Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Afegitons

Redes Wireless con Linux (295763 lectures)

Per **Ricardo Galli Granada**, ***gallir*** (http://mnm.uib.es/gallir/)

Creat el 12/05/2002 00:04 modificat el 12/05/2002 00:04

Las redes Wireless 802.11b, de 11 Mbps están convirtiéndose en una opción muy válida y popular en los últimos meses. Además que los precios han bajado notablemente, dan una enorme comodidad si tenemos varios ordenadores en casa y el soporte que hay en Linux es muy potente y variado. En éste artículo explico las diferentes opciones y como instalarlo y configurarlo en Linux. Aunque sin duda la estrella es, al menos para mí, ***como montar un punto de acceso (access point) wireless en Linux con una tarjeta PCMCIA normal y corriente, y barata*** *(en página 4).*

Introducción

Como veremos en este artículo, con Linux se pueden hacer muchas *virguerías* que son imposibles en sistemas operativos propietarios, como por ejemplo construir un *Access Point* basado en Linux a partir de una tarjeta Conceptronic (Chipset PRISM2) bastante barata y que no está, en principio, preparada para hacer de *Master (o access point)* de una red.

Las forma de trabajo de las tarjetas wireless tienen básicamente tres formas: *ad-hoc* y *managed* y *master*.

- **Ad-hoc**: estas redes se construyen normalmente con ordenadores con las tarjetas "normales" y se configuran de modo que todos los ordenadores de la red trabajan "par a par", todos reciben los paquetes de todos y envían sus propios paquetes a todos los ordenadores de la red. Para esto no se necesita nada especial, sólo definir una red con un nombre (ESSID), preferiblemente encriptar a 128 bits (con WEP) y no tener demasiados ordenadores en la misma red.
- **Managed**: en este caso exite un servidor independiente, *Access Point* (o *Base Station* en terminología comercial de Apple) al cual se conectan todos los ordenadores. El *access point* entonces envía las tramas 802.11 a los destinatarios finales. Normalmente los *access points* soportan *roaming*, es decir los clientes pueden estar en movimiento a ir cambiando de punto de acceso de acuerdo a la potencia de la señal. La diferencia fundamental entre una tarjeta que soporte ser *access point* (o *modo master*) es que hay que hacer bridging de paquetes IP y además manipulan los bits de 802.11 a bajo nivel, normalmente en la propia tarjeta. También los *access points* suelen ofrecer servicios de enrutado IP, servidor DHCP y bridging sobre una Ethernet. Cuando un ordenador o tarjeta está conectado a la red a través de un punto de acceso se dice que está en modo *managed*.
- **Master**: es el modo en que trabaja el access point descrito en el punto anterior. Como veremos al final, es posible también fabricar un *access point* con Linux.

Módulos del kernel

La mayoría de las tarjetas que se venden actualmente son del tipo Orinoco (Lucent), Symbol HR y Prism 2. Todas ellas están soportadas por el driver **orinoco_cs** incluido en el kernel 2.4.x, pero **sólo para trabajar en modo managed o ad-hoc**, no soportan el modo *master*.

Además tienen un pequeño problema, los drivers no están del todo actualizados con la última versión y cuando la tarjeta comparte interrupciones con otros dispositivos genera un montón de líneas de logs por "eventos vacíos". Este problema ya está solucionado en las últimas versiones que se pueden bajar de http://ozlabs.org/people/dgibson/dldwd/[1] y compilarlos. Es bastante sencillo y el README lo explica claramente.

Pero como veremos más adelante, existen otras opciones, los nuevos linux-wlan-ng[2] o el fantástico *hostap driver* para Prism2[3] (i.e. los usados por Conceptronic) que nos permitirán hacer que nuestro Linux se convierta en un *access point* de muy bajo coste si tenemos una tarjeta Prism2 de Intersil.

Configuración de la tarjetas: iwconfig

Una vez cargados los módulos del kernel que sean necesarios, la configuración de las tarjetas se hace de forma similar a las ethernet con el comando ifconfig pero esta vez ayudado con un nuevo comando, el **iwconfig**, que permite cambiar los parámetros específicos de las redes inalámbricas. Por ejemplo:

- Identificador de red (**essid**)
- Frecuencia o canal (**freq/channel**)
- Modo (**mode**: *master|managed|ad-hoc*)
- Velocidad (**rate**)
- Clave de encriptación (**key/enc**)
- Potencia de transmisión (**txpower**)
- etc.

En pocas palabras, con **iwconfig** configuramos los parámetros especiales de wireless y con el **ifconfig** configuramos los parámetros normales de la red IP.

NOTA sobre encriptación

Si especificamos una clave (key) en el iwconfig, las transmisiones estarán encriptadas con el protocolo WEP. En Linux has dos formas de especificarlas:

1. Con passphrase: `iwconfig interface key "s:mi_clave"`. La clave debe ser de 5 caracteres para encriptación de 40 bits y de 13 para 128 bits (en realidad de clave de 104 bits).
2. Con clave en hexadecimal: `iwconfig interface key "mi_clave_en_hexa"`. En este caso se introduce la clave directamente con 5 o 13 caracteres especificados en hexadecimal.

Para mayor seguridad se recomienda que los caracteres que forman la clave sean aleatorios.

Usuarios de Mac OS X

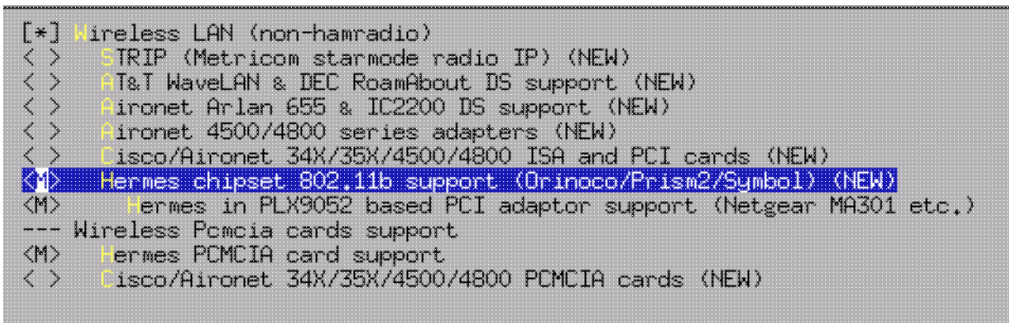
Para introducir la clave en el gestor del Airport del Mac OS X, hay que hacerlo con los caracteres en hexadecimal poniendo un **$** al principio.

Configuración de un cliente Linux (*Managed* o *infraestructure*)

Vayamos primero a lo más simple: **ya tienes un *access point*** y quieres hacer funcionar tu tarjeta PCMCIA en tu Linux, en este caso se suele llamar modo **"infraestructura"** y consiste en poner a la tarjeta en modo ***managed*** para que se conecte en una *especie de punto a punto* con un *access point*.

Quizás hayas comprado la tarjeta con un adaptador PCI (como la Conceptronic PCI), en ambos casos tienes que **habilitar el soporte Cardbus** en el kernel. Para ello tienes que ir a "**General Setup:PCMCIA CardBus Support**" y seleccionar "**Cardbus Support**".

Ahora tienes que seleccionar los drivers orinoco_cs, para ello vas a "**Networking Device Support:Wireless LAN (non-hamradio)**" y seleccionar las opciones y subopciones del "**Hermes Chipset 802.11 support (Orinoco/Prism/Symbol)**".



Ahora deberías compilar el kernel y asegurarte que tienes instalado los paquetes para soporte PCMCIA, deberías tener un directorio /etc/pcmcia/ con varios ficheros alli dentro. Si tienes eso y el kernel compilado e instalado, sólo hace falta configurar la red. Yo trabajo normalmente en Debian, pero también explicaré las diferencias en la configuración para RedHat.

Debian (PCMCIA)

La configuración de la red en Debian puede hacerse dirctamente en los ficheros .opts de /etc/pcmcia.

Mirad por favor la actualización de la configuración de las PCMCIA en Debian[4].

Primero hay que hacer que reconozca automáticamente la tarjeta Conceptronic, para ello hay que editar el fichero **/etc/pcmcia/config.opts** y agregar lo siguiente para que reconozca la tarjeta y cargue el módulo orinoco:

```
card "Conceptronic Wireless"
 version "802.11", "11Mbps Wireless LAN Card"
 bind "orinoco_cs"
```

Ahora hay que editar el fichero **/etc/pcmcia/wireless.opts** y poner las siguientes líneas:

```
*,*,*,*)
 INFO="Nombre..."
 ESSID="Nombre_de_red"
 MODE="Managed"
 RATE="auto"
# La clave se pone si es encriptada
 KEY="s:mi_clave"
 ;;
```

Ahora en el **/etc/pcmcia/networks.opts** hay que poner los datos de la red IP:

```
case "$ADDRESS" in
*,*,*,*)
 INFO="Nombre...."
 # Transceiver selection, for some cards -- see 'man ifport'
 IF_PORT=""
 # Use BOOTP (via /sbin/bootpc, or /sbin/pump)? [y/n]
 BOOTP="n"
 # Use DHCP (via /sbin/dhcpcd, /sbin/dhclient, or /sbin/pump)? [y/n]
 # Solo si tenemos DHCP
 DHCP="y"
 ...
 # Host's IP address, netmask, network address, broadcast address
 # Solo si no tenemos DHCP
 #IPADDR="192.168.0.130"
 #NETMASK="255.255.255.128"
 #NETWORK="192.168.0.128"
 #BROADCAST="192.168.0.255"
 # Gateway address for static routing
 #GATEWAY="192.168.0.1"
```

```
 # Things to add to /etc/resolv.conf for this interface
 ...
```

Debian (PCI y Airport)

En caso que la tarjeta no sea una PCMCIA, es decir tengamos un Apple con tarjeta Airport o una con adaptador PCI, la configuración en Debian se hace directamente en el **/etc/network/interfaces**. Por ejemplo:

```
auto eth1
# ejemplo con dhcp
iface eth1 inet dhcp
 #address 192.168.0.140
 #netmask 255.255.255.0
 #network 192.168.0.0
 #gateway 192.168.0.1
 wireless_essid Nombre_de_red
 wireless_mode Managed
 wireless_key s:mi_clave
 wireless_rate auto
 wireless_nick sofi
```

Red Hat

Red Hat usa una filosofía diferente (al menos en la versión 7.2), en vez de configurar la red dentro de los ficheros en /etc/pcmcia, lo hace en los mismos ficheros donde se configuran las interfaces de red, en **/etc/sysconfig/network-scripts/ifcfg-ethX**. Por lo tanto los parámetros de red hay que configurarlos en dichos ficheros, por ejemplo:

```
DEVICE=eth1
MODE=managed
ESSID="Nombre_de_red"
RATE=auto
TXPOWER=auto
KEY="s:mi_clave" # Solo si va encriptado
BOOTPROTO=static
IPADDR=192.168.0.3
BROADCAST=192.168.0.255
NETMASK=255.255.255.0
NETWORK=192.168.0.0
ONBOOT=yes
```

Configuración de Redes *ad-hoc*

La configuración de un red *ad-hoc* permite montar una red entre pocos ordenadores **sin necesidad de contar con un *access point***. El modo de funcionamiento es *peer-to-peer*, todos los ordenadores reciben los paquetes enviados por algunos de ellos. Por eso una red de tipo ad-hoc funciona bien cuando hay pocos ordenadores.

La **única diferencia en cuanto a configuración** es que hay que poner modo ***ad-hoc*** en vez de **managed**. Por ejemplo:

```
iface eth1 inet dhcp
 #address 192.168.0.140
 #netmask 255.255.255.0
 #network 192.168.0.0
 #gateway 192.168.0.1
 wireless_essid Nombre_de_red
 wireless_mode ad-hoc
 wireless_key s:mi_clave
 wireless_rate auto
 wireless_nick sofi
```

Conectividad entre la wireless y la ethernet

¿Quieres que haya interconectividad entre los ordenadores con wireless y los conectados a la ethernet? Ah!!!!! **Ésta es una de las importantes tareas realizadas por los *access-points***.

Si piensas hacer algo de lo que se explica a continuación, es necesario conocer los fundamentos de funcionamiento de redes IP y Ethernet (para eso están los cursos de redes en la Universidad, FP o academias de informática :-). Si no tienes idea no intentes entender lo que viene a continuación, llama a un amigo que sí sepa de redes y/o ponte a estudiar de cursos o tutoriales disponibles en Internet.

En caso que montes una red *ad-hoc*, necesitas que uno de los ordenadores con una interfaz inalámbrica y otra Ethernet se encargue de ese trabajo de enrutado. Por suerte, el protocolo IP está basado totalmente en software y el Linux es capaz de hacer dichas tareas de enrutado IP. En este caso te recomiendo que aprendas enrutamiento IP y configures dos redes IP distintas, una será la ethernet y la otra la wireless. Uno de los Linux deberá hacer el enrutado IP.

Pero además **hay otras opciones que trabajan a niveles más bajos y que permiten tener una sola red IP a partir de dos redes *físicas* distintas**:

- **Proxy arp**[5]: Para que dos ordenadores en una LAN Ethernet con TCP/IP puedan comunicarse necesitan conocer la dirección MAC (ethernet) del otro ordenador. De estar tarea se encarga el protocolo ARP, que mediante paquetes broadcast averiguan y crean un tabla que relaciona direcciones IP con direcciones MAC (probar el comando *arp -a*). El Linux soporta proxy arp por defecto, sólo hay que configurarlo mediante las variables disponibles en /proc[6].
- **Bridging**: El *bridging* es una opción más avanzada que el proxy arp, y se necesita opciones especiales del kernel además del paquete **bridge-utils**. Si deseas esta opción, mira como se configura el bridge en la sección que viene a continuación (*Configuración de Linux como un Access Point*).

Configuración de Linux como un Access Point

NOTA: Montar un access point, con todos sus requerimientos, no es algo trivial (para los Windozeros: no penséis que en Windows sería más fácil, **ahora mismo es imposible hacer en Windows lo que explico aquí**, no os queda más opción que comprar un *Access Point*), sino que hay que saber de redes y configurar y compilar el kernel como así también sentirse cómodos usando las utilidades y configurando PCMCIA. Si no es así, quizás no entiendas lo que se explica a continuación, es mejor que empieces con algo más sencillo, como lo explicado en los pasos anteriores, o que te ayude "en vivo" algún amigo que ya lo haya hecho.

Mi filosofía es que cuando no se conoce algo y hay que aprender, hay que hacerlo paso a paso, desde lo más fácil a lo más complejo. Aunque en mis primeras pruebas con *wireless* en Linux pasaron por todas las etapas explicadas anteriormente, mi objetivo desde el principio era poder tener un *access point* corriendo en Linux.

Debo decir que he tenido mucha suerte, ya que las tarjetas que compé, la Conceptronic PCI C11iDT[7]:

y la Conceptronic PCMCIA Airbridge 11CC[8]:

ambas (como creo que todas las Conceptronic, por cierto, el web que tienen es realmente lamentable para el tipo de empresa que son), usan el chipset Prism2.5 de Intersil. Digo que he tenido mucha suerte porque he encontrado que un grupo de *pirados* liderados por Jouni Malinen están desarrollando un excelente módulo de Linux para las Prism2[3] (**Host AP**) que **permite trabajar en modo *Master***, aunque el fabricante diga que no se puede :-).

> Me acaban de avisar que nuestro conocido Jordi Murgó es parte del *equipo de pirados*, y que su colega David Farré está haciendo un applet Gnome[9] para mostrar estadisticas del enlace.

```
[gallir@ponti gallir]$ /sbin/iwconfig wlan0
wlan0 IEEE 802.11-DS ESSID:"Antoli" Nickname:"ponti"
 Mode:Master Frequency:2.422GHz Access Point: 00:50:C2:01:96:14
 Bit Rate:2Mb/s Tx-Power=20 dBm Sensitivity=1/3
```

```
 Retry min limit:8 RTS thr:off Fragment thr:off
 Power Management:off
 Link Quality:0 Signal level:0 Noise level:0
 Rx invalid nwid:0 Rx invalid crypt:0 Rx invalid frag:0
 Tx excessive retries:3167 Invalid misc:2038 Missed beacon:0
```

Así que al final he logrado lo que quería, la tarjeta PCI que había instalado en mi Linux de mesa funcionó en modo Master y pude interconectar ordenadores con Linux en PC e iBook, Mac OS X y del *innombrable*, con encriptación WEP de 128 bits.

Como la tarjeta PCI es sólo una PCMCIA con un adaptador, la saqué del adaptador y terminé instalándola en un portátil muy antiguo (P133, 32 MB RAM) para que me hiciese de servidor de acceso para toda la red de casa, con ***bridging***, servidor DHCP y control de direcciones MAC:

```
[gallir@ponti gallir]$ ps ax
 PID TTY      STAT TIME COMMAND
 1    ?       S    0:03 init [3]
 2    ?       SW   0:00 [keventd]
 3    ?       SW   0:00 [kapmd]
 4    ?       RWN  0:00 [ksoftirqd_CPU0]
 5    ?       SW   0:06 [kswapd]
 6    ?       SW   0:00 [bdflush]
 7    ?       SW   0:00 [kupdated]
 8    ?       SW   0:01 [kjournald]
 228  ?       S    0:02 /sbin/cardmgr
 694  ?       S    0:05 syslogd -m 0
 699  ?       S    0:03 klogd -2
 831  ?       S    0:00 /usr/sbin/apmd -p 10 -w 5 -W -P...
 851  ?       SL   0:00 ntpd -U ntp
 933  ?       S    0:33 /usr/sbin/sshd
 984  ?       S    0:00 gpm -t ps/2 -m /dev/mouse
 1002 ?       S    0:00 crond
 1038 ?       S    0:00 /usr/sbin/atd
 1045 tty1    S    0:00 login -- root
 1046 tty2    S    0:00 login -- root
 1047 tty3    S    0:00 /sbin/mingetty tty3
 5636 ?       S    0:04 /usr/sbin/dhcpd
 5835 tty1    S    0:00 -bash
 6734 tty2    S    0:00 -bash
 7691 ?       S    0:03 /usr/sbin/sshd
 7692 pts/0   S    0:00 -bash
 7905 pts/0   R    0:00 ps ax
[gallir@ponti gallir]$ /sbin/lsmod
Module        Size Used by
hostap_cs    82496 1
3c574_cs      8400 1
ds            6384 2 [hostap_cs 3c574_cs]
yenta_socket  8368 2
pcmcia_core  38016 0 [hostap_cs 3c574_cs ds yenta_socket]
```

También he probado el mismo portátil con la otras tarjetas Conceptronic que compré (la PCMCIA de la segunda foto) y funciona perfectamente, aunque su alcance es levemente inferior debido a la antena pequeña incluida, pero aún así la redfunciona a 11 mbps en todo el piso a pesar de que el portátil está en una mala ubicación.

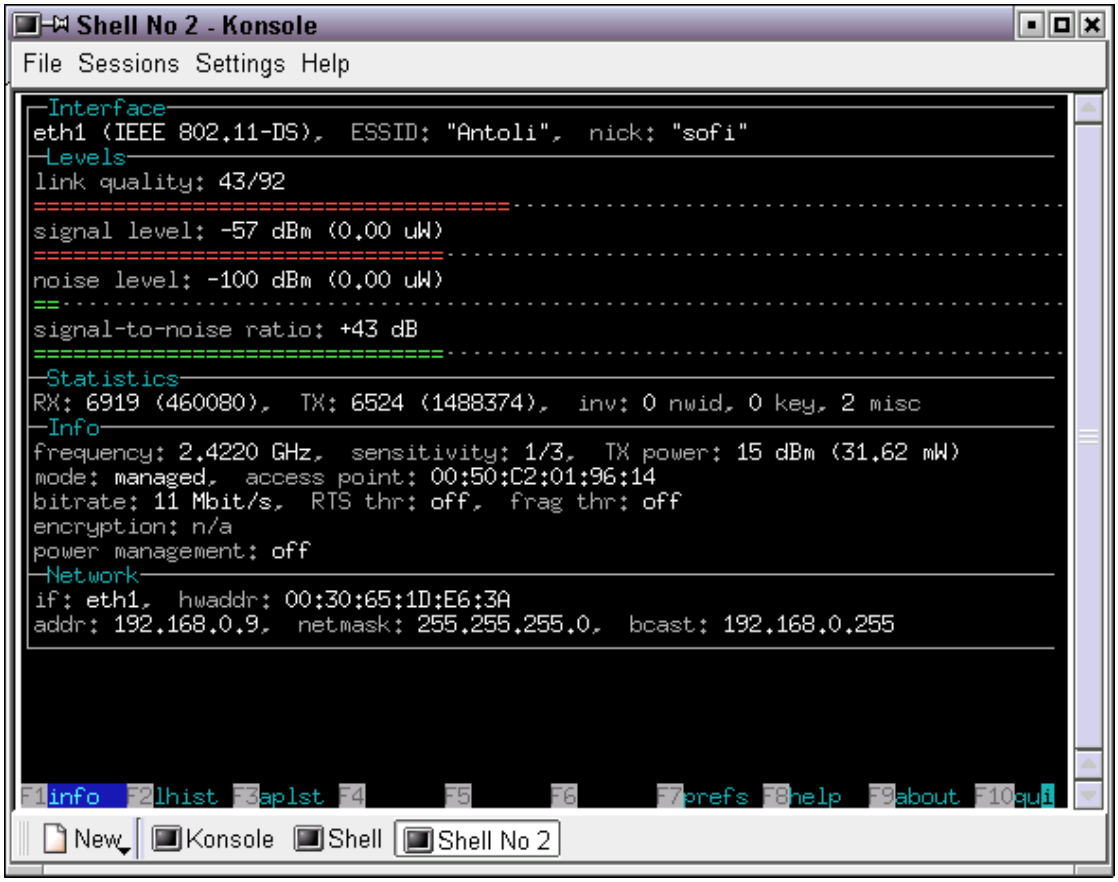
Instalación del módulo Host AP

En la página web del Host AP y en el README del software (un tgz) está bien explicado como instalar el módulo **hostap** en sus distintas variantes, especialmente PCMCIA y PCI. Básicamente hace falta los fuentes del kernel que se está usando (se pone el *path* en el Makefile), luego sólo hay que compilarlo y hacer un make install para que instale los módulos en el directorio correspondiente (/lib/modules/2.4.18/pcmcia/ en mi caso) y el fichero de configuración de la PCMCIA (**hostap_cs.conf**) en el directorio /etc/pcmcia .

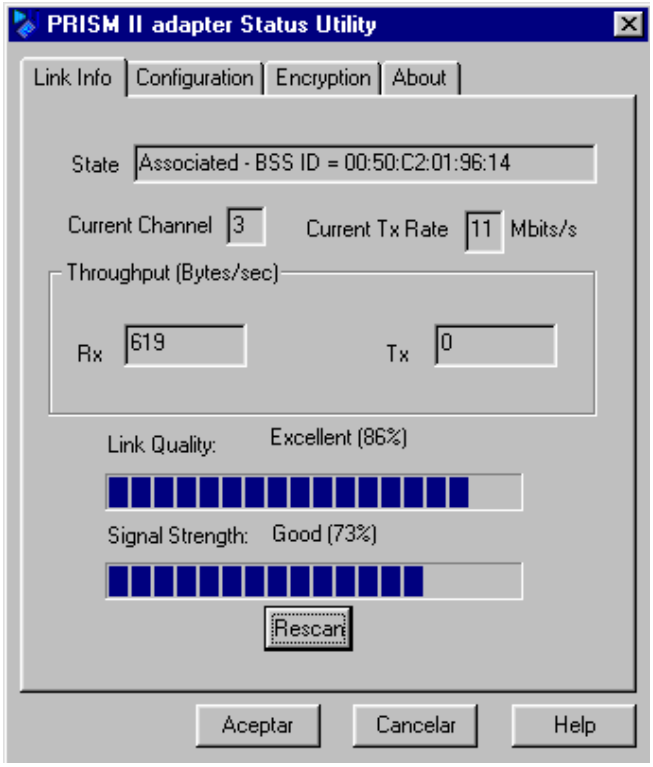
Una vez instalado, si usamos la opción PCMCIA (seguramente sí, aunque tengamos el adaptador PCI) debemos indicar a los módulos del PCMCIA que cuando detecte esa tarjeta cargue el módulo hostap_cs.

Para ello editamos el fichero **/etc/pcmcia/config.opts** y ponemos la siguientes líneas:

```
card "Conceptronic Wireless"
 version "802.11", "11Mbps Wireless LAN Card"
 bind "hostap_cs"
```



wavemon **sobre un Linux en el Apple iBook con Airport conectado el Access Point**



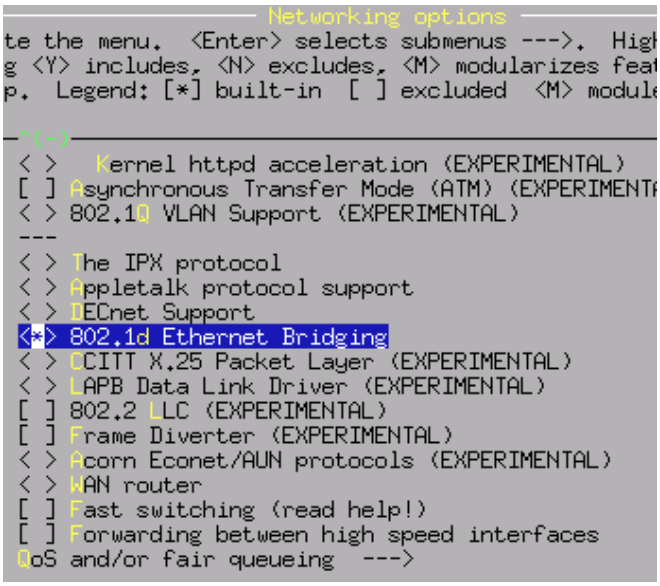
Este es el Windows de mi hija, en su habitación, también con una tarjeta Conceptronic conectado a la red wireless a traves del Linux AP

Con eso ya tenemos casi todo lo referente a drivers, ahora hay que configurar la red.

Bridging

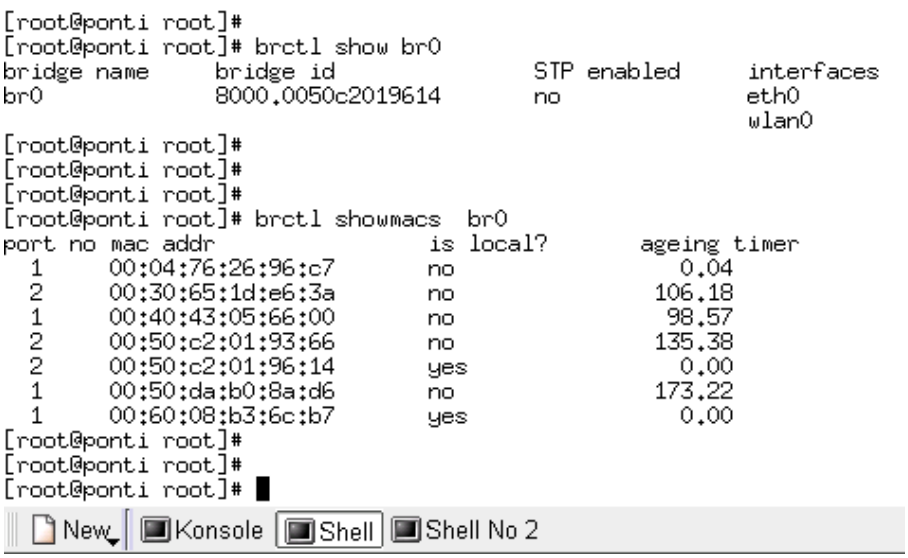
Como comenté anteriormente, no basta con poner la tarjeta en modo *Master* para tener un *access point*, sino que hay además interconectar la red Ethernet y la inalámbrica. Yo opté por hacer bridging entre ambas redes, así trato toda la red de mi casa como si fuese una sola, sin preocuparme en dar direcciones IP de distintas redes si estoy conectado por Ethernet o Wireless.

Para hacer bridging en Linux hay que habilitar dicha opción en el kernel:



Luego hay que instalar el paquete **bridge-utils** que está disponible en Debian (en Red Hat hay que buscarlo en rpmfind.net[10], cuando yo lo hice sólo estaba disponible en la versión RawHide pero me funcionó perfectamente sobre una RedHat 7.2).

El programa principal de dicho paquete, **brctl**, permite crear y configurar interfaces *virtuales* que incluyen las interfaces sobre las que se aplicará el *bridging*.



En la imágen de arriba se puede observar primero la configuración del *bridge* definido (**br0**), que incluye a las interfaces **eth0** y **wlan0** (en el módulo *hostap* las interfaces *wireless* se denominan *wlanX*). Veréis que he deshabilitado el *spanning tree protocol* ya que en mi red estoy seguro que no hay bucles (es muy simple), pero si tú no lo estás, déjalo habilitado.

A continuación se puede observar (con el argumento *showmacs*) los ordenadores conectados a cada red, la 1 es la ethernet y la 2 es la *wlan0*. Las marcadas como *local* son las interfaces del servidor, las demás son las remotas.

ATENCIÓN: las interfaces que forman parte de un bridge **no deben tener ninguna dirección asignada**, en los ficheros de configuración de la tarjeta hay que ponerle una IP **0.0.0.0** en vez de una IP real, ya que deben trabajar en modo promiscuo.

Definición del br0 en Debian

En Debian es muy fácil configurarlo, ya que el **/etc/network/interfaces** es muy flexible y potente, en mi caso sólo tuve que poner

```
auto br0
iface br0 inet static
 address 192.168.0.10
 netmask 255.255.255.0
 network 192.168.0.0
 gateway 192.168.0.1
 bridge_ports eth0 wlan0
 bridge_stp off
 bridge_maxwait 5
```

Os recomiendo leer además lo de RedHat para entender mejor el procedimiento completo.

Definición del br0 en Red Hat

En Red Hat es un poco más complicado por dos razones, no hay un ficheros interfaces como en Debian y además las tarjetas de red PCMCIA se configuran siempre desde el **/etc/sysconfig/network-scripts/ifcfg-xxx**. Pero yo le hice un truco bastante *sucio*, tengo un **ifcfg-eth0**, **ifcfg-wlan0** y **ifcfg-zbr0** (el z lo pongo para que se llame después de llamar a los de las interfaces). El contenido de cada fichero es:

/etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE=eth0
IPADDR=0.0.0.0
BOOTPROTO=static
ONBOOT=yes
```

/etc/sysconfig/network-scripts/ifcfg-wlan0

```
DEVICE=wlan0
MODE=Master
ESSID=Antoli
RATE=auto
TXPOWER=auto
KEY="s:mi_clave"
BOOTPROTO=static
ONBOOT=yes
/usr/bin/prism2_param wlan0 host_decrypt 1
```

NOTA: Con la ejecución de "/usr/bin/prism2_param wlan0 host_decrypt 1" estoy obligando a hacer el desencriptado WEP de las tramas en el propio driver, por software, en vez de hacerlo en la tarjeta. La conveniencia o no de usarlo depende de la tarjeta que uséis y la versión del driver. O sea, probadlo, quizás vuestra tarjeta no funcione con encriptación de 128/104 bits sin esa opción. El comando **prism2_param** es un script incluido en el paquete hostap y sirve para simplificar las llamadas al **iwpriv** para cambiar parámetros de la tarjeta.

/etc/sysconfig/network-scripts/ifcfg-zbr0

```
/usr/sbin/brctl delif br0 eth0
/usr/sbin/brctl delif br0 wlan0
/usr/sbin/brctl delbr br0
/usr/sbin/brctl addbr br0
/usr/sbin/brctl addif br0 eth0
/usr/sbin/brctl addif br0 wlan0
/usr/sbin/brctl stp br0 off
DEVICE=br0
BOOTPROTO=static
BROADCAST=192.168.0.255
IPADDR=192.168.0.3
NETMASK=255.255.255.0
NETWORK=192.168.0.0
ONBOOT=yes
```

Creo que os comenté al menos lo más importante, si me he dejado algo en el tintero, disculpas, son muchas cosas en un sólo artículo... pero quería que tengáis al menos una guía para demostrar **la potencia de Linux en el área de *wireless***.

Nota final: El artículo está orientado para los que quieran montarse una red casera, sin embargo, como me ha recalcado Jordi Murgó (uno de los desarrolladores del módulo hostap), el módulo permite muchas más cosas a nivel profesional y montar redes muy sofisticadas con *roaming* transparente:

- Sistema de distribución.
- Delegación de autentificación.
- Notificación a procesos de usuario, lo que permite cosas más complejas como cambios de *firewalls*, enrutados, notificación de eventos, etc.

Ricardo Galli

Comentarios en la siguiente página...

Comentarios...

Lista de enlaces de este artículo:

1. http://ozlabs.org/people/dgibson/dldwd/
2. http://www.linux-wlan.org/
3. http://hostap.epitest.fi/
4. http://bulma.net/body.phtml?nIdNoticia=1891
5. http://www.sjdjweis.com/linux/proxyarp/
6. http://www.tldp.org/HOWTO/Adv-Routing-HOWTO-16.html#ss16.2
7. http://www.conceptronic.net/products.asp?p=C11iDT&Aktie=5&mt=C11iDT&
8. http://www.conceptronic.net/products.asp?p=CON11C&Aktie=5&mt=C ON11C&am
9. http://www.polakilandia.org/gwlan/
10. http://rpmfind.net/

E-mail del autor: gallir _ARROBA_ uib.es
Podrás encontrar este artículo e información adicional en: http://bulma.net/body.phtml?nIdNoticia=1309