

802.11 Physical Modes

- 802.11 (WiFi) is a set of standards for wireless networking, which were defined in 1997 but started to become popular in the market around 2001.
- **802.11a** (1999) at 5 GHz, 54MBit maximum speed; range about 30m.
- **802.11b** (1999) at 2.4GHz, 11Mbit maximum speed, range about 30m.
- **802.11g** (2003) at 2.4GHz, 54Mbit maximum speed, range about 30m.

- **802.11n** (2008) at 2.4GHz/5GHz, 200 Mbit (typical), range about 50m.
- is planned to support up to about 540Mbit/ 600 Mbit.
- Improves the previous 802.11 standards by adding multiple-input multiple-output (MIMO)
  - multiple antennas.
  - High Throughput (HT).
  - Use packet aggregation
    - The ability to send several packets together at one time.

- Still is considered a proposal.
  - Expected to be approved only in December 2009 or later.
- iwlagn and ath9k are the only drivers that support 80211.n in the Linux kernel at the moment.
- Tip: how can I know whether my wireless nic supports 80211.n?
  - Run: *iwconfig*
  - You should see : "IEEE 802.11abgn" or somesuch.

Appendix: mac80211 implementation details

- BSSID = Basic Service Set Identification.
- Each BSS has an BSSID.
- BSSID is an 48 bit number (like MAC address).
  - This avoids getting broadcasts from other networks which may be physically overlapping.
  - In infrastructure BSS, the BSSID is the MAC address of the Access Point which created the BSS.
  - In IBSS, the BSSID is generated from calling a random function (generating 46 random bits; the other 2 are fixed).

Modes of operation

- A wireless interface always operates in one of the following modes:
- **Infrastructure mode**: with an AccessPoint (AP)
  - The access point hold a list of associated stations.
  - also called managed)
- **IBSS** (Independent BSS,Ad-Hoc) mode
  - When using ad-hoc, an access point is not needed.
- **Monitor** mode
- **WDS** (Wireless Distribution System)

Modes of operation - contd.

  - Wireless Distribution System (WDS) - allows access points to talk to other access points.
- **Mesh**

see: include/linux/nl80211.h:

```
enum nl80211_iftype {
  NL80211_IFTYPE_UNSPECIFIED,
  NL80211_IFTYPE_ADHOC,
  NL80211_IFTYPE_STATION,
  NL80211_IFTYPE_AP,
  NL80211_IFTYPE_AP_VLAN,
  NL80211_IFTYPE_WDS,
  NL80211_IFTYPE_MONITOR,
  NL80211_IFTYPE_MESH_POINT,
}
```

cfg80211 and nl80211

- Wireless-Extensions has a new replacement;
- It is cfg80211 and nl80211 (message-based mechanism, using netlink interface).
- iw uses the nl80211 interface.
  - You can compare it the the old ioctl-based net-tools versus the new rtnetlink IPROUTE2 set of tools.
  - You cannot set master mode with iw.
  - You cannot change the channel with iw.

- Wireless git trees:
- Wireless-testing
- Was started on February 14, 2008 by John Linville.
  - primary development target.
  - the bleeding edge Linux wireless developments.
- wireless-next-2.6
- Wireless-2.6
- Daily compat-wireless tar ball in:
- http://www.orbit-lab.org/kernel/compat-wireless-2.6/
- The compat-wireless tar ball includes only part of the kernel
  - (Essentially it includes wireless drivers and wireless stack)

- Fedora kernels are usually up-to-date with wireless-testing git tree.
- There is usually at least one pull request (or more) in a week, to the netdev mailing list (main Linux kernel networking mailing list).
- The Maintainer of the wireless (802.11) in the Linux kernel is John Linville (RedHat), starting from January 2006.

- For helping in delving into the mac80211 code little help.
- Important data structures:
- struct ieee80211_hw – represents hardware information and state (include/net/mac80211.h).
    - Important member: void *priv (pointer to private area).
    - Most drivers define a struct for this private area , like *lbtf_private* (Marvell) or *iwl_priv* (iwlwifi of Intel) or *mac80211_hwsim_data* in mac80211_hwsim.
    - Every driver allocates it by *ieee80211_alloc_hw()*
    - *A pointer to ieee80211_ops (see later) is passed as a parameter to ieee80211_alloc_hw().*
    - Every driver calls *ieee80211_register_hw()* to create wlan0 and wmaster0 and for various initializations.

- You set the machine mode prior to calling *ieee80211_register_hw()* by assigning flags for the interface_modes flags of wiphy member
  - wiphy itself is a member of ieee80211_hw structure.
  - For example,

```
hw->wiphy->interface_modes =
  BIT(NL80211_IFTYPE_STATION) |
  BIT(NL80211_IFTYPE_AP);
```

- This sets the machine to be in Access Point mode.

- struct ieee80211_if_ap – represents an access point. (see ieee80211_i.h)
- Power saving members of ieee80211_if_ap:
  - ps_bc_buf (multicast/broadcast buffer).
  - num_sta_ps (number of stations in PS mode).

- struct ieee80211_ops – The drivers use its members. (include/net/mac80211.h).
- For example, ***config*** (to change a channel) or ***config_interface***

to change bssid.

- Some drivers upload firmware at the start() method, like lbtf_op_start() in libetras_tf driver or zd_op_start() (which calls zd_op_start() to upload firmware zd1211rw
- All methods of this struct get a pointer to struct ieee80211_hw as a first parameter.
  - There are 24 methods in this struct.
  - **Seven** of them are mandatory: tx,start,stop,add_interface,remove_interface,config and configure_filter.
  - (If anyone of them is missing, we end in BUG_ON())

- Receiving a packet is done by calling *ieee80211_rx_irqsafe()* from the low level driver. Eventually, the packet is handled by *__ieee80211_rx()*:

-
```
 __ieee80211_rx()(struct ieee80211_hw *hw,
                          struct sk_buff *skb,
                          struct ieee80211_rx_status *status);
```

- *ieee80211_rx_irqsafe()* can be called from interrupt context.
  - There is only one more mac80211 method which can be called from interrupt context:
  - *ieee80211_tx_status_irqsafe()*

- Data frames
  - Addr1 – destination (receiver MAC address).
  - Addr2 – source (transmitter MAC address).
  - Addr3 - DS info
  - Addr4 – for WDS.
- Management frames
  - Addr1 – destination (receiver MAC address).
  - Addr2 – source (transmitter MAC address).
  - Addr3 - DS info

Firmware

- Firmware:
  - Most wireless drivers load firmware in the probe method (by calling *request_firmware()*)
  - Usually the firmware is not open source.
  - Open FirmWare for WiFi networks site:
  - http://www.ing.unibs.it/openfwwf/
    - Written in assembler.
  - B43 firmware will be replaced by open source firmware.
  - ath5k/athk9k driver doesn't load firmware. (its fw is burnt into an onchip ROM)

Wireless Future trends (WiMax)

- WiMax - IEEE 802.16.
- There are already laptops which are sold with
- WiMax chips (Toshiba, Lenovo).
- WiMax and Linux:
- http://linuxwimax.org/
- Inaky Perez-Gonzalez from Intel
  - (formerly a kernel USB developer)
- Location in the kernel tree: *drivers/net/wimax.*

Wireless Future trends (WiMax) - contd

- Two parts:
- Kernel module driver
- User space management stack, WIMAX Network Service.
- A request to merge linux-wimax GIT tree with the netdev GIT tree was sent in 26.11.08
- http://www.spinics.net/lists/netdev/msg81902.html
- There is also an initiative from Nokia for a WiMax stack for Linux.

Tips

- How can I know if my wireless nic was configured to support power management ?
  - Look in *iwconfig* for “Power Management” entry.
- How do I know if my USB nic has support in Linux?
  - http://www.qbik.ch/usb/devices/
- How do I know which Wireless Extensions does my kernel use?
- Grep for #define WIRELESS_EXT in include/linux/wireless.h in your kernel tree.

- How can I know the channel number from a sniff?
  - Look at the radiotap header in the sniffer output; channel frequency translates to a channel number (1 to 1.)
  - See also Table 15-7—DSSS PHY frequency channel plan , in the 2007 80211
  - Often, the channel number appears in square brackets. Like:
  - channel frequency 2437 [BG 6]
  - BG stands for 802.11B/802.11G, respectively

- Channel 14 for example would show as B, because you're not allowed to transmit 802.11G on it.
- Israel regdomain:
  - http://wireless.kernel.org/en/developers/Regulatory/Database?alpha2=IL
  - IL is in the range 1-13.
  - With US configuration, only channel 1 to 11 are selectable. Not 12,13.
  - Many Aps ares shipped on a US configuration.

- What is the MAC address of my nic?
  - cat /sys/class/ieee80211/phy*/macaddress
  -
  - **Common Filters for wireshark sniffer:**

Management Frames wlan.fc.type eq 0

Control Frames wlan.fc.type eq 1

Data Frames wlan.fc.type eq 2

Association Request wlan.fc.type_subtype eq 0

Association response wlan.fc.type_subtype eq 1

Reassociation Request wlan.fc.type_subtype eq 2

Reassociation Response wlan.fc.type_subtype eq 3

Probe Request wlan.fc.type_subtype eq 4

Probe Response wlan.fc.type_subtype eq 5

Beacon wlan.fc.type_subtype eq 8

Announcement Traffic Indication Map (ATIM) wlan.fc.type_subtype eq 9

Disassociate wlan.fc.type_subtype eq 10

Authentication wlan.fc.type_subtype eq 11

Deauthentication wlan.fc.type_subtype eq 12

Action Frames wlan.fc.type_subtype eq 13

Block Acknowledgement (ACK) Request wlan.fc.type_subtype eq 24

Block ACK wlan.fc.type_subtype eq 25

Power-Save Poll wlan.fc.type_subtype eq 26

Request to Send wlan.fc.type_subtype eq 27

Sniffing a WLAN

- You could sniff with wireshark
- Sometime you can't put the wireless interface to promiscuous mode (or it is not enough). You should set the interface to work in monitor mode (For example: iwconfig wlan0 mode monitior).
- If you want to capture traffic on networks other than the one with which you're associated, you will **have to** capture in **monitor** mode.

Sniffing a WLAN - contd.

- See the following wireshark wiki page, talking about various wireless cards and sniffing in Linux;
- WLAN (IEEE 802.11) capture setup:
  - http://wiki.wireshark.org/CaptureSetup/WLAN#head-
- Using a filter from command line:
  - tshark -R wlan -i wlan0
  - tethereal -R wlan -i wlan0 -w wlan.eth
  - You will see this message in the kernel log:
    - “device wlan0 entered promiscuous mode”

Sniffing a WLAN - contd.

- Sometimes you will have to set a different channel than the default one in order to see beacon frames (try channels 1,6,11)
  - iwconfig wlan1 channel 11
  - Tip: usefull wireshark display filter:
    - For showing only beacons:
    - *wlan.fc.type_subtype eq 8*
  - For tshark command line:
    - *tshark -R "wlan.fc.type_subtype eq 8" -i wlan0*
    - *(this will sniff for beacons).*

Glossary

- AMPDU=Application Message Protocol Data Unit.
- CRDA = Central Regulatory Domain Agent
- CSMA/CA = Carrier Sense Multiple Access with Collision Avoidance
- CSMA/CD Carrier Sense Multiple Access with Collision Detection
- DS = Distribution System
- EAP = The Extensible Authentication Protocol
- ERP = extended rate PHY

- HWMP = Hybrid Wireless Mesh Protocol
- MPDU = MAC Protocol Data Unit
- MIMO = Multiple-Input/Multiple-Output
- PSAP = Power Saving Access Points
- PS = Power Saving.
- RSSI = Receive signal strength indicator.
- TIM = Traffic Indication Map
- WPA = Wi-Fi Protected Access
- WME = Wireless Multimedia Extensions

Links

- 1) IEEE 80211 specs:
  - http://standards.ieee.org/getieee802/802.11.html
- 2) Linux wireless status June - 2008
  - http://www.kernel.org/pub/linux/kernel/people/mcgrof/presentations/linux-wireless-status.pdf
- 3) official Linux Wireless wiki hosted by Johannes Berg.
  - http://wireless.kernel.org/
  - or http://linuxwireless.org/

- 4) A book:
  - 802.11 Wireless Networks: The Definitive Guide
  - by Matthew Gast
  - Publisher: O'Reilly
- 5) Wireless Sniffing with Wireshark - Chapter 6 of Syngress Wireshark and Ethereal Network Protocol Analyzer Toolkit.
- 6) http://www.lesswatts.org/
  - Saving power with Linux (an Intel site)

- 7) A book: Wireless Mesh Networking: Architectures, Protocols And Standards

by Yan Zhang, Jijun Luo, Honglin Hu (Hardcover – 2006)

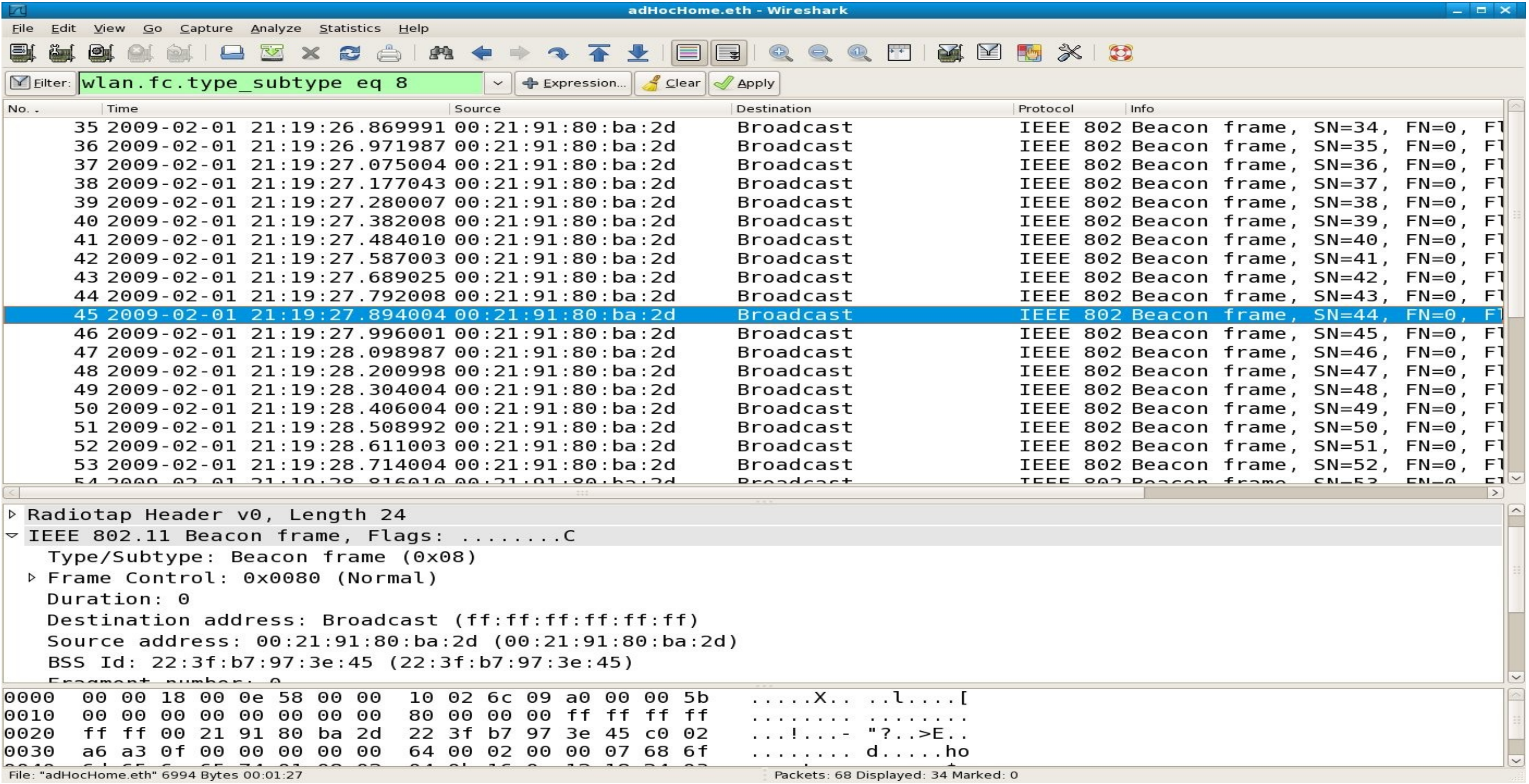
Auerbach Publications

8) http://www.radiotap.org/

Images

- Beacon wireshark filter:
- wlan.fc.type_subtype eq 8
  - shows only beacons.

Beacon filter – sniff



Filter: wlan.fc.type_subtype eq 8

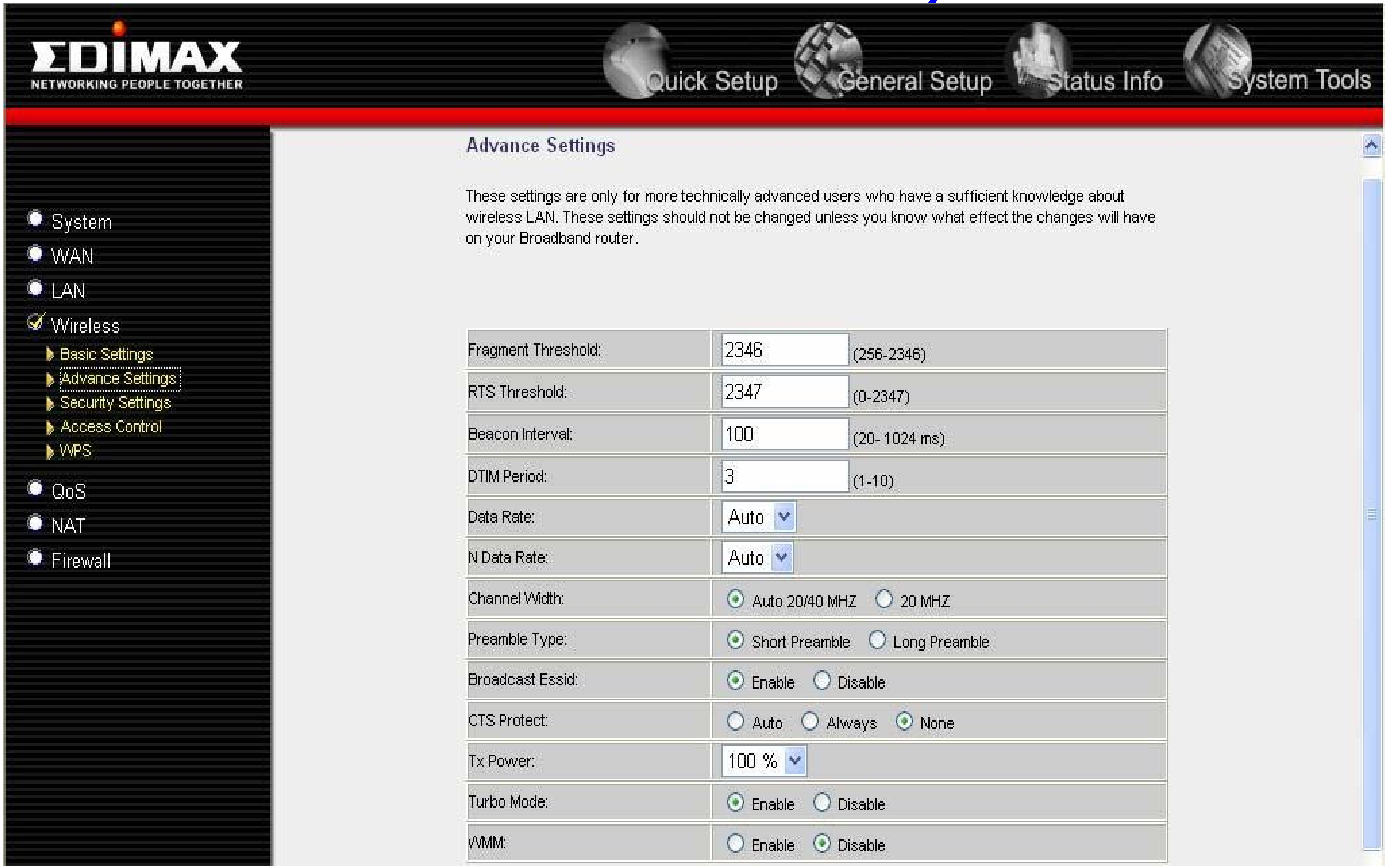
```
No. .  Time                       Source             Destination   Protocol  Info
    35 2009-02-01 21:19:26.869991 00:21:91:80:ba:2d  Broadcast     IEEE 802 Beacon frame, SN=34, FN=0, Fl
    36 2009-02-01 21:19:26.971987 00:21:91:80:ba:2d  Broadcast     IEEE 802 Beacon frame, SN=35, FN=0, Fl
    37 2009-02-01 21:19:27.075004 00:21:91:80:ba:2d  Broadcast     IEEE 802 Beacon frame, SN=36, FN=0, Fl
    38 2009-02-01 21:19:27.177043 00:21:91:80:ba:2d  Broadcast     IEEE 802 Beacon frame, SN=37, FN=0, Fl
    39 2009-02-01 21:19:27.280007 00:21:91:80:ba:2d  Broadcast     IEEE 802 Beacon frame, SN=38, FN=0, Fl
    40 2009-02-01 21:19:27.382008 00:21:91:80:ba:2d  Broadcast     IEEE 802 Beacon frame, SN=39, FN=0, Fl
    41 2009-02-01 21:19:27.484010 00:21:91:80:ba:2d  Broadcast     IEEE 802 Beacon frame, SN=40, FN=0, Fl
    42 2009-02-01 21:19:27.587003 00:21:91:80:ba:2d  Broadcast     IEEE 802 Beacon frame, SN=41, FN=0, Fl
    43 2009-02-01 21:19:27.689025 00:21:91:80:ba:2d  Broadcast     IEEE 802 Beacon frame, SN=42, FN=0, Fl
    44 2009-02-01 21:19:27.792008 00:21:91:80:ba:2d  Broadcast     IEEE 802 Beacon frame, SN=43, FN=0, Fl
    45 2009-02-01 21:19:27.894004 00:21:91:80:ba:2d  Broadcast     IEEE 802 Beacon frame, SN=44, FN=0, Fl
    46 2009-02-01 21:19:27.996001 00:21:91:80:ba:2d  Broadcast     IEEE 802 Beacon frame, SN=45, FN=0, Fl
    47 2009-02-01 21:19:28.098987 00:21:91:80:ba:2d  Broadcast     IEEE 802 Beacon frame, SN=46, FN=0, Fl
    48 2009-02-01 21:19:28.200998 00:21:91:80:ba:2d  Broadcast     IEEE 802 Beacon frame, SN=47, FN=0, Fl
    49 2009-02-01 21:19:28.304004 00:21:91:80:ba:2d  Broadcast     IEEE 802 Beacon frame, SN=48, FN=0, Fl
    50 2009-02-01 21:19:28.406004 00:21:91:80:ba:2d  Broadcast     IEEE 802 Beacon frame, SN=49, FN=0, Fl
    51 2009-02-01 21:19:28.508992 00:21:91:80:ba:2d  Broadcast     IEEE 802 Beacon frame, SN=50, FN=0, Fl
    52 2009-02-01 21:19:28.611003 00:21:91:80:ba:2d  Broadcast     IEEE 802 Beacon frame, SN=51, FN=0, Fl
    53 2009-02-01 21:19:28.714004 00:21:91:80:ba:2d  Broadcast     IEEE 802 Beacon frame, SN=52, FN=0, Fl
    54 2009-02-01 21:19:28.816010 00:21:91:80:ba:2d  Broadcast     IEEE 802 Beacon frame, SN=53, FN=0, Fl
```

```
▷ Radiotap Header v0, Length 24
▽ IEEE 802.11 Beacon frame, Flags: ........C
    Type/Subtype: Beacon frame (0x08)
  ▷ Frame Control: 0x0080 (Normal)
    Duration: 0
    Destination address: Broadcast (ff:ff:ff:ff:ff:ff)
    Source address: 00:21:91:80:ba:2d (00:21:91:80:ba:2d)
    BSS Id: 22:3f:b7:97:3e:45 (22:3f:b7:97:3e:45)
    Fragment number: 0
```

```
0000   00 00 18 00 0e 58 00 00  10 02 6c 09 a0 00 00 5b   .....X.. ..l....[
0010   00 00 00 00 00 00 00 00  80 00 00 00 ff ff ff ff   ........ ........
0020   ff ff 00 21 91 80 ba 2d  22 3f b7 97 3e 45 c0 02   ...!...- "?..>E..
0030   a6 a3 0f 00 00 00 00 00  64 00 02 00 00 07 68 6f   ........ d.....ho
```

File: "adHocHome.eth" 6994 Bytes 00:01:27   Packets: 68 Displayed: 34 Marked: 0

Beacon interval and DTIM period in edimax router (BR-6504N) (From the manual)



Advance Settings

These settings are only for more technically advanced users who have a sufficient knowledge about wireless LAN. These settings should not be changed unless you know what effect the changes will have on your Broadband router.

Fragment Threshold:	2346 (256-2346)
RTS Threshold:	2347 (0-2347)
Beacon Interval:	100 (20- 1024 ms)
DTIM Period:	3 (1-10)
Data Rate:	Auto
N Data Rate:	Auto
Channel Width:	● Auto 20/40 MHZ ○ 20 MHZ
Preamble Type:	● Short Preamble ○ Long Preamble
Broadcast Essid:	● Enable ○ Disable
CTS Protect:	○ Auto ○ Always ● None
Tx Power:	100 %
Turbo Mode:	● Enable ○ Disable
WMM:	○ Enable ● Disable

Thank You !