



## Autoconfiguración según la localización

# Movilidad

Godofredo Fernández Requena

El objetivo de este artículo es explicar cómo conseguir la autoconfiguración de nuestros interfaces de red según el entorno donde nos encontremos: en nuestra casa, en nuestro lugar de trabajo, en casa de un amigo o de un cliente donde solemos desplazarnos con frecuencia, etc.

### Introducción

La idea de elaborar este artículo se me ocurrió cuando mi portátil no era capaz de conectar a Internet justo cuando arrancaba, independientemente de la localización donde se encontrara, y así comenciar a actualizar los paquetes de mi sistema operativo (Debian GNU/Linux), ir descargando el correo electrónico, los feeds RSS, etc. Uno tiene la costumbre de aprovechar el tiempo de arranque para, alejado del ordenador, ir haciendo otras cosas... y al volver, encontrárselo todo preparado para comenzar a trabajar.

Para lograr dicho objetivo iremos recorriendo las siguientes materias:

- Instalación de los drivers apropiados para nuestra tarjeta de red inalámbrica.
- Estudio de las herramientas de configuración de red.
- Breve presentación práctica de conceptos de programación en shell (scripting).

Terminaremos juntándolo todo en nuestro programa de autoconfiguración y viendo algunas herramientas de monitorización del interfaz de red WiFi que nos ayuden a detectar si se ha apagado el punto de acceso, si nos hemos alejado demasiado de él... En definitiva, si hay problemas en la señal que recibimos, etc.

### Detectando el hardware: drivers

Para poder disfrutar de nuestra tarjeta de red inalámbrica necesitamos que nuestro sistema GNU/Linux sea capaz de detectar nuestra tarjeta WiFi y permitir, tras ello, que seamos capaces de configurarla según nuestras necesidades.

El soporte de tarjetas WiFi en el kernel Linux sigue siendo escaso a día de hoy, principalmente porque muchos fabricantes no abren el código de sus drivers; ni siquiera los facilitan, y esto hace que el desarrollo de los mismos sea complicado. No obstante, cada vez se van incorporando más drivers a la actual rama de desarrollo del núcleo: 2.6.

A continuación veremos qué chipsets (conjunto de componentes electrónicos utilizados en la construcción de la tarjeta de red) están soportados directamente en el kernel, y algunos proyectos externos a este que existen para propiciar los drivers adecuados a aquellas tarjetas basadas en chipsets populares.

Por tanto, lo primero que tenemos que conocer es qué chipset utiliza nuestra tarjeta. Esta información suele venir en la propia caja utilizada como envoltorio y en la documentación que la acompaña. También la podemos consultar en la web del fabricante, o en las listas de dispositivos soportados que están disponibles en las páginas web de los distintos proyectos que presentamos en este apartado.

Es habitual el uso de un mismo chipset por distintos fabricantes, y encontrar así qué diferentes productos comerciales están soportados por un único driver.

### Soportados en el kernel

En la fecha en la que escribimos este artículo la versión estable del kernel es la 2.6.16.14. Y los drivers que soportados por esta versión pueden ser consultados en el siguiente enlace:

```
http://www.kernel.org/git/?p=linux/
kernel/git/stable/linux-2.6.16.y.git;
a=tree;h=08a4253093e76e0b6a2e9b4ccfc59200
070244d;hb=bf7d8bacaaf241a0f0157986fd4e1e6
834873d50;f=drivers/net/wireless
```

En esa página web tenemos un listado con los ficheros que contienen el código fuente de los drivers soportados por esta versión del kernel. Mirando el nombre de muchos de ellos podemos deducir los chipsets soportados. Para obtener más información, al lado de cada fichero aparecen dos palabras: "blob" e "history". Si, por ejemplo, en la fila del fichero "airo.c" pulsamos sobre la palabra "blob" veremos el código fuente de dicho fichero y, al principio de este, un pequeño resumen sobre el driver y los dispositivos que soporta. En este caso: "Aironet driver for 4500 and 4800 series cards".



Recientemente han sido incorporados los drivers de los chipset "Intel Pro Wireless 2100 y 2200", este hito es importante ya que estas tarjetas vienen integradas en muchos ordenadores portátiles.

Si comprobamos que nuestra tarjeta está soportada, y el kernel suministrado en nuestra distribución no la tiene aún soportada, tendremos que compilar la última versión disponible de éste. No es objeto de este artículo ofrecer una explicación detallada de cómo compilar el kernel Linux; es por ello por lo que no nos extenderemos más en este punto.

### MadWiFi

Este proyecto desarrolla los drivers para el chipset "Multiband Atheros". La página web del proyecto es <http://madwifi.org/>.

Algunas de las tarjetas soportadas son 3Com® Wireless 11a/b/g PCI Adapter, y una amplia gama de las marcas DLink, Netgear y SMC Networks. Un listado detallado lo encontramos en <http://madwifi.org/wiki/Compatibility#HardwareSupport> edbyMadwifi.

Para obtener información detallada sobre el proceso de instalación podemos consultar en <http://madwifi.org/wiki/UserDocs/FirstTimeHowTo>.

En Debian existen tres paquetes que nos permitirán compilar el driver y utilizarlo: madwifi-dev, madwifi-source y madwifi-tools.

### ZyDAS ZD1211 802.11b/g USB WLAN chipset

Este proyecto desarrolla los drivers para el chipset "ZyDAS ZD1211 802.11b/g USB WLAN". La página web oficial del proyecto es <http://zd1211.ath.cx/>.

En ella podemos encontrar una lista de dispositivos USB soportados, así como las instrucciones de instalación de dichos drivers (la página web es extensa y el apartado "installation" se encuentra casi al final, y está descrito paso por paso).

En Debian existen los siguientes paquetes que nos permitirán compilar el driver y utilizarlo: zd1211-source y zd1211-firmware.

### NDISWrapper

Este proyecto implementa el API del kernel de Windows y el de NDIS (Network Driver Interface Specification) dentro del kernel de Linux. De esta forma podemos utilizar los drivers aportados por muchos fabricantes para el sistema operativo Windows, pero haciéndolos trabajar en nuestro sistema Linux. El driver se ejecuta nativamente, como si estuviera instalado en un sistema Windows, sin emulación binaria.

En la siguiente página podemos encontrar un Wiki con mucha información sobre tarjetas soportadas, proceso de instalación, etc.: [http://ndiswrapper.sourceforge.net/mediawiki/index.php/Main\\_Page](http://ndiswrapper.sourceforge.net/mediawiki/index.php/Main_Page).

En Debian GNU/Linux podemos encontrar los siguientes paquetes que nos suministran este software: ndiswrapper-common (Userspace utilities for ndiswrapper), ndiswrapper-source (Source for

the ndiswrapper linux kernel module), ndiswrapper-utils (Userspace utilities for ndiswrapper) y ndiswrapper-utils-1.7 (Userspace utilities for ndiswrapper).

Existe un proyecto parecido a este, puesto en marcha por la compañía Linuxant (<http://www.linuxant.com/driverloader/>), que suministra una capa de software que hace utilizables los drivers existentes para el sistema operativo Windows en un sistema Linux. Sin embargo este proyecto pertenece a una empresa privada y requiere licencias de uso tras un determinado tiempo de prueba.

### Consejos para el éxito

El autor de este artículo ha conseguido instalar los siguientes modelos de tarjetas WiFi:

- Intel Pro Wireless 2200, compilando el correspondiente módulo del kernel Linux.
- Wireless 11a/b/g PCI Adapter (3CRDAG675B), haciendo uso de los drivers desarrollados por el proyecto MadWifi.
- Linksys WMP54G, haciendo uso de la capa software desarrollada por el proyecto NDISWrapper.

Lo más importante es asegurarse de conocer el chipset en el que se basa vuestro hardware, buscar el proyecto que lo soporta y leer atentamente las instrucciones de instalación. Quizá, el inconveniente aquí para muchos principiantes sea manejarse con los procesos de compilación del kernel y sus módulos. Es por ello muy recomendable familiarizarse con estos conceptos lo antes posible para continuar ensanchando nuestra aventura GNU/Linux.

### Autoconfiguración según la localización

Una vez que hemos conseguido detectar nuestro hardware, nuestra tarjeta de red WiFi, tenemos que configurarla para poder transmitir y recibir información a través de ella.

Veremos una serie de herramientas de configuración que, paso a paso, nos irán permitiendo ajustar una serie de parámetros en nuestro hardware. Posteriormente indicaremos breves conceptos de programación de shell que serán los que nos permitan automatizar todo el proceso de configuración en el arranque de nuestro sistema.

### Herramientas de configuración de red

El primer paso que tenemos que dar para configurar un interfaz de red inalámbrico es decidir su modo de funcionamiento. Existen dos modos de funcionamiento dentro del estándar 802.11. El modo ad-hoc nos permite configurar una red de comunicaciones entre dispositivos iguales, por ejemplo varios portátiles junto con algunas PDA, etc., sin necesidad de punto de acceso. El modo infraestructura exige, por el contrario, la presencia de un punto de acceso con el cual sincronizan el resto de dispositivos, cursando este todas las comunicaciones a su través.

**Es conveniente utilizar el DNS suministrado por el servidor de acceso**



```
[13:04:32(root@olivo)/home/gfr]> iwconfig
lo          no wireless extensions.

eth0       no wireless extensions.

eth1       unassociated  ESSID:"GodoNet"
           Mode:Managed Channel=0 Access Point: Not-Associated
           Bit Rate=0 kb/s Tx-Power=20 dBm
           Retry limit:7 RTS thr:off Fragment thr:off
           Encryption key: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:0 Invalid misc:0 Missed beacon:0

sit0       no wireless extensions.
```

Figura 1. Interfaz eth1 con capacidades WiFi.

El modo habitual de funcionamiento utilizado tanto en nuestros hogares como en lugares de trabajo es el modo infraestructura: existe un punto de acceso que normalmente se utiliza como puerta de acceso a Internet al que se van asociando el resto de dispositivos móviles.

Veremos a continuación qué parámetros son los habituales para lograr dicha asociación con el punto de acceso, y cómo se configuran estos.

### Interfaz inalámbrico

Si el proceso de detección de hardware ha sido satisfactorio al ejecutar el siguiente comando en un terminal, y como usuario root, obtendremos algo parecido a lo que podemos ver en la figura 1:

```
iwconfig
```

Observamos que los interfaces: lo, eth0, y sit0 no tienen capacidades WiFi; sin embargo el interfaz eth1 sí que las posee. De hecho vemos, en la primera línea de información perteneciente a este interfaz, que su estado es "unassociated" y que se ha intentado asociar a una red WiFi con identificación "GodoNet". En la siguiente línea vemos información sobre el modo de operación: "Managed" (correspondiente al modo "Infraestructura" descrito anteriormente), sobre el canal de radiofrecuencia utilizado: "0" y sobre el punto de acceso al cual nos hemos asociado (por el momento vemos que no estamos asociados a ninguno).

Pasamos pues a configurar el interfaz de red inalámbrico con el objetivo de asociarnos al punto de acceso apropiado. Para ello necesitamos el valor de algunos parámetros que ya han sido configurados en el punto de acceso, por ejemplo:

- Nombre de la red (ESSID). Es conveniente que tras configurar este nombre en los puntos de acceso evitemos su difusión. Así dificultamos que alguien con malas intenciones pueda detectarlo, con lo que dispondría de la primera llave para el acceso a nuestra red inalámbrica.
- Si hemos decidido proteger nuestras comunicaciones con alguna clave de cifrado, también necesitaremos el valor hexadecimal correspondiente de esta, algo como esto:

```
53B3-63F5-3AC6-42CD-33C8-4F77-0E
```

Es posible configurar algunos parámetros más, pero con estos dos es suficiente para lograr establecer nuestra asociación con el punto de acceso. Para saber más sobre los parámetros de configuración se recomienda leer la página de manual:

```
man iwconfig
```

### wireless-tools

Para poder utilizar las herramientas de configuración y monitorización del interfaz inalámbrico es necesario haber instalado el paquete (conjunto de software) que las suministra. En la distribución Debian GNU/Linux basta con ejecutar el siguiente comando:

```
apt-get install wireless-tools
```

Tras haberse instalado correctamente, tendremos del siguiente conjunto de herramientas, todas ellas situadas en el directorio "/sbin" (y por tanto sólo utilizables por el superusuario de nuestro sistema):

- iwconfig
- iwevent
- iwgetid
- iwlist
- iwpriv
- iwspy

En la figura 2 podemos ver un ejemplo del uso de la herramienta iwlist. Esta herramienta es muy útil para detectar las capacidades soportadas por nuestra tarjeta de red. Con la opción "rate" obtenemos una lista de velocidades soportadas por nuestra tarjeta de red inalámbrica; con la opción "key" un listado de posibles claves de cifrado, etc. La opción más importante es "scanning", que permite realizar un rastreo de frecuencias en busca de señales WiFi, y así, de posibles redes abiertas que nos permitan asociarnos. Esta opción nos dará información sobre los valores utilizados por las distintas redes encontradas en sus parámetros de configuración. Posteriormente podremos utilizar esta información para configurar nuestro interfaz haciendo uso de la herramienta iwconfig.

Una vez conocido el nombre de red a la que nos deseamos asociar, y la clave de cifrado que tenemos que utilizar, bastará con ejecutar el siguiente par de comandos (como usuario root, insistimos) para lograrlo, siempre que estemos en la zona de cobertura de la señal:

```
iwconfig eth1 essid GodoNet
```

```
iwconfig eth1 key
53B363F53AC642CD33C84F770E
```

Para comprobar que estamos asociados al punto de acceso que difunde dicho nombre de red, volvemos a ejecutar el comando:

```
iwconfig
```

Si todo ha ido bien debemos de obtener un resultado similar al mostrado en la figura 3.



Podemos observar las diferencias entre la información mostrada en esta figura (ya asociados a un punto de acceso) y la mostrada en la figura 1 (aún no asociados).

El aprovechamiento de algunas de las diferencias obtenidas entre ambos estados será parte fundamental de nuestro script de autoconfiguración.

### Interfaz lógico

Una vez que hemos conseguido asociarnos a un punto de acceso y que, por tanto, nuestro interfaz físico ya está "levantado", hemos de realizar la configuración lógica del mismo: dirección IP, máscara de red y puerta de enlace.

Para la realización de estas tareas utilizaremos el comando:

```
ifconfig
```

En la distribución Debian GNU/Linux esta herramienta es suministrada por el paquete net-tools. En la instalación básica de cualquier sistema Debian este paquete debe de haber sido instalado; si no, bastará con ejecutar el siguiente comando como usuario root:

```
apt-get install net-tools
```

### net-tools: ifconfig

ifconfig es una herramienta muy potente de obtención de información de red y de configuración de interfaces. Es por ello por lo que recomendamos la lectura de su página de manual a todo aquel lector que quiera profundizar en su conocimiento (man ifconfig).

En este artículo sólo vamos a explicar su uso básico: la utilizaremos para asignar la información de red anteriormente descrita al interfaz correspondiente.

En la figura 4 podemos ver la información suministrada por ifconfig sobre un interfaz que aún no ha sido configurado.

Configuramos el interfaz con la siguiente instrucción:

```
ifconfig eth1 inet 192.168.44.17 netmask 255.255.255.0
```

Tras la palabra "inet" aparece la dirección IP (192.168.44.17) y tras la palabra "netmask" la máscara de red (255.255.255.0). Esta información ha de ser coherente con la configurada en el punto de acceso así como en el resto de dispositivos de nuestra red. Supongamos pues que nuestro punto de acceso ha sido configurado con la dirección IP 192.168.44.1 y con igual máscara de red que la utilizada (255.255.255.0). Sólo queda configurar la puerta de enlace por defecto que nuestro ordenador va a utilizar para acceder a información que no se encuentra directamente en nuestra red local, sino en el resto de redes (por ejemplo en Internet). Para ello utilizaremos el siguiente comando (ejecutado como usuario root):

```
route add default gw 192.168.44.1
```

```
[16:40:30(root@olivo)/home/gfr]> iwlist eth1 rate
eth1      12 available bit-rates :
          1 Mb/s
          2 Mb/s
          5.5 Mb/s
          11 Mb/s
          6 Mb/s
          9 Mb/s
          12 Mb/s
          18 Mb/s
          24 Mb/s
          36 Mb/s
          48 Mb/s
          54 Mb/s
          Current Bit Rate=0 kb/s

[22:26:34(root@olivo)/home/gfr]> iwlist eth1 scanning
eth1      Scan completed :
          Cell 01 - Address: 01:10:A9:03:2E:EB
                   ESSID: "<hidden>"
                   Protocol:IEEE 802.11g
                   Mode:Master
                   Channel:6
                   Encryption key:on
                   Bit Rates:54 Mb/s
                   Extra: Rates (Mb/s): 1 2 5.5 6 9 11 12 18 24 36 48 54
                   Quality=87/100  Signal level=-42 dBm
                   Extra: Last beacon: 62ms ago
          Cell 02 - Address: 10:01:48:3C:67:75
                   ESSID: "007113131"
                   Protocol:IEEE 802.11bg
                   Mode:Master
                   Channel:1
                   Encryption key:on
                   Bit Rates:54 Mb/s
                   Extra: Rates (Mb/s): 1 2 5.5 6 9 11 12 18 24 36 48 54
                   Quality=29/100  Signal level=-82 dBm
                   Extra: Last beacon: 244ms ago
```

La herramienta route también es suministrada por el paquete net-tools en Debian GNU/Linux. Para consultar más información acerca de esta herramienta se recomienda consultar la página de manual (man route).

En la figura 5 podemos ver la información suministrada por ifconfig acerca del interfaz que acabamos de configurar, así como la información suministrada por el comando route.

Para comprobar que nuestra configuración de red es correcta y que somos capaces de intercambiar información con el punto de acceso, podemos utilizar el siguiente comando:

```
ping 192.168.44.1
```

Si todo ha sido configurado correctamente debemos recibir respuestas del punto de acceso (con dirección IP 192.168.44.1), tal y como se muestra en la figura 6.

### Servidor de nombres de dominio (DNS)

Uno de los principales problemas que nos encontramos los usuarios de dispositivos móviles a la hora de configurar nuestros parámetros de red es el de los servidores de nombres.

Estos servidores de nombres vienen identificados por su correspondiente dirección IP, y son utilizados para obtener la dirección IP correspondiente a un determinado nombre utilizado en Internet.

Figura 2. Ejemplo de uso de la herramienta iwlist.



```
[17:02:16(root@olivo)/home/gfr]# iwconfig eth1
eth1      IEEE 802.11g  ESSID:"Godonet"
          Mode:Managed  Frequency:2.437 GHz  Access Point: 01:10:A9:03:2E:EB
          Bit Rate=54 Mb/s   Tx-Power=20 dBm
          Retry limit:7  RTS thr:off  Fragment thr:off
          Encryption key:53B3-63F5-3AC6-42CD-33C8-4F77-0E  Security mode:open
          Power Management:off
          Link Quality=84/100  Signal level=-46 dBm  Noise level=-86 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0  Missed beacon:0
```

Figura 3. Información de un interfaz asociado a un punto de acceso.

Por ejemplo, el nombre "godest.vivencias.net" corresponde a la dirección IP 208.97.140.210; y, aunque los humanos utilizamos nombres para trabajar más cómodamente, los ordenadores los traducen a direcciones IP que son las que utilizan para intercambiar información entre ellos.

Los DNS que debemos utilizar suelen variar, al igual que la dirección IP, máscara y puerta de enlace, dependiendo de la localización donde nos encontremos. Así, el proveedor de acceso a Internet que utilizamos en nuestro hogar nos ofrecerá unos determinados DNS, mientras que en el lugar de trabajo tendremos que utilizar otros distintos. Es conveniente utilizar en cada sitio el ofrecido por el proveedor de acceso o por el administrador correspondiente, porque entre otras cosas el uso de otros distintos puede estar filtrado, y sobre todo porque estarán más cerca de nosotros (disminuyendo el tiempo necesario para obtener una respuesta) y pueden poseer información sobre recursos locales que no podremos obtener, y utilizar, si estamos consultando otros distintos.

Hemos considerado resolver estos problemas mediante el uso de la herramienta pdnsd por las ventajas que comentaremos a continuación. Para instalarla en Debian GNU/Linux basta con ejecutar:

```
apt-get install pdnsd
```

Las ventajas de utilizar esta herramienta son las siguientes:

- Funciona como un servidor proxy de DNS con cache permanente en disco duro. Esto quiere decir que una vez que resolvamos un determinado nombre (obtenemos su correspondiente IP) esta información quedará almacenada en nuestro disco duro local hasta que expire un determinado temporizador. La próxima vez que tengamos que acudir a ese determinado nombre no tendremos que resolverlo en el correspondiente servidor de DNS sino que lo haremos localmente, disminuyendo el tiempo de resolución y el ancho de banda utilizado.
- Podemos configurar varios servidores DNS a los que preguntaremos, así como determinar algunos de estos para que sean los que resuelven las consultas sobre recursos específicos. Esto nos permitirá configurar una única vez, y en un único archivo, todos los servidores de DNS que solemos utilizar: el de casa, el del trabajo, el de la localización x donde solemos desplazarnos con frecuencia, etc. Será la propia herramienta la que intente detectar cuáles de ellos están dis-

ponibles para ser utilizados, y los que no respondan a las pruebas de detección se desearán. La herramienta hace un sondeo de todos ellos cada cierto tiempo y los clasifica como disponibles o desechados según responda satisfactoriamente al sondeo o no.

- Protege sobre eventuales caídas de servidores de DNS, ya que si tenemos la información disponible en nuestro caché no tenemos que consultarlos (somos autosuficientes durante el tiempo de vida de la información de la caché).
- Podemos hacer de proxy DNS para el resto de dispositivos de nuestra red, evitando consumo de ancho de banda innecesario al intentar estos dispositivos resolver nombres que nosotros ya hemos resuelto. La configuración que veremos en este artículo tiene que ser modificada para ser capaces de realizar esta función. Tendremos que cambiar la dirección IP donde escucha nuestro demonio (la 127.0.0.1 que aquí utilizamos) a una correspondiente a un interfaz de red alcanzable para el resto de las máquinas de nuestro entorno.

### pdnsd

Una vez instalada la herramienta hemos de configurarla con los parámetros adecuados y arrancar el demonio (programa correspondiente que es ejecutado de forma transparente para el usuario final, y que escucha las peticiones que las demás aplicaciones le van realizando).

El archivo de configuración correspondiente a la herramienta pdnsd se halla en el directorio "/etc" con nombre "pdnsd.conf".

Un ejemplo de configuración del mismo puede consultarse en el listado 1, disponible en el CD-ROM y que no reproducimos aquí por motivos de espacio.

Es recomendable la lectura de las siguientes páginas de manual para conseguir un uso eficiente y particularizado del mismo:

```
apropos pdnsd
```

```
pdnsd (8) - dns proxy daemon
```

```
pdnsd-ctl (8) - controls pdnsd
```

```
pdnsd.conf (5) - The configuration file
for pdnsd config
```

Para poder utilizarlo bastará con arrancarlo como demonio del sistema con la siguiente instrucción (escrita en una sola línea):

```
/usr/sbin/pdnsd -c /etc/pdnsd.conf -daemon
-debug -p /var/run/pdnsd.pid
```

Hemos añadido la opción "-debug" porque creemos que las primeras veces es conveniente ejecutarlo en modo de depuración de fallos. Haciéndolo así, podemos consultar el archivo "/var/cache/pdnsd/pdnsd.debug" para ver, por ejemplo, qué servidores de los utilizados han respondido exitosamente a la prueba de alcanzabili-



dad y cuáles no; cuáles están siendo utilizados para la resolución de nombres; ver información intercambiada durante el proceso, etc.

Si detectamos que algo va mal, tendremos que volver a editar el archivo de configuración, rectificarlo, guardar los cambios y volver a arrancar el demonio con las siguientes dos instrucciones:

```
/etc/init.d/pdnsd stop
```

```
/usr/sbin/pdnsd -c /etc/pdnsd.conf -daemon
-debug -p /var/run/pdnsd.pid
```

Con lo realizado hasta ahora, la herramienta pdnsd ya está en marcha y trabajando correctamente. Ahora sólo nos queda decirle a todo nuestro sistema que utilice como servidor de DNS a nuestro servidor proxy de DNS local. Esto se consigue editando, como usuario root, el archivo: "/etc/resolv.conf" y dejando como único contenido de éste la siguiente línea:

```
nameserver 127.0.0.1
```

Ya que es en esa dirección IP (localhost), y en el puerto 53, donde está escuchando nuestro demonio "pdnsd" (como podemos apreciar en su archivo de configuración, líneas 12 y 11 respectivamente del listado 1).

Nota: si se está utilizando la herramienta "resolvconf - nameserver information handler" lo indicado en este apartado puede ser ligeramente diferente, y puede surgir algún conflicto que tendrá que ser eliminado. Recomendamos a los lectores que vayan a realizar la configuración que proponemos en sus sistemas, que se informen sobre cómo integrar ambas herramientas o, como ha hecho el autor de este artículo, eliminen la herramienta "resolvconf" con el siguiente comando:

```
apt-get remove resolvconf
```

### Sincronizando nuestra hora: rdate, ntp

A estas alturas del artículo, si todo ha ido bien, hemos de ser capaces de utilizar nuestras aplicaciones favoritas con las que trabajamos en Internet: nuestro navegador, nuestro lector de correo electrónico... e incluso herramientas para sincronizar la hora de nuestra máquina con servidores de hora disponibles en la red.

A modo de comprobación, y como divulgación de herramientas no tan conocidas por el público en general, en este apartado vamos a ver cómo se utilizan un par de herramientas de sincronización horaria. La primera de estas herramientas que vamos a utilizar es "rdate". Es suministrada por el paquete del mismo nombre (instalamos con: apt-get install rdate) y su uso es muy sencillo: basta ejecutar el comando rdate y pasarle como argumento el nombre (o dirección IP) del servidor de tiempo del cual queremos obtener la hora.

Por ejemplo, el Centro Informático y Científico de Andalucía (CICA) dispone de un servidor público de hora del cual podemos obtener esta información así:

```
[17:21:50(root@olivo)/home/gfr]# ifconfig eth1
eth1      Link encap:Ethernet  Hwaddr 00:14:CB:57:45:07
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14971 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
          Interrupt:193 Base address:0xa000 Memory:dfcff000-dfcfffff
```

```
rdate ntp.cica.es
```

Mientras que esta herramienta básica rdate (man rdate) sólo sincroniza nuestra máquina con el servidor en el instante de tiempo en el que se ejecuta, existe otra herramienta (es un protocolo estandarizado) un poco más compleja que mantiene sincronizada nuestra máquina con los servidores que le indiquemos en su archivo de configuración. Este protocolo se denomina ntp (network time protocol).

Para su utilización es necesario ejecutar un demonio que correrá en nuestro sistema manteniendo la sincronización entre este y los servidores indicados. El demonio que corre en nuestra máquina es la herramienta software que utilizamos. El diálogo entre esa herramienta software y las instaladas en los servidores que consultamos es lo que normaliza el protocolo NTP.

Al instalar los siguientes paquetes el demonio quedará dispuesto para ejecutarse al inicio de nuestro sistema, siendo su funcionamiento transparente para el usuario final.

Los paquetes que tenemos que instalar si utilizamos Debian GNU/Linux son:

```
apt-get install ntp ntp-server ntp-simple
```

El archivo de configuración donde añadimos los servidores con los que sincronizar nuestra hora es "/etc/ntp.conf", y en las últimas versiones de los paquetes anteriores ni siquiera es necesario editarlo ya que viene configurado para utilizar un conjunto de servidores de hora públicos agrupados tras el nombre:

```
pool.ntp.org.
```

El uso de servicios de sincronización de hora es muy importante en entornos de red donde se realizan tareas de trazabilidad de mensajes intercambiados entre sistemas, copias de seguridad, auditorías de red, etc., ya que permiten un seguimiento de las incidencias en orden cronológico, consiguiendo el detalle de cuál ocurrió antes y pudo ser causa de otra que le siguió, y así sucesivamente.

### Programación shell

En este apartado vamos a ver las instrucciones básicas que posteriormente utilizaremos para construir nuestro programa de autoconfiguración de nuestra tarjeta de red inalámbrica.

En el mundo Unix los intérpretes de comandos, aquellos programas que interactúan con el usuario a través de ventanas de terminal (xterm, rxvt, konsole...) son capaces de interpretar un fichero de

Figura 4. Información lógica de interfaz de red aún no configurado.



## Listado 2

## Script de autoconfiguración

```
#!/bin/sh

#
# Suponemos inicialmente que estamos en casa
# Configuración del interfaz WiFi
#
/sbin/iwconfig eth1 essid GodoNet
/sbin/iwconfig eth1 key 53B363F53AC642CD33C84F770E

#
# Esperamos 5 segundos para que de tiempo a detectar el punto de acceso
#
echo "Intentando detectar el punto de acceso de casa..."
sleep 5

#
# Recogemos el estado del interfaz WiFi: ¿es "unassociated"?
#
estado="`/sbin/iwconfig eth1 | /usr/bin/awk '/AdmorumNet/ {print $2}`"

# Podemos comprobar lo almacenado en la variable estado
# descomentando la siguiente línea
#echo $estado

#
# El estado será "unassociated" si no encontramos el punto de acceso
# WiFi, en este caso seguimos buscando. Sin embargo, si lo encontramos
# configuramos el interfaz de red correspondiente.
#

if [ $estado != "unassociated" ] ; then
    echo "¡Bienvenido a casa! Configurando interfaz de red WiFi..."
    /sbin/ifconfig eth1 inet 192.168.44.17 netmask 255.255.255.0
    /sbin/route add default gw 192.168.44.1
    # Podemos desactivar el resto de interfaces
    # que no utilicemos con el siguiente comando:
    #/sbin/ifconfig eth0 down
    # Para trabajar sin proxy http
    export http_proxy=""
else
    echo "No asociado a punto de acceso de casa."
    echo "Intentamos asociarnos al punto de acceso del trabajo..."
    echo "Si no lo conseguimos es que estamos -cableados-"
    # Configuramos la interfaz inalámbrica
    /sbin/iwconfig eth1 essid GodoWork
    /sbin/iwconfig eth1 key off
    # Esperamos mientras nos asociamos
    sleep 5
    # Comprobamos el estado de la asociación
    estado="`/sbin/iwconfig eth1 | /usr/bin/awk '/GodoTSol/ {print
$2}`"

    # Analizamos lo ocurrido
    if [ $estado != "unassociated" ] ; then
        echo "¡Trabaja duro pero sin estresarte! Configurando
interfaz de red WiFi..."
        /sbin/ifconfig eth1 inet 192.168.1.17 netmask 255.255.255.0
        /sbin/route add default gw 192.168.1.2

        # Para trabajar sin proxy http
        export http_proxy="http://proxy.godonet"
    else
        #####
        # DETECCIÓN DE REDES CABLEADAS
        #####

        # Tiramos el interfaz inalámbrico
        /sbin/ifconfig eth1 down

        #
        # Red de un cliente
        #

        # Levantamos la interfaz cableada
        /sbin/ifconfig eth0 inet 192.168.162.142 netmask
255.255.255.224
        /sbin/route add default gw 192.168.162.129
    fi
fi
```

órdenes. Este fichero está escrito utilizando palabras clave que los programadores de los intérpretes ponen a nuestra disposición para poder realizar diversas tareas. A estos ficheros se les denominan comúnmente "scripts" y en algunos textos esta palabra ha sido traducida como "guiones", en referencia a la sucesión de comandos shells que almacenan.

Existen diferentes intérpretes de comandos, diferentes shells, en el mundo GNU/Linux. La shell por defecto utilizada en muchas distribuciones es "bash"; es por ello por lo que nuestro script estará construido utilizando el lenguaje que entiende esta shell. Y es por ello por lo que nuestro fichero comienza con la siguiente línea:

```
#!/bin/sh
```

Si utilizamos una shell de otra familia (Korn o C) esta línea cambiará de forma apropiada. Evitamos extendernos más aquí por no ser materia de este artículo.

Pasamos a describir brevemente la sintaxis y la función que realizan los siguientes comandos de la shell que utilizaremos en nuestro script:

- echo: La sintaxis de este comando es:

```
echo cadena_de_caracteres
```

La función que realiza es imprimir en pantalla la cadena de caracteres que se le pasa como argumento. La utilizaremos para ir señalando lo que el programa está haciendo.

- sleep: La sintaxis de este comando es:

```
sleep numero[sufijo]
```

Los corchetes indican que el sufijo puede o no aparecer. Si no aparece, la cantidad indicada por "numero" es considerada como segundos. La función que realiza es detener la ejecución del script durante la cantidad de tiempo indicada. La utilizaremos para que nuestro programa espere, durante un tiempo de estabilización, los resultados de las órdenes ejecutadas: proceso de asociación a un punto de acceso.

- |: Esta rayita vertical es el operador tubería, y lo que hace es enviar la salida del comando\_1 como entrada del comando\_2. Por ejemplo:

```
echo "hola godon" | grep go
```

La salida del comando echo ahora no aparece en pantalla sino que es redirigida a la herramienta grep, que busca la ocurrencia del patrón go en la cadena que está recibiendo, y si lo encuentra lo imprimirá por pantalla.

- awk: No es una instrucción interna de la shell sino una herramienta externa (como grep) que hemos de instalar en nuestro sistema de forma independiente (apt-get install mawk), aunque en una instalación normal de Debian GNU/Linux debe de estar ya instalada. Es una potente herramienta de procesamiento de patrones de texto. Nosotros solamente la vamos a utilizar para el siguiente propósito: le enviaremos todas las



líneas que suministran información sobre un interfaz de red (siendo obtenida esta información con `iwconfig`), filtraremos la línea que contiene un determinado patrón (el identificador de nuestra red inalámbrica) y seleccionaremos el segundo campo de esta línea, que nos indicará si estamos o no asociados al punto de acceso. Si estamos asociados su valor será "IEEE", si no lo estamos será "unassociated".

- `if [ ] ; then ... else ... fi`: La sintaxis de esta instrucción es la siguiente:

```
if [ condición ] ; then
bloque_1 de instrucciones
else
bloque_2 de instrucciones
fi
```

Como en otros muchos lenguajes de programación, la shell nos brinda la posibilidad de evaluar una determinada condición; si el resultado de la evaluación es verdadero ejecutaremos el bloque de instrucciones que sigue a la sentencia `if` (bloque\_1), si es falso ejecutaremos el bloque que sigue a la sentencia `else` (bloque\_2).

Es importante señalar el hecho de que estos bloques `if... else` pueden anidarse, como veremos en nuestro script.

- `export`: La sintaxis de este comando es la siguiente:

```
export nombre_variable="valor"
```

Utilizaremos esta sentencia para exportar variables útiles que hemos de configurar para el resto de aplicaciones según el entorno donde estemos ubicados. Por ejemplo, la herramienta de gestión de paquetes en la distribución Debian, `apt`, utiliza la variable `http_proxy` para actualizar y descargar los paquetes si estamos en una localización que exija el uso de un proxy para acceder a Internet utilizando el protocolo `http`. Esto suele ser habitual en muchas organizaciones.

- `$?`: Esta variable interna de la shell recoge el resultado devuelto por el último programa ejecutado. La utilizaremos para saber si nuestra prueba de conectividad (utilizaremos la herramienta `ping` para ello) en una determinada localización ha sido satisfactoria. Si lo ha sido es que estamos en ella, si no, seguiremos comprobando si nos encontramos en el resto de localizaciones. Cuando nos conectamos a una red inalámbrica con identificador (ESSID) no es necesario detectar si hemos conseguido conectar a ella o no de esta forma pues lo hacemos analizando la información de red suministrada por el interfaz WiFi, pero si nuestro script también va a ser utilizado para conectarnos a redes fijas utilizando un cable Ethernet, tendremos que saber si nos estamos conectando en una determinada localización o en otra. Como el interfaz de red cableado no posee un nombre de red (ESSID) utilizaremos

## Listado 2 (continuación)

## Script de autoconfiguración

```
interfaz          # Esperamos 3 segundos para que de tiempo a levantar el
                  sleep 3
                  # Hacemos un ping a una máquina de dicha red
                  /bin/ping -c 1 -w 1 192.168.162.129
cero              # Si está viva la máquina el comando anterior devuelve
"vivo"           # que se almacena de la siguiente forma en la variable
                  vivo=$?
                  if [ $vivo -eq 0 ] ; then
                  echo "Bienvenido a la red del cliente"
                  else
                  #
                  # Red de un amigo
                  #
                  # Limpiamos el interfaz
                  /sbin/ifconfig eth0 down
                  # Esperamos...
                  sleep 3
                  # Y lo configuramos apropiadamente
                  echo "Intentando conexión cableada en red de tu
amigo."          # Leemos la configuración del archivo /etc/net-
work/interfaces-v02
                  /sbin/ifup -i /etc/network/interfaces-v02 -a
                  # Esperamos 3 segundos para que de tiempo a
levantar el interfaz
                  sleep 3
                  # Hacemos un ping al router
                  /bin/ping -c 1 -w 1 10.69.174.1
devuelve cero    # Si está viva la máquina el comando anterior
                  vivo=$?
                  if [ $vivo -eq 0 ] ; then
                  echo "Bienvenido a la diversión."
                  else
                  #Llegados a este punto es que no hemos
sido             #capaces de asociarnos a ninguna red.
                  #Desactivamos el interfaz cableado
                  /sbin/ifconfig eth0 down
                  fi
                  fi
fi              fi
#
# Configuramos el DNS cache
#
echo
echo "Activando el servicio de DNS cache (pdnsd)..."
/usr/sbin/pdnsd -c /etc/pdnsd.conf -daemon -debug -p /var/run/pdnsd.pid
# Esperamos unos segundos para que se establezcan las conexiones.
# Lo hacemos enviando pings, lo cual ayuda a que los routers ADSL
# establezcan la conexión con internet
/bin/ping -c 3 -w 1 131.111.8.35
sleep 5
# Sincronizamos la hora con un servidor en internet
echo "Actualizando hora..."
rdate 131.111.8.35 &
# Fin del script
```



```
[22:50:16(root@olivo)/home/gfr]> ifconfig
eth1      Link encap:Ethernet  HWaddr 00:14:CB:57:45:07
          inet addr:192.168.44.17  Bcast:192.168.44.255  Mask:255.255.255.0
          inet6 addr: fe80::214:cbff:fe57:4507/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:68499 errors:0 dropped:0 overruns:0 frame:0
          TX packets:65523 errors:0 dropped:0 overruns:0 carrier:2
          collisions:0 txqueuelen:1000
          RX bytes:254042003 (242.2 MiB)  TX bytes:213023528 (203.1 MiB)
          Interrupt:193 Base address:0x4000 Memory:dfcfff00-dfcfffff

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:5205 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5205 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:761133 (743.2 KiB)  TX bytes:761133 (743.2 KiB)

[22:50:17(root@olivo)/home/gfr]> route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.44.0   0.0.0.0        255.255.255.0   U     0      0      0 eth1
0.0.0.0        192.168.44.1   0.0.0.0         UG    0      0      0 eth1
```

Figura 5. Información lógica del interfaz de red ya configurado.

la herramienta ping para "explorar nuestro entorno": haremos ping a una máquina que esperamos encontrar, por ejemplo al dispositivo utilizado como router por defecto (puerta de enlace predeterminada), y si este nos contesta es que "está vivo" y nosotros estamos conectados a su red.

- #: Las líneas que comienzan con este símbolo son comentarios que aportarán información a los programadores del script pero no serán interpretados por la shell.

### Todo junto

En el listado 2 podemos ver el script construido para lograr el objetivo planteado en este artículo.

El script intenta primero asociarse a redes de acceso inalámbrico y va comprobando si ha sido capaz de hacerlo; finalmente realiza lo mismo para redes cableadas tradicionales.

Con todo lo visto en los apartados anteriores creemos que el lector de este artículo no tendrá problemas para interpretar y modificar el script a su gusto, adaptándolo a sus diferentes localizaciones.

Ahora sólo queda situarlo en el directorio correspondiente para que sea ejecutado automáticamente al iniciar nuestro ordenador. En Debian GNU/Linux el directorio que alberga todos los archivos que se pueden ejecutar en el arranque es el directorio "/etc/init.d". Sin embargo, el sistema lanzará unos u otros según el nivel de arranque elegido (para mayor información sobre esto consultad las páginas de manual "man init" y "man inittab"). El nivel de arranque normal de un sistema Debian es el 2; por tanto, se ejecutarán todos los enlaces simbólicos que existan en el directorio "/etc/rc2.d". Así, tendremos que realizar dos acciones:

1. Copiar nuestro script, como usuario root, al directorio "/etc/init.d", asignarle los permisos adecuados con el comando:

```
chmod 700 nombre_script
```

Conseguimos con el comando anterior asignar permisos de lectura, escritura y ejecución al usuario root y ningún permiso a otros. Esto es conveniente ya que el script posee información privada (es aconsejable que no sea conocida) como claves de cifrado, nombre de redes WiFi, etc.

2. Crear un enlace simbólico a este archivo en el directorio correspondiente al nivel de arranque 2 (/etc/rc2.d). Para ello utilizamos el siguiente comando (suponiendo que el nombre de nuestro script es "red\_up"):

```
update-rc.d red_up start 15 2 .
```

Y obtendremos la siguiente salida por pantalla que confirma que el trabajo ha sido realizado:

```
Adding system startup for
/etc/init.d/red_up ...
/etc/rc2.d/S15red_up -> ../init.d/red_up
```

A partir de ahora, una de las primeras cosas que hará nuestra máquina al arrancar será ejecutar el script "red\_up", e intentar así configurar sus interfaces de red con los parámetros adecuados según donde nos encontremos: ¡Objetivo conseguido!

### ¿Y si no estamos conectados?

Al igual que en un apartado anterior hemos presentado un proxy caché del servicio DNS (pdnsd) también disponemos de herramientas que nos hacen de proxy caché del servicio web. Entre estos hemos seleccionado uno al que podemos indicar si estamos conectados (on-line) o desconectados (off-line) a Internet. Su nombre es wwwwoffle.

Para instalarlo en Debian GNU/Linux basta con ejecutar el siguiente comando como root:

```
apt-get install wwwwoffle
```

Tras ser instalado se nos preguntará sobre los parámetros básicos de configuración. Bastará con ir aceptando las opciones que nos va proponiendo por defecto, excepto cuando nos pregunta: "Does wwwwoffle rely on a dialup (PPP) interface for its connection?" a lo que responderemos "no" si estamos conectados por LAN o ADSL, y "sí" si lo hacemos por módem tradicional a través de la Red Telefónica Básica (RTB). Termina este diálogo solicitándonos una password para tener acceso a la administración de esta herramienta que, supuestamente, grabará en el fichero de configuración "/etc/wwwoffle/wwwoffle.conf". En las versiones que hemos utilizado para escribir el artículo este paso falla. Podéis comprobarlo si, como usuario root, miráis el fichero anterior y buscáis la palabra "password": veréis que está igualada a un valor. En nuestro caso siempre es "secret", independientemente de lo que hayamos tecleado anteriormente. Es por ello por lo que os recomendamos que la cambiéis editando dicho fichero. Tras realizar el cambio, hemos de volver a arrancar el servidor proxy caché para que surja efecto. Esto se hace con



el siguiente comando, ejecutado como root, en un terminal:

```
/etc/init.d/wwwoffle restart
```

Para afinar su configuración disponemos de una página web donde iremos modificando aquellos parámetros que nos interesen. Ver una configuración al detalle de esta herramienta, que muestre todas sus posibilidades podría ser, quizá, objeto de un próximo artículo. Por los límites existentes en la extensión de los artículos, aquí sólo vamos a indicar cómo se cambia el estado de esta herramienta a on-line u off-line según nos interese en cada momento (el resto de parámetros configurados por defecto son suficientes para su utilización normal). Para ello nos dirigimos a la página web de configuración: <http://localhost:8080/Welcome.html>.

Una vez que nos sea presentada la página inicial, de los enlaces que hay en la cabecera seleccionamos "Control del programa", y en la nueva página a la que accedemos seleccionamos "control". Pulsando sobre él iremos a una página web. (Nos solicitará usuario y password; el usuario lo dejamos en blanco y para la password utilizamos la que elegimos en el momento de la instalación) que nos presenta una serie de botones:

- Conectado (Online). Nos permite poner el programa on-line. Es decir, le indicamos que estamos conectados a Internet y que puede ir recogiendo páginas, si es necesario porque no las tiene ya almacenadas o porque ha habido cambios en ellas, y guardándolas en su caché.
- Desconectado (Offline). Nos permite indicarle que no hay conexión a Internet, y que ha de servir las páginas que solicite nuestro navegador desde su caché si las tiene. Si no las tiene, guarda una referencia a ellas para descargarlas en el momento en que tengamos conexión, asegurándonos su presencia en caché la siguiente vez que la vayamos a consultar. Obtendremos el siguiente mensaje indicativo: "Your request for URL <http://www.pagina.solicitada.com> has been recorded for download".
- Status. Nos ofrece información sobre el estado de nuestro proxy caché. Para utilizar este servidor proxy caché hemos de configurar en nuestro navegador la sección correspondiente. Por ejemplo, en firefox hemos de elegir en el menú "Editar" la opción "Preferencias"; se abrirá una nueva ventana y elegimos la pestaña/icono "General" y en ella "Configuración de conexión". Marcamos la opción "Configuración manual del proxy", y en la casilla "Proxy http" ponemos 127.0.0.1 y en "Puerto" 8080.

A partir de ese momento nuestro navegador (firefox) utilizará nuestro proxy caché (wwwoffle). Será wwwoffle quien servirá las páginas desde su almacén en nuestro disco duro siempre que las tenga (con lo que ahorramos ancho de banda), y sólo las descargará de Internet cuando sea necesario.

```
[17:38:46(root@olivo)/home/gfr]# ping 192.168.44.1
PING 192.168.44.1 (192.168.44.1) 56(84) bytes of data.
64 bytes from 192.168.44.1: icmp_seq=1 ttl=64 time=0.036 ms
64 bytes from 192.168.44.1: icmp_seq=2 ttl=64 time=0.035 ms
64 bytes from 192.168.44.1: icmp_seq=3 ttl=64 time=0.036 ms

--- 192.168.4.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.035/0.035/0.036/0.006 ms
```

## Herramientas gráficas de monitorización

Podemos monitorizar el funcionamiento de nuestro interfaz inalámbrico haciendo uso de una de las siguientes herramientas:

- KDE applet - información de red inalámbrica
- gnome-netstatus-applet
- gkrellmwireless
- kismet
- wmwave

Los parámetros habitualmente mostrados son: el nombre de la red WiFi, la calidad del enlace, el nivel de señal, la relación señal ruido y la velocidad a la que estamos operando.

Es aconsejable seleccionar una de estas herramientas según nuestro gusto y mantenerla abierta (algunas de ellas se empotran en nuestra bandeja del sistema) para detectar con un simple vistazo si hay problemas en la conexión de red inalámbrica.

## Conclusión

En este artículo hemos aprendido a instalar el driver apropiado para nuestra tarjeta de red inalámbrica, a utilizar una serie de herramientas de configuración/monitorización de red. Hemos visto cómo funciona cada una de ellas de forma independiente al resto, y cómo se interrelacionan todas ellas en un script que nos permite ajustar los parámetros de red de nuestra máquina según la localización donde nos encontremos.

Además, hemos visto una serie de herramientas que nos ofrecen prestaciones adicionales a las que íbamos buscando.

Así, la herramienta pdnsd hace de servidor proxy para la resolución de nombres, que además almacena en un archivo (caché) en el disco duro, disminuyendo tiempos de resolución y consumo de ancho de banda.

También hemos presentado la herramienta wwwoffle que es un proxy caché del servicio web, y que nos permite navegar por su caché de páginas web cuando no tenemos conexión a Internet (estado off-line o desconectado). ↵

## Sobre el autor

Godofredo Fernández Requena puede ser localizado en <http://godev.vivencias.net>, donde estará encantado de compartir con los lectores sus impresiones, recibir sus críticas (constructivas a ser posible) y ayudarles en aquellos temas que no hayan quedado suficientemente claros (quizá se pueda palpar así el interés de los lectores sobre artículos de esta temática para futuras publicaciones).

Figura 6. Respuesta del punto de acceso a la prueba de ping.