Wireless−HOWTO

Table of Contents

Table of Contents

Wireless Howto

Roberto Arcomano bertolinux@galactica.it

v1.2, 12 January 2001

Wireless HOWTO for Linux Systems Roberto Arcomano, bertolinux@galactica.it v1.2, 12 January 2001 Wireless is a new technology in networking cards, with high speed rate (up to 11 Mbps). This document explains how to setup Wireless in Linux, compatibility problems, something about geographic requirements and more. Latest release of this document can be found at http://web.tiscalinet.it/bertolinux

---

6. Setup

7. More about Wireless

8. FAQ − Frequently asked questions

9. Appendix A − Netmask 255.255.255.255, proxy arp and bridging

10. Appendix B − Siemens DECT Radio Modem

---

1. Introduction

1.1 Introduction

This document explains something about Wireless networking, how to setup it, problems with it. Unliked wired network, Wireless requires some additional trick to work well. You should know something about antennas, pointing it, roaming info and so on. Feedback are welcome. You can found more interesting help at Jean Tourrilhes Wireless Howto

For any suggestion and feedback write to my email address

1.2 Copyright

---

2. Background Knowledge

2.1 What about Wireless?

Wireless is a new technology that can help you to connect computers at distance. It works with Wireless cards with a TX/RX inside at 2.4 GHz while the software interface is Ethernet−like, with an hardware address different for each card in the world. Typical transmit power is 10−20 mW till 100mW (see standard IEEE 802.11 and FCC/CEPT licenses).

2.2 What's the max distance between radio cards?

The most important thing in Wireless communications is the line of sight clear: you MUST SEE (with eyes or with a binocular) the antenna from the other end or you can have (at most) a little tree between them.

The distance depends on the antenna and (eventually amplifier) used: 2−300 meters with a omnidirectional antenna; 1 km with a directive one; 2−3 km with a omnidirectional amplified (200mW); some km with parabolic antenna. 50−60 km with parabolic or directive antenna amplified (some Watts).

Be aware that it is not always legal to amplifier Wireless cards, cause you could violate FCC/CEPT (and also your country relative) specifics.

2.3 What's the difference between wired and Wireless network?

Wired networks are very simple to setup (at least at low level). Wireless networks are very difficult to setup, to manage, to debug... Typical problem with wired networks like hardware install, software install, debug and so on become very critical with Wireless:

1. You have to choose the right Wireless card: there are many cards from many vendor with many requirement and specs. If you want to create a little LAN/WAN you have to buy IEEE 802.11 compliant Wireless cards with an Access Point.
2. Many cards are PCMCIA, so you have to install pcmcia Linux source first.
3. You have to test it with 2 running systems, first at very short distance, then you can get far.
4. You should test it at any weather (typically rain).
5. Finally be happy for setting up.

If you installed a repeater (Linux box that has many Wireless and wired cards) you may have problem editing its configuration at distance!

2.4 What I need to know to setup a Wireless network?

There are a number of requirement to setup a Wireless network;

```
software requirement:
```

1. Generic network knowledge like IP address, netmask, routing... covered by generic Linux NET3−4−HOWTO; *
2. Specific network knowledge like proxy arp, bridging, proc fs, contained in Proxy−ARP−Subnet,

Bridge Mini−Howto and in Linux Kernel Source (2.2.x or 2.4.x) under Documentation/networking/ ip−sysctl.txt) *

3. Wireless network knowledge like access mode (ADHOC, INFRASTRUCTURE and ACCESS POINT), channel concept, outdoor and indoor defines and so on that you can find in any document concerning Wireless: IEEE standard 802.11, CEPT, etc.

```
non software requirement:
```

1. Minimal experience in antennas, physical mounting, pointing
2. Pc hardware installation with particular attention to not produce interference between different Wireless Cards (if required).

finally a great luck!

* All Howtos needed by this document can be retrieved from http://www.linuxdoc.org

2.5 Why should I setup a Wireless network and what I expect from it?

Why? Because you're not satisfied of wired network!

With Wireless cards you can go across garden, parks, houses, (but you MUST SEE the other end!).

High Level Protocol used in Wireless Cards are the same used in Ethernet cards: TCP/IP over Wireless Ethernet−like but make attention to Windows Sharing Application, cause if you use Linux to forward, you are warned that a ip forwarder doesn't let pass through broadcast messages (see more on NetBIOS protocol): in this case you should use a WINS server to support Network Browsing (see Samba doc).

Wireless let you create a little LAN/WAN with a central point of access (maybe with Internet Access!) and give access to anyone by air!

Imagine a country all cabled by radio machines.

Imagine a network that can connect all country people together, sharing files, audio applications, video applications at high bandwidth (like cable network).

All that can be done (and it's already done in some country) using Wireless cards with Wireless Access Point and Wireless Linux Boxes that can operate as repeater (at IP level such as router or, if you want, at data−link level, with bridge driver, see more at Bridge Http Link or Bridge Ftp Link

2.6 What Wireless cards are covered by this howto?

In this howto I start with a generic configuration (to introduce Wireless networking), then I describe an example for each card I knew directly, with some trick you can use to improve its performance.

```
Wireless Card list:
```

1. Proxim Symphony − http://www.proxim.com
2. Webgear AviatorPRO 2.4 (pcmcia support needed) − http://www.webgear.com

3. Lucent Wavelan I, II, Orinoco − http://www.lucent.com and http://www.orinoco.net
4. Cabletron − http://www.cabletron.com
5. YDI am930_isa − http://www.ydi.com
6. Siemens Radio Modem (Dect) − http://www.siemens.com
7. RadioLan (5 GHZ) − http://www.radiolan.com

For a very more exhaustive list see Jean Tourrilhes Wireless Howto.

Siemens Radio Modem is not a really 802.11 Wireless card, they are modems that you can attach to serial and they act as modem (at 1800 MHz, so DECT technology). Appendix B describe their use.

RadioLan cards work at 5.4GHz in a Windows 9x environment and with a RadioLan Access Point that bridges between Wired and Wireless networks (there are no Linux driver as I know).

2.7 How much do they cost?

Wireless cards listed above are very low expansive: they start from very few hundred of dollars up to some thousand of dollars for Access Point that have 2 Wireless card (Lucent, for example) that can act as a bridge.

3. Technical info about Wireless

Here I report some technical info to understand basic Wireless environment.

3.1 Physical Layer

At first layer ISO/OSI we can have 3 kind of spec:

1. FHSS, Frequency Hopping Spread Spectrum
2. DSSS, Direct Sequence Spread Spectrum
3. Infrared connections, not covered by this howto

3.2 Configurations

2 types of configurations:

1. AdHoc mode (also called Independent mode), where you have independent networks with a BSS (Basic Service Set) each one. Each station has the same BSS.
2. Infrastructure mode, where a number of networks (with a BSS each one) can communicate each other by means of an Access Point (one for each BSS) to create a ESS (Extended Service Set). Also there is a roaming function letting a station "attach" to the nearer Access Point.

Adhoc is the simpler method (and the also the less scalable) and let many hosts communicate each other directly. The restrictive requirement is that all one are to be visible directly to reach a complete coverage of the network. (at least Ideally, because this problem could be solved at IP level! For more see Par 5.4).

```
                  Adhoc mode

                A - - - - - C
```

```
  \       /
|   \   /   |
      /\
|   /    \  |
  /        \
B − − − − − D
```

In a Infrastructure environment you use the Access Point to which ALL other hosts must connect to share the network.

```
Infrastructure mode
                                    ESS

          A − − − | − Access Point − −  Access Point − | − − − D

          B − − − |   BSS1                   BSS2      | − − − E

          C − − − |                                    | − − − F
```

B and C could not see D,E and F, but they can communicate as well cause all one are using the same ESS. Important: A,B and C could also not see each other.

In addition there are terms like indoor and outdoor to distinguish short area coverage from long area coverage.

3.3 Compatibility

Keep on mind that there are a number of Wireless cards in the world, but not every card can communicate with every other one. For talking together the cards have to use to:

1. same configuration mode: all Adhoc or all Infrastructure
2. same physical layer: all DSSS or all FHSS
3. same protocol (for example Proxim has its particular proprietary protocol OpenAir that cannot talk with other FHSS cards).

3.4 Should I use Adhoc or Infrastructure?

Access Point are very useful and killing problem but they are expansive. Ideally, for a more concentrated network you could use Infrastructure mode, while for few hosts you can choose Adhoc: why to spend much money for few hosts?

Anyway be aware that if you spend much money probably all works well while spending less you could have some trouble.

3.5 A Linux Box cannot act as an AccessPoint?

Good asking! Unlucky we have no kind of software (freeware) that can do it, so we have to arrange with Adhoc mode or buying more AccessPoints.

Note: if you are expert in reverse engineering you could download an Access Point firmware, use a processor compatible interpreter and hack the code behind Access Point rewriting one for Linux.

4. Toolbox required

4.1 Hardware requirement

You need a Linux Box (486 or, better, a Pentium 100+ with 16MB+ ram), the Wireless network card, an antenna (see par 2.2). You need the same on the other end (with Win9x or WinNT, if you prefer...) cause you have to simulate a communication!

4.2 Software requirement

You need:

1. recent stable kernel sources (2.2.x)
2. recent stable pcmcia sources (pcmcia−cs) if you bought a pcmcia card
3. Wireless network driver: if you don't have it you can download it from the vendor web site or the card manufacturer web site. If you don't find it you can search at Jean Tourrilhes Wireless Howto.

If you don't find even here you probably have to wait or to convert a Windows driver to a Linux driver!! (good luck!).

After that, you have to recompile your kernel, recompile your pcmcia source (if need by the Wireless card), finally recompile your Wireless driver. That is the generic situation, maybe for some card you have to perform step 3 only or 1 and 3, it depends on specific driver.

5. Overview on Wireless network setup.

5.1 Fundamental steps

Once you have got the needed material and you have compiled all the needed you should do the fundamental step in a Wireless configuration:

1. Low level kernel config Let the Linux Kernel see your Wireless card (at low level, such as ioport, interrupts, dma...): you must see some kind of kernel message that advertise you that Wireless card has been right found and configured.
2. Data−link level setting For each particular Wireless card there is an utility that can set typical Wireless data−link level value. For example in Proxim Symphony the utility is called "rl2cfg" while in pcmcia cards settings are in pcmcia config files. You have to set all your Wireless cards with coherence to make them talk together.
3. Ip setting Now you should be able to use ifconfig and route capabilities to change IP settings.
4. Tricks for better performance and to a avoiding conflicts. Now your Wireless Network is basically working: in addition you have to adjust some particular setting like proxy−arp, icmp echo redirect, bridging, channel change and so on to optimize your network and avoiding strange and bandwidth killing conflicts

N.B.: step 1, 2 and 3 correspond to level 1, 2 and 3 of standard ISO/OSI, while step 4 is an addendum to solve situation generated by netmask 255.255.255.255. In fact 32 bit netmask violates standard ISO/OSI cause the network force to use the same address for broadcast and ip machine and the network address doesn't exist.

Someone could criticize this point of view, but if you use the standard ISO/OSI to configure Wireless network with you'll loss many ones configuring subnets; for each subnet usually you discard 2 IP number (Network and broadcast) and you cannot achieve the flexibility on IP assigning (geographically kind). You can find more on Appendix A about this.

You could notice that step 2 is not present in Wired cards cause there's no particular settings to do there.

5.2 Low Level Kernel Config

Always it's a problem for Pc administration: to let kernel (or in general) see your hardware.

Wireless cards are more complex because many of them usually have a Pcmcia plug, so first of all you have to let your kernel see Pcmcia adapter card, then you can try to install specific hardware driver for your Wireless card.

```
So, in Pcmcia config you have to:
```

1. install linux kernel source, from http://www.kernel.org to /usr/src/linux (see tar and gzip utilities)
2. install linux pcmcia source, from ftp://projects.sourceforge.net/pub/pcmcia−cs to install to /usr/src/pcmcia (see tar and gzip utilities)
3. config and recompile your kernel: read file README in your linux directory (/usr/src/linux)
4. config and recompile your pcmcia source: under /usr/src/pcmcia use configure and make. Be sure your driver is here, else your have to install it following driver instructions (usually a tar zxvf driver.tgz under pcmcia dir is sufficient). After type "make all" to compile. At the end type "make install".
5. After typed install you'll find some useful config files under /etc/pcmcia .

```
In non pcmcia case:
```

1. If your driver is present (99% not) under linux sources, you have to install it in a directory, then to compile it.

Once you know module name you have to load it: in pcmcia config you only need to start pcmcia daemon (/etc/rc.d/init.d/pcmcia start for RedHat), for other "modprobe module_name options". With options you'll give ioport, irq and data−link settings (see Par 5.3) to Wireless driver. Anyway your useful tools to know if hardware has correctly been seen by driver are:

1. "tail /var/log/messages" that explains info about syslog
2. "dmesg" for more info.
3. /proc dir: ioports, devices, irq files and driver specific sub−directories.

5.3 Data−link level setting

```
What is that?
```

Wired networks need only to connect each other and then you'll be able to set TCP/IP parameters.

In opposite Wireless networks need data−link settings, such as:

1. What kind of Wireless network I belong to? (Adhoc or Infrastructure)
2. What channel I have to use?
3. What subnet (BSSID) I belong to, what is my ESS ID?
4. Is my communication protected by such a encryption algorithm? Length key?

As you see there are many settings you have to adjust, the reason come from the architecture of Wireless network: there could be someone, in near distance, that could see your packets, use your services only pointing his antenna on the right direction and setting up right TCP/IP parameters.

In addition there could be many Wireless subnets that could generate interference each other.

So here are:

1. Options at load−time module: "modprobe ray_cs essid='LINUX'" for example or
2. Utilities at run−time driver: "rl2cfg eth1 master".

5.4 Ip setting

This is the third problem you have to face. Here situation become problematic only when your network begin to evolve in a bigger one.

```
Remember Wireless IP Networking doesn't stress you if you don't stress it!
```

A simple configuration

```
                       All the hosts view each other

                               A − − − − − C
                                 \       /
                               |   \   /   |
                                     /\
                               |   /    \  |
                                 /       \
                               B − − − − − D
```

A configuration like this is very simple and don't require nothing special (at Ip level): you only need to assign an IP address for each host and to assign a coherent global netmask.

A more complex configuration

```
                       A doesn't see B directly

                                A <− − − −
                              NO\        |
                            TALK\        C
                                \        |
                                B <− − − −
```

Here A and B can communicate only passing through C.

If the network is in Infrastructure mode and C is the Access Point all is ok. In Adhoc mode you also can design a host to "master" capability (I know the term is not so formal!), a host that creates a BSS and to which any other host can join that BSS.

Full connectivity now is reached at IP level: A and B talk to C using the same C interface, so if you try to ping from A to B you'll receive many ICMP REDIRECT packets from C, cause C is telling A that the destination is already in the network from which come the request.

Solution: type a "echo 0 > /proc/sys/net/ipv4/conf/ethx/send_redirects" (where ethx is the interface on C towards A and C) to null all that.

Another problem: what netmask I assign to A and C? If you assign a netmask to A that include A and C nothing works because A don't use the gateway (C) but make the ARP request with unknown destination MAC address.

You could think to use proxy arp, but without effect cause proxy arp reply to source only when the destination is in a different interface from the source: this is not the case!!

So you have to set a very little netmask (Win9x let it be 255.255.255.254, WinNT at least 255.255.255.248), and you have to assure that hosts A and C don't have the same net address.

```
Examples:
```

1. IP(A) = x.y.z.2/31, IP(B) = x.y.z.3/31. This doesn't work cause A asks for B in its network (ARP request) and C doesn't answer cause, for it, A and B belong to the same interface (so, no proxy arp).
2. IP(A) = x.y.z.1/31, IP(B)= x.y.z.2/31. This works cause A ask to C (send requests to B with C MAC address) for B.

In general with a netmask 255.255.255.254 system works with 2 IP changing only for the final bit.

All that is a TCP/IP forcing but is the only method to obtain an high level of flexibility.

Note: If you use an Access Point (network in Infrastructure mode) you haven't redirect problem, cause all is solved at data−link level (almost every Access Point acts as a bridge...). But Access Point are expansive (about 1000 USD or more) and it is more economic to use a P133 32MB Ram to forward, even with 2 or more cards.

Internet Access

```
A - - - - - C - - Internet
  \       /
|   \   /   |
      /\
|   /    \  |
  /        \
B - - - - - D
```

There is a number of situation:

1. C is the only Public IP address. You only have to set private IP address (192.168.x.y for example) for the Wireless network enabling, on C, forwarding and masquering. A, B and D will have C as default

GW.
2. You have a public netmask visible from Internet and C is your default GW to Internet for the network. You only need to enable forwarding on C, setting up default GW on A, B and D to point to C.
3. You have a public netmask visible from Internet and C is not the default GW to Internet. You have 2 possible solutions: Modify your default GW to let it point to C for your network. Symmetrically you have to let C point to default GW to go to Internet. You could, instead, enable proxy arp feature to C (echo 1 > /proc/sys/net/ipv4/conf/ethx/proxy_arp where ethx is the interface towards the default GW) and set your default GW on C to point to the default GW. Proxy arp is a TCP/IP forcing but works well.

Mixed network: Wired and Wireless

```
               Internet
                      \
                       \        E
                        \     /
                         \  /        Wireless
              A - - - - - C - - F
                \       /   \
              |   \   /   |   \
       Wired        /\          G
              |   /    \  |
                /        \
              B - - - - - D
```

Now C joins 2 networks: on the right Wireless and Wired on the left.

More you have Internet Access, so in total you have 3 network cards in C.

What IP Address I assign to hosts? You have 2 possible solutions:

1. Split up network in 2 subnets: for example 192.168.1.0/24 and 192.168.2.0/24. This solution is quickly but is not scalable if you are using Internet IP addresses cause you have to drop too many IPs.
2. Enable Proxy Arp feature to C for all 2 interfaces. Network parameters (net address and netmask) are the same for Wireless and Wired, but with proxy−arp enabled I can choose which IPs are on Wired and which on Wireless.

```
Now we examine solution 2
```

For example: Consider you have Internet public subnet x.y.z.0/24.

```
Interfaces are:
```

1. ifconfig eth0 x.y.z.C netmask 255.255.255.255 (Wired)
2. ifconfig eth1 x.y.z.C netmask 255.255.255.255 (Wireless)
3. ifconfig eth2 x.y.z.C netmask 255.255.255.255 (to Internet)

```
Static routes on eth2:
```

1. route add IPGW dev eth2

2. route add default gw IPGW

This route stands for addressing all Internet requests to your Default GW: as you notice, first you have to tell Linux where is the router, then let default requesting through it.

```
Static routes on eth0:
```

1. route add x.y.z.A dev eth0
2. route add x.y.z.B dev eth0
3. route add x.y.z.D dev eth0

Hosts A,B and D on the Wired Network

```
Static routes on eth1:
```

1. route add x.y.z.E dev eth1
2. route add x.y.z.F dev eth1
3. route add x.y.z.G dev eth1

Hosts E,F and G on the Wireless Network

Note that flexibility is very high, but you have to manual set each host.

---

6. Setup

Here I report some examples (I hope useful!) on how to configure more diffuse and not expansive Wireless Cards.

6.1 General setup info

Wireless cards have interface similar to any Ethernet cards, so you have to add in /etc/conf.modules:

1. "alias ethx module", where ethx is the interface you want to assign to your wireless card and module is name of kernel module.
2. "options module io=0xAAA irq=I ...", where 0xAAA is the io base address to assign the card, I is the IRQ and so on if there are other parameters.

After this you'll be able to use ifconfig and route commands to configure your card at IP level.

6.2 Proxim Symphony

Network type: FHSS, Adhoc only and with proprietary protocol OpenAir.

Web site: http://www.proxim.com where you need to download documentation and driver for Linux and Win9x.

Drivers come with source code to compile:

1. untar it in a empty directory
2. type make for help.
3. make modules; make modules_install to install the driver rlmod.o and the utility rl2cfg.
4. to run the driver (after modified /etc/conf.modules: see Par.6.1.) you only need to turn up the interface with ifconfig command.

Utility rl2cfg (for help type man rl2cfg once done c step) let you change typical data−link level settings:

1. "rl2cfg dev ethx sta" to set it to station (Slave)
2. "rl2cfg dev ethx msta" to set it to master station (Master)
3. "rl2cfg dev ethx alt" to set it to automatically mode

This is all you have to know to get it properly working.

6.3 Webgear Aviator 2.4 and AviatorPro

Network type: FHSS, Adhoc only for Aviator 2.4 and Infrastructure for AviatorPro.

These cards need more for working, because you have to compile Pcmcia source as they come with Pcmcia plug.

Web site is http://www.webgear.com.

To configure:

1. You have to download source pcmcia and to expand it to /usr/src/pcmcia (see Par 5.2)
2. Download driver form http://www.webgear.com and type "tar zxvf driver.tgz" in /usr/src/pcmcia directory.
3. reconfig pcmcia (see Par 5.2)
4. Following instructions you have to append to file /etc/pcmcia/config.opts entry "source ./ray_cs.opts".
5. Note that in /etc/pcmcia/ray_cs.opts there is a line like this: "module "ray_cs" opts "...". Here you have to modify some settings at data−link level present in "opts" .

```
Arguments:
```

- pc_debug=x , where x is the log level.
- net_type=x, x=0 for AdHoc, x=1 for Infrastructure.
- essid=x, x is the ESSID

Finally, to verify configuration with pc_debug > 0, you will see data−link messages in your console like these: "network started" for a new Wireless network created and "network joined" for a new Wireless network joined to another one.

Also File /proc/ray_cs can help you: flied BSSID report to which Subnet you belong to, if it is null you can receive data from no one cards.

6.4 Lucent Wavelan I, II, Orinoco products and Cabletron

Network type: DSSS, Adhoc and Infrastructure.

Lucent products are very professional ones.

Web site: http://www.lucent.com and http://www.orinoco.net.

Setup is like WebGear−like: step 1,2,3 are similar

Then you have to add to file /etc/pcmcia/config.opts: module "wavelan_cs" opts " ..." for Lucent Wavelan I module and "wavelan2_cs" opts " ...", for Lucent Wavelan II or Orinoco.

Under opts you will specify:

1. port_type=x, where x indicates Adhoc(3) or Infrastructure(1)
2. channel=x, x=channel, option relevant for AdHoc mode only.
3. transmit_rate=x, to fix the speed rate: attention to this setting for compatibility with Cabletron cards.

Note: Ideally, it is possible in a Linux Box to have 2 Lucent Wavelanx cards, one in Adhoc mode and the other in Infrastructure mode. Only one of them could properly works because, when starting pcmcia service, all 2 cards are set with same opts value (so in Adhoc or Infrastructure mode). So we have to create a Linux module (or maybe a user mode program) that can change data_link parameters at run−time such as access mode and channel used in Adhoc mode!

The channel parameter is usually used to avoid interference with near other Wireless TX/RX.

Lucent drivers could also be used for Cabletron cards http://www.cabletron.com

6.5 YDI

Network type: DSSS, Adhoc and Infrastructure.

YDI sells very professional cards with antennas, amplifiers and more.

Web−site http://www.ydi.com

`To install:`

1. untar it in a empty directory.
2. type make for compile.
3. make install to install the driver am930_isa and the wlanctl utility

Once done you can choose if type manual commands using "wlanctl" data−link utility or run the "scripts/wlan" file or "scripts/rc.wlan" file to automatically config your network.

In manual case these are major settings:

1. "wlanctl scan ..." to search for BSSs already present.
2. "wlanctl netlist" show you what's found with "wlanctl scan ...".

3. "wlanctl bsscreate ... ssid" to create a new network with ssid parameter.
4. "wlanctl bssjoin bssid" to join the network identified by bssid.
5. "wlanctl authen" and "wlanctl assoc" for authentication services.

7. More about Wireless

7.1 A Wireless Linux distribution

The wireless distribution FlyingLinux started in October 1999 in the Telecommunication Systems Lab at Teleinformatics KTH with the objective of studying the possibilty of using MobileIPv4 and standard DHCP−based wireless access for student labs.

The result of that work was the FlyingLinux environment available for one hundred students and teachers during the 2G1303 project course that was held from March to May year 2000.

FlyingLinux is the first linux distribution oriented to mobility services. We have taken care of the security issues including Kerberos support and OpenSSH.

FlyingLinux is part of the Open Source movement. We have included software that have been developed at KTH under the GPL licence.

You can find the Wireless Linux distribution at this Web Site.

Mantainer: Alberto Escudero Email,

8. FAQ − Frequently asked questions

Q1: What's the difference between BSSID and ESSID and when I need a ESSID?

A1: BSSID is 48 bit number used to identify the BSS short area, where all hosts talk each other (eventually with an Access Point) ESSID is a variable length string that can let communicate different BSS are to extend it to a Extended Service Set (ESS). There is one Access Point for each BSS and all they talk together only if you belong to the same ESSID. Really you need ESSID if you have a large network with at least 2 Access Points.

Q2: What Access Point I have to buy?

A2: The less expansive you find: what is important is that the Access Point and the cards you are using use the same Physical Layer Specific: all FHSS compatible or all DSSS compatible. Attention to Proxim RangeLan2 cards that cannot talk with other standard FHSS because they use the proprietary protocol OpenAir.

Q3: What do I use the channel setting for?

A3: When you have more network with different BSS (and with different vendors) you could have interference problem: changing channel on Access Points or in Adhoc mode hosts could help you avoiding this kind of problems.

Q4: Why I cannot set channel on Infrastructure hosts?

A4: Because the channel is decided by the Access Point.

9. Appendix A − Netmask 255.255.255.255, proxy arp and bridging

Here we view some Linux advantages in Wireless Internetworking.

Linux let you specify a netmask like 255.255.255.255 for an interface which can help you assign IP addresses in to any interface you want, for example one in eth0, another in eth1 and so on...

This has not particularly side−effects.

In addition you have proxy arp setting under /proc/sys/net/ipv4/conf/ethx/proxy_arp where ethx if your interface.

If you "echo 1 > proxy_arp" you enable proxy_arp for that interface while with "echo 0 > proxy_arp" you disable it.

What's proxy_arp? Quickly proxy arp help you when you want a router answer to an ARP request if the destination address is in another interface of the linux router.

Example:

```
192.168.1.1 ---- 192.168.1.2 Linux router 192.168.2.2 ----192.168.2.1
```

To get this example working you should:

```
Without proxy-arp
```

1. In 192.168.1.1 host to set 192.168.1.2 as gateway
2. In 192.168.2.1 host to set 192.168.2.2 as gateway
3. pinging with success from any edge.

```
With proxy-arp
```

1. In 192.168.1.1 host to set 192.168.1.2 as gateway
2. do not set gateway for 192.168.2.1 host but enable proxy_arp for right interface of the router.
3. pinging with success from any edge

Proxy−arp in 2. case let the linux router answer when you ping from 192.168.2.1 host, saying that it has the 192.168.1.1 host so it can answer for it. After, when the source start sending ICMP packet, Linux router knows that it have to redirect it to the real host 192.168.1.1

In Wireless network proxy arp can help you if you have many Linux boxes that acts as IP Forwarders and you wouldn't set to all hosts a number of static route.

You also can experiment Linux bridging in Wireless network:

1. install a recent stable kernel
2. download good bridge utils at http link or ftp link

Bridging should be more simple to manage.

10. Appendix B − Siemens DECT Radio Modem

Web Site: http://www.siemens.com

What's that? These 2 components are not real PC cards but are more like modems that you use as external device.

```
Host1-serial - RadioModem1 - - - - - RadioModem2 - serial-Host2
```

How can I connect it?

If you see them in an abstract vision you can model them like this:

```
Host1-serial - - NULL MODEM CABLE - - serial-Host2
```

So you'd have a connection between 2 far serials with 2 possible configurations:

1. Linux with Windows, Linux has a daemon that answer to a ppp call while Windows has a Dial up connection under Remote Access.
2. Linux with Linux, where you can run (on all 2 hosts) a ppp connection, with IP addresses inverted.

For 1 you can use this simple script in Linux:

"/usr/sbin/pppd −detach lock idle 300 crtscts connect "/usr/sbin/chat −v TIMEOUT 5 AT OK AT OK AT OK AT OK" IPLINUX:IPWINDOWS /dev/ttySx 115200 disconnect "/usr/sbin/chat −v AT OK" ms−dns IPDNS"

where:

- /dev/ttySx is your serial port,
- IPDNS is the IP address of your dns server,
- IPLINUX is the Linux IP address as IPWINDOWS is the Windows IP address.

The above script is need for letting Windows believe there is a modem on the serial end!

In Windows you need to create a Dial Up connection with crtscts enabled, speed at 115200 and with a stupid number to call (need by Remote Access but absolutely not used).

With 2 Linux boxes you only have to run a very simple script like this at each Linux side:

"/usr/sbin/pppd passive local crtscts IPLINUX1:IPLINUX2 /dev/ttySx 115200 noauth persist"

where you'll invert IPLINUX1 with IPLINUX2 at the other edge.

Note that you can do authentication even Linux−Windows than in Linux−Linux if you want.