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

Résaux Sans Fils avec Linux (32444 lectures)

Per **Daniel Rodriguez**, ***DaniRC*** (http://www.ibiza-beach.com/)

Creado el 15/05/2002 00:52 modificado el 15/05/2002 00:52

Les cartes de réseaux Sans Fils 802.11b, à 11 Mbps deviennent une option très intéressante et populaire ces derniers mois. En plus d'avoir un prix qui baisse continuellement, elles sont très pratiques si on possède plusieurs ordinateurs à la maison et le support existant pour Linux est puissant et très varié. Avec cet article j'explique les différentes options et la manière d'installer et configurer le Sans Fil sur Linux. Mais sans aucun doute, au moins pour moi, la star de l'article est de savoir comment créer un point d'accès (access point) radio avec Linux, avec une carte PCMCIA tout à fait normale et très économique (voire la page 4)

Merci a Phillipe de la corréction!! (DaniRC)

Introduction

On verra au fur et à mesure de l'article, comment faire, avec Linux, plusieurs *virgueries* qui sont impossibles sur des Systèmes d'Exploitation propiétaires, comme par exemple construire un *Acces Point* sur Linux avec une carte Conceptronic (Chipset PRISM2) assez bon marché et qui n'est pas en principe préparée pour servir de *Maître (ou access point)* d'un réseau.

Les cartes sans fil ont basiquement trois façons de travailler: *ad-hoc*, *managed* et *master*.

- **Ad-hoc**: ces réseaux sont constitués généralement par des ordinateurs avec des cartes sans fil "normales" et ils sont configurés pour faire travailler les ordinateurs du réseau "deux à deux", tous les ordinateurs reçoivent les paquets de tous et ils envoyent leurs paquets à tous les ordinateurs du réseau. Pour le faire, on n'a pas besoin de choses spéciales, il suffit de créer un réseau avec un nom (ESSID), si possible encrypter les données en 128 octets (avec WEP) et ne pas avoir trop d'ordinateurs sur un même réseau.
- **Managed**: sur ce genre de réseaux il existe un serveur indépendant, *Access Point* (ou *Base Station* selon la terminologie commerciale d'Apple) sur lequel se branchent tous les ordinateurs. L' *access point* alors se charge d'envoyer des trames 802.11 aux destinataires. Habituellement les *access points* sont capables de faire du *roaming*, c'est à dire que les clients peuvent être en mouvement et changer de point d'accès selon la puissance du signal reçu. La différence la plus grande avec une carte qui peut être *access point* (ou *mode master*) est qu'il faut faire du *bridging* de paquets IP et manipler en plus des octets du 802.11 à très bas niveau, normalement directement avec carte. De la même manière, les *access points* offrent des services supplémentaires de *routage d'IP*, de serveur DHCP et de *bridging* comme sur une Ethernet. Quand un ordinateur ou une carte sont branchés au réseau grâce à un point d'accès, on dit qu'ils sont en mode *managed*.
- **Master**: c'est la façon de travailler de l' *access point* décrit antérieurement. Comme on le verra à la fin de l'article, il est possible de fabriquer un *access point* avec Linux.

Modules du noyau (kernel)

La plupart des cartes que l'on vend maintenant sont du genre Orinoco (Lucent), Symbol HR et Prism 2. Toutes ces cartes sont supportées par le "pilote" (driver) **orinoco_cs** inclus dans le noyau 2.4.x, mais **seulement pour travailler en mode managed ou ad-hoc**, elles ne sont pas capables de travailler en mode *master*.

Il y a en plus un petit probleme, car les *pilotes* ne sont pas tout à fait actualisés avec la dernière version et quand la carte partage les lignes d'interruptions avec des autres dispositifs, elle génère beaucoup de lignes de logs dûs au "*événements vides*". Ce problème est déjà réparé sur les dernières versions que l'on peut télécharger sur

http://ozlabs.org/people/dgibson/dldwd/[1] et les compiler. C'est assez simple et le README est assez clair.

Mais comme on le verra plus tard, il existe d'autres options, les nouveaux linux-wlan-ng[2] ou le fantastique pilote pour Prism2[3] (i.e. ceux employés par Conceptronic) qui vont nous permetre de transformer Linux en un *access point* à un prix ridicule si on a une carte Prism2 de Intersil.

Configuration des cartes: iwconfig

Une fois que les modules du noyau nécessaires ont été chargés, la configuration des cartes est similaire à celle des cartes ethernet avec la commande ifconfig mais cette fois aidé par une nouvelle commande de **iwconfig**, qui permet de changer les paramètres spécifiques des réseaux sans fils. Par exemple

- Identificateur de réseau (**essid**)
- Fréquence ou chaîne (**freq/channel**)
- Modalité (**mode**: *master|managed|ad-hoc*)
- Vitesse (**rate**)
- Clé de cryptage (**key/enc**)
- Puissance de transmission (**txpower**)
- etc.

Bref, avec **iwconfig**, on configure les paramètres spéciaux du sans fil et avec **ifconfig**, on configure les paramètres normaux du réseau IP.

Note sur le cryptage

Si on spécifie une clé (key) pour iwconfig, les transmissions seront cryptées avec le protocole WEP. Sur Linux il y a deux manières de les spécifier:

1. Avec un mot de passe : `iwconfig interface key "s:mi_clave"`. La clé doit être de 5 caractères pour le cryptage de 40 octets et de 13 pour 128 octets (en réalité, la clé ne fait que 104 octets).
2. Avec la clé en hexadécimal: `iwconfig interface key "mi_clave_en_hexa"`. Dans ce cas, on introduit directement la clé de 5 ou 14 caractères directement en hexadecimal.

Pour une plus grande sécurité, les caractères aléatoires pour former la clé sont conseillés.

Utilisateurs de Mac OS X

Pour introduire la clé sur le gestionnaire Airport du Mac OS X, il faut que les caractères soient en hexadécimal et avec un $ au début.

Configuration d'un client Linux (*Managed* ou *infrastructure*)

Il faut commencer d'abord par le plus simple: **on a déjà un *access point*** et on veut simplement faire marcher notre carte PCMCIA sur Linux, on parle alors du mode **"infrastructure"** et il consiste à mettre la carte en mode ***managed*** pour qu'il se connecte avec un *genre de point à point* avec un *access point*.

Lorsque la carte achetée possède un adaptateur PCI (comme le Conceptronic PCI), il faut **valider le support Cardbus** dans le noyau (kernel). Pour le faire, il faut aller sur "**General Setup:PCMCIA CardBus Support**" et choisir "**Cardbus Support**".

Il faut ensuite sélectioner les pilotes orinoco_cs et pour ce faire, il faut aller sur "**Networking Device Support:Wireless LAN (non-hamradio)**" et choisir les options et les sous-options du "**Hermes Chipset 802.11 support (Orinoco/Prism/Symbol)**".

C'est le moment de compiler le noyau et de regarder si on a bien installé les paquets pour les support PCMCIA, on devrait avoir un répertoire /etc/pcmcia/ contenant plusieurs fichiers. Si c'est le cas et que le noyau compilé est installé, il nous manque seulement la configuration du réseau. Étant sur Debian, je vais présenter certaines différences pour RedHat.

Debian (PCMCIA)

La configuration du réseau avec Debian peut être faite directement dans les fichiers .opts de /etc/pcmcia.

Il faut d'abord s'assurer qu'il reconnaisse automatiquement la carte Conceptronic et pour ce faire, il faut éditer le fichier **/etc/pcmcia/config.opts** et y ajouter les lignes suivantes pour qu'il charge le module orinoco:

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

On édite alors **/etc/pcmcia/wireless.opts** et on y ajoute les lignes suivantes:

```
*,*,*,*)
 INFO="Nombre..."
 ESSID="Nombre_de_red"
 MODE="Managed"
 RATE="auto"
# La clé est mise ici si il y a cryptage
 KEY="s:mi_clave"
 ;;
```

Maintenant sur **/etc/pcmcia/networks.opts** il faut saisir les données du réseau IP:

```
case "$ADDRESS" in
*,*,*,*)
 INFO="Nom...."
 # 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]
 # Rien si on a 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 et Airport)

Si on ne possède de carte PCMCIA, c'est à dire si on a un Apple avec une carte Airport ou une carte avec un adaptateur PCI, la configuration en Debian se fait directement dans **/etc/network/interfaces**. Par exemple:

```
auto eth1
# exemple avec 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 suit une philosophie différente (au moins dans la version 7.2), au lieu de configurer le réseau dans les fichiers /etc/pcmcia, elle le fait dans les mêmes fichiers où sont configurées les interfaces réseau, dans **/etc/sysconfig/network-scripts/ifcfg-ethX**. Alors les paramètres du réseau doivent être configurés dans ces fichiers, par exemple:

```
DEVICE=eth1
MODE=managed
ESSID="Nom_du_réseau"
RATE=auto
TXPOWER=auto
KEY="s:ma_cle" # Seulement si c'est encrypté
BOOTPROTO=static
IPADDR=192.168.0.3
BROADCAST=192.168.0.255
NETMASK=255.255.255.0
NETWORK=192.168.0.0
ONBOOT=yes
```

Configuration de Réseau *ad-hoc*

La configuration d'un réseau *ad-hoc* permet de monter un réseau entre deux ordinateurs **sans la nécessité d'avoir un *access point***. La manière de travailler du ad-hoc est *peer-to-peer*, tous les ordinateurs reçoivent les paquets envoyés par l'un d'entre eux. C'est pour cette raison que le réseau ad-hoc marche bien pour un petit nombre d'ordinateurs.

La **seule différence de configuration** est qu'il faut écrire ***ad-hoc*** au lieu de **managed**. Par exemple:

```
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 Nom_du_réseau
 wireless_mode ad-hoc
 wireless_key s:ma_cle
 wireless_rate auto
 wireless_nick sofi
```

Communication entre sans fil et ethernet

Vous voulez avoir une connexion entre les ordinateurs connectés en sans fil et les ordinateurs connectés par ethernet ? Ah!!!!! **Voici une des importantes tâches réalisées par les *access-points***.

Si vous pensez faire quelque chose de ce que je m'apprête à vous expliquer maintenant, il est nécessaire de connaître les bases du fonctionnement des réseaux IP et Ethernet (pour cela il y a des cours de réseaux à la Fac, ou dans les écoles d'informatique :-). Si vous ne les connaissez pas, ne tentez pas de comprendre ce qui vient ensuite, appelez un ami qui connait et/ou étudiez des cours ou des formations disponibles sur Internet.

Lors de la création d'un réseau *ad-hoc*, il est nécessaire qu'un des ordinateurs possède 2 cartes réseau, une sans fil et une autre Ethernet qui se chargera du travail du *routing*. On a de la chance parce que le protocole IP est tout à fait orienté vers cette logique et Linux peut effectuer toutes les fonctions du *IP routing*. En ce cas je vous conseille d'apprendre l'IP routing et de configurer deux réseaux IP différents, un réseau sans fil et un autre Ethernet et un des Linux fera le *routing*.

Mais en plus, il y a **d'autres options fonctionnant à un niveau plus bas et qui permettent d'avoir un seul réseau IP à partir de deux réseaux physiques différents**:

- **Proxy arp**[4]: Pour que deux machines d'un LAN Ethernet avec TCP/IP puissent communiquer, elles doivent connaître l'adresse MAC (ethernet) de l'autre ordinateur. Le protocole ARP se charge de cette tâche, qui grâce aux paquets de broadcast apprend et forme une table qui met en relation les adresses IP avec les adresses MAC (essayer l'ordre arp -a). Linux peut faire du proxy arp sans problème, il faut simplement configurer les variables disponibles dans /proc[5].
- **Bridging**: Le *bridging* est une option plus avancée que le proxy arp, et il est nécessaire d'avoir des options spéciales du noyau en plus du paquet **bridge-utils**. Si vous voulez cette option, regardez comment se configure le bridge dans la section qui suit (*Configuration de Linux comme un Access Point*).

Configuration de Linux comme un Access Point

NOTE: Monter un acces point, avec tous ses éléments, n'est pas très facile (pour les amis du Windows: ne croyez pas qu'avec windows, c'est plus facile, **en ce moment c'est impossible de faire avec Windows ce que j'explique là**, vous n'avez pas d'autre option que d'acheter un *Access Point*), car il faut connaître les réseaux, configurer et compiler le noyau et être accoutumé avec les utilitaires et configurations de PCMCIA. Si ce n'est pas votre cas, il est possible que vous ne compreniez pas ce qui va suivre, commencez plutôt avec quelque chose de plus simple, comme ce que j'ai expliqué avant ou demandez de l'aide à une personne qui l'a déjà réalisé.

Ma philosophie est que quand on ne connait pas quelque chose il faut l'apprendre, il faut le faire pas à pas, du plus simple au plus compliqué. Même si mes premier tests avec le *sans fil* sont passés par toutes les étapes antérieures, l'idée que j'avais dans la tête dès le début était d'avoir un *access point* avec mon Linux.

Je doit admetre que j'ai eu aussi beaucoup de chance, car les cartes que j'ai acheté, les Conceptronic PCI C11iDT[6]:

et la Conceptronic PCMCIA Airbridge 11CC[7]:

emploient le chipset Prism2.5 de Intersil. Je fais référence à ma chance, parce que j'ai rencontré un groupe de *fous* menés par Jouni Malinen qui sont en train de développer un excellent module de Linux pour les Prism2[3] (**Host AP**) qui **permet travailler en mode *Master***, même si le fabricant indique le contraire :-).

> On vient de me dire qu'une de nos connaissances (de Bulma) Jordi Murgó est parti de l'*équipe de fous*, et que son collègue David Farré est en train de faire un applet Gnome[8] pour montrer les statistiques de la connection.

```
[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
```

Alors finalement j'ai réussi ce que je voulais, la carte PCI que j'avais installée sur mon Linux du bureau marchait en mode Master et je pouvais interconnecter les ordinateurs avec mon Linux. iBook, MAC OS X et même l'*innommable,* avec le cryptage WEP de 128 octets.

Comme après tout, la carte PCI était simplement un PCMCIA avec un adaptateur j'ai fini par metre la PCMCIA sur mon vieux portable (P133, 32 MB RAM) pour qu'il serve de serveur d'accès pour toute la maison, avec le bridging, le serveur DHCP et le contrôle d'adresses 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]
```

J'ai aussi essayé le même portable avec les autres cartes Conceptronic que j'avais acheté (la PCMCIA de la seconde photo) et elle marche parfaitement, même si elle n'est pas aussi performante à cause de sa petite antenne, mais même de cette manière le réseau fonctionne à 11Mbps dans toute la maison, même si le portable n'est pas du tout bien placé.

Installation du module Host AP

Sur la page web du Host AP et sur le README du logiciel (un tgz), il est bien expliqué comment faire pour installer le module **hostap** avec toutes ses variantes, spécialement PCMCIA et PCI. Il faut avoir les sources du noyau que l'on emploie (on met le *chemin* dans le Makefile), il suffit ensuite de compiler et faire un make install pour installer les modules dans le répertoire correspondant (/lib/modules/2.4.18/pcmcia/ dans mon cas) et le fichier de configuration de la PCMCIA (**hostap_cs.conf**) dans le répertoire /etc/pcmcia .

Une fois installé, si on emploie l'option PCMCIA (même si on a un adaptateur PCI, on emploie généralement cette option), on doit indiquer aux modules de la PCMCIA que lors de la détection de la carte, elle doit charger le module hostmap_cs.

Pour ce faire, on édite **/etc/pcmcia/config.opts** et on ajoute les lignes suivantes:

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

***wavemon* sur Linux dans un Apple iBook avec Airport connecté avec l'Access Point**

Ça, c'est le Windows de ma fille, dans sa chambre, aussi avec un carte Conceptronic connectée au réseau sans fil avec Linux AP

Avec ça, tout le travail concernant les pilotes est terminé, il nous manque encore la configuration du réseau.

Bridging

Comme je l'ai déjà indiqué, il ne suffit pas de brancher la carte en mode *Master* pour avoir un *access point,* en plus il faut interconnecter le réseau sans fil et le réseau Ethernet. Personnellement j'ai opté pour faire du bridging entre les deux réseaux et de cette manière je traite toute la maison comme s'il n'y avait qu'un seul réseau, sans me préoccuper de trouver des adresses IP différentes si je suis branché sur Ethernet ou sur du sans fil.

Pour faire du bridging avec Linux il faut valider cette option dans le noyau:

Il faut ensuite installer le paquet **bridge-utils** qui est disponible pour Debian (pour Red Hat il faut le chercher sur rpmfind.net[9], quand à moi, je l'ai cherché et il n'y avait que la version RawHide mais elle fonctionnait parfaitement sur une RedHat 7.2).

Le logiciel principal de ce paquet est, **brctl**, qui permet de créer et configurer des interfaces *virtuelles* qui sont les interfaces sur lesquelles on apliquera le *bridging*.

Sur l'image supérieure, on peut observer d'abord la configuration du *bridge* (**br0**), qui prend en charge les interfaces **eth0** et **wlan0** (dans le module *hostap* les interfaces *sans fil* sont appelées *wlanX*). Vous remarquerez que j'ai désactivé le *spanning tree protocol* parce que je suis sûr et certain que sur mon petit réseau il n'y a pas de boucles (loops) puisqu'il est très simple, mais si vous en n'êtes pas si sûr que moi il est préférable de le laisser validé.

Nous pouvons ensuite voir (avec la commande *showmacs*) les ordinateurs branchés sur chaque réseau, le 1 est pour ethernet et le 2 est le wlan0. Celles signalées comme *local* sont les interfaces du serveur, les autres sont les interfaces distantes.

ATTENTION: les interfaces qui forment une partie d'un bridge **ne doivent pas avoir une adresse assignée**, dans les fichiers de configuration de la carte il faut mettre une IP **0.0.0.0** au lieu d'une IP réelle, puisqu'elles doivent travailler sur mode *restreint (promiscous) mode*.

Définition du br0 sur Debian

Sur Debian, il est très simple de le configurer, puisque **/etc/network/interfaces** est très flexible et puissant. Dans mon cas, j'ai simplement du mettre

```
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
```

Je vous conseille de lire aussi la discussion pour Red Hat pour mieux comprendre le procédé complet.

Définition du br0 sur Red Hat

Sur Red Hat, c'est un peu plus compliqué car il n'y a pas de fichiers d'interfaces comme sur Debian et en plus les cartes de réseau PCMCIA sont toujours configurées depuis **/etc/sysconfig/network-scripts/ifcfg-xxx**. Mais, j'ai fait un truc assez *brouillon*, j'ai un **ifcfg-eth0**, **ifcfg-wlan0** et un **ifcfg-zbr0** (le z je le met pour qu'il soit appelé après avoir appelé les autres interfaces). Voici les fichiers un par un:

/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
```

NOTE: Avec l'exécution de "/usr/bin/prism2_param wlan0 host_decrypt 1" j'oblige le décryptage WEP des trames par le pilote logiciel, au lieu de le faire à travers la carte. La possibilité ou non d'employer cette option dépend de la carte que vous avez et de la dernière version du pilote. C'est à dire qu'il faut faire l'essai ; peut être que votre carte ne marchera pas avec le cryptage 128/104 sans cette option. La commande **prism2_param** est un script inclus dans le paquet hostap et elle est utile pour simplifier les appels à **iwpriv** pour changer les paramètres de la carte.

/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
```

Il me semble que je vous ai indiqué le plus important, si j'ai oublié quelque chose, désolé, il y a déjà beaucoup trop de choses pour un seul article ... mais je voulais que vous ayez au moins un guide pour démontrer **la puissance de Linux sur le sujet du sans fil**.

Note finale: L'article est orienté pour ceux qui veulent créer un petit réseau chez eux, mais pourtant, comme Jordi Murgó l'a remarqué (un des développeurs du hostap), le module permet beaucoup d'autre choses au niveau plus professionnel et monter des réseaux très sophistiqués avec du *roaming* transparent:

- Système de distribution.
- Délégation de l'authentification.
- Notification de processus à l'utilisateur, ce qui permet de choses plus complexes comme changements de firewalls, le routing, la notification d'événements, etc.

Ricardo Galli

Traduit par: Daniel Rodriguez.

Note du traducteur: Je suis désolé les amis de ne pas savoir écrire mieux en français, mais l'article vaut bien la peine de le lire malgré les fautes d'orthographe. Si quelqu'un veux le corriger, nous en serons tous contents et l'article sera publié sur Bulma avec les coordonées du correcteur :-)

Note du correcteur: Comme toujours, les espagnols sont trop modestes, le travail de traduction est très bien fait. ;-)

Note du traducteur: Merci a tous! et remerciez le correcteur aussi.

Lista de enlaces de este artículo:

1. http://ozlabs.org/people/dgibson/dldwd/
2. http://www.linux-wlan.org/
3. http://people.ssh.com/jkm/Prism2/
4. http://www.sjdjweis.com/linux/proxyarp/
5. http://www.tldp.org/HOWTO/Adv-Routing-HOWTO-16.html#ss16.2
6. http://www.conceptronic.net/products.asp?p=C11iDT&Aktie=5&mt=C11iDT&
7. http://www.conceptronic.net/products.asp?p=CON11C&Aktie=5&mt=CON11C&
8. http://www.polakilandia.org/gwlan/
9. http://rpmfind.net/

E-mail del autor: danircJUBILANDOSEbulma.net