Detección de problemas. Monitorización

Autores:

- Pablo Boronat Pérez (Universitat Jaume I)
- Miguel Pérez Francisco (Universitat Jaume I)
- David Rubert Viana (Universitat Jaume I)

Introducción

La red libre evoluciona continuamente, cada día que pasa nacen nuevos nodos o supernodos, y ante un sistema tan dinámico como éste, disponer de herramientas para poder encontrar y diagnosticar posibles problemas es imprescindible.

La detección y resolución de problemas es una labor compleja, y que se aprende únicamente en base a la experiencia. Veremos en este tema cómo utilizar diferentes herramientas de diagnóstico de red para detectar problemas, o incluso preverlos antes de que sucedan.

Logs. Linux, AirOS i RouterOS

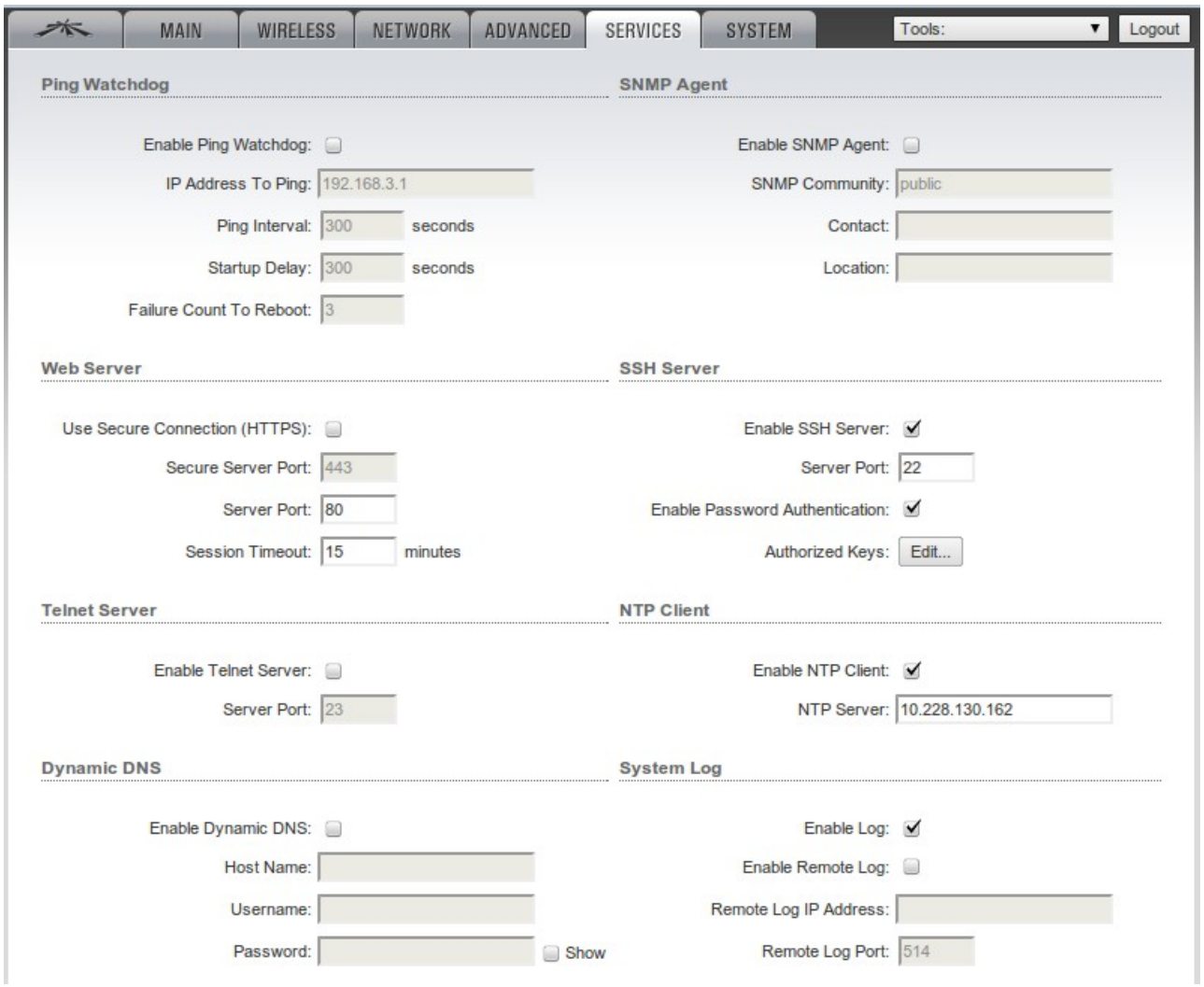
Los archivos de log de nuestro sistema nos proporcionan información histórica y en tiempo real sobre lo que está pasando. Es muy importante revisarlos en el momento en que sospechemos que algo no funciona bien, ya que cualquier cosa que pase en el sistema se debería registrar allí.

Veamos cómo activar y procesar los logs en función del sistema en el que estemos trabajando.

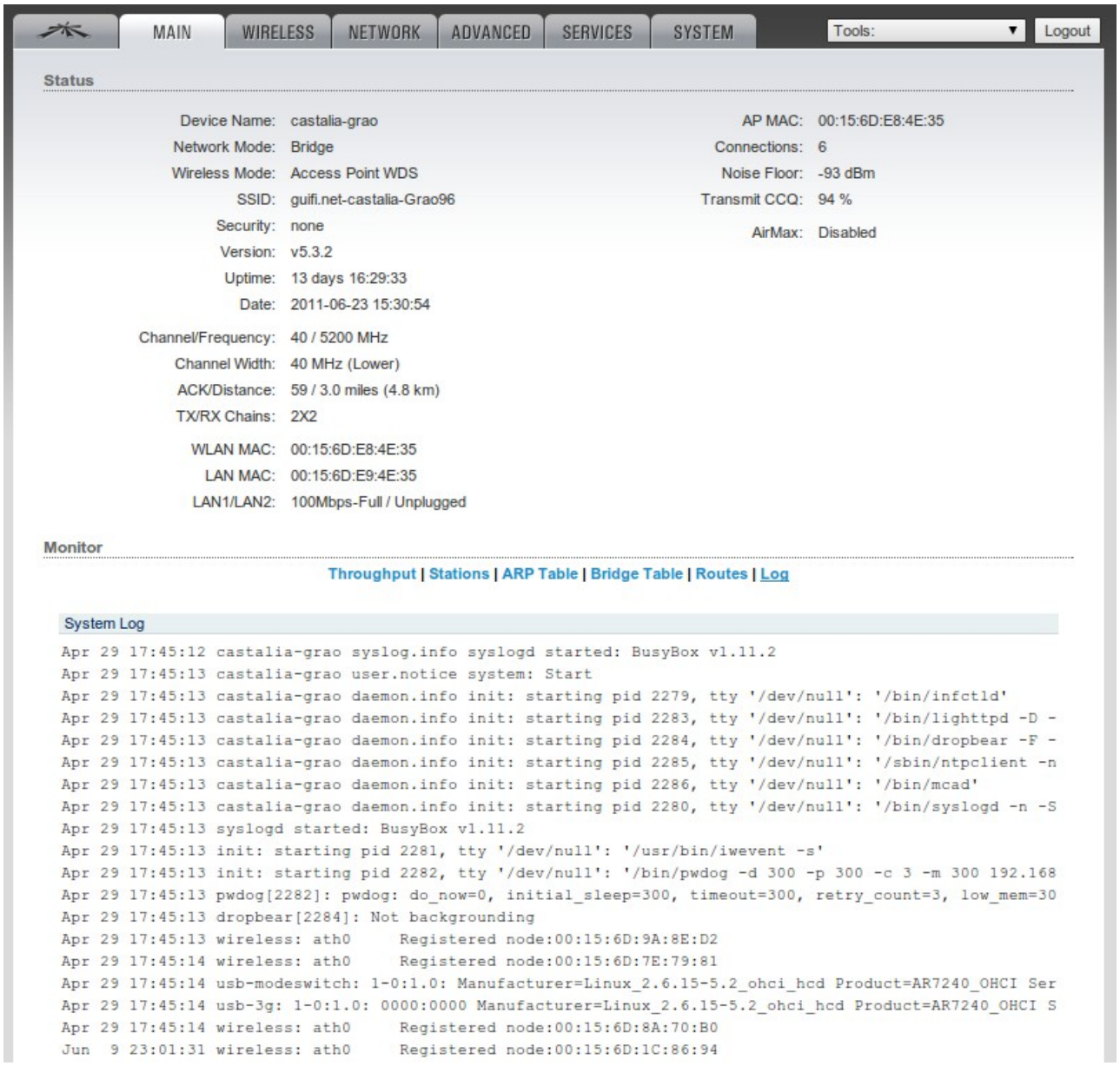
AirOS

AirOS es un sistema operativo que no engloba toda la complejidad de otros sistemas como puede ser Linux o RouterOS, por lo que los logs se activan y procesan de una manera sencilla, dándonos a su vez una información limitada al buen funcionamiento de la antena.

Activación de los logs



Procesado de los logs



RouterOS

En RouterOS se pueden consultar los logs del sistema en el apartado `/log`

```
[admin@CS-UJI-BiblAP] > /log print
08:10:07 wireless,info wlan4: data from unknown device
00:0C:42:61:86:69, sent deauth
08:10:07 wireless,info wlan4: data from unknown device
00:0C:42:61:86:69, sent deauth
08:10:07 wireless,info wlan4: data from unknown device
00:0C:42:61:86:69, sent deauth
08:10:07 wireless,info wlan4: data from unknown device
00:0C:42:61:86:69, sent deauth
08:10:07 wireless,info wlan4: data from unknown device
00:0C:42:61:86:69, sent deauth
08:10:07 wireless,info wlan4: data from unknown device
00:0C:42:61:86:69, sent deauth
```

```
08:10:07 wireless,info wlan4: data from unknown device
00:0C:42:61:86:69, sent deauth
08:10:07 wireless,info wlan4: data from unknown device
00:0C:42:61:86:69, sent deauth
08:10:07 wireless,info wlan4: data from unknown device
00:0C:42:61:86:69, sent deauth
08:10:07 wireless,info 00:0C:42:61:86:69@wlan4: connected, is AP,
wants WDS
08:10:07 route,ospf,info Discarding packet: no neighbor with this
source address
08:10:07 route,ospf,info     RouterId=10.228.133.97
08:10:07 route,ospf,info     source=172.16.107.37
08:10:10 route,ospf,info Discarding packet: no neighbor with this
source address
08:10:10 route,ospf,info     RouterId=10.228.133.97
08:10:10 route,ospf,info     source=172.16.107.37
08:10:11 route,ospf,info Discarding packet: no neighbor with this
source address
08:10:11 route,ospf,info     RouterId=10.228.133.97
08:10:11 route,ospf,info     source=172.16.107.37
08:13:27 wireless,info 00:15:6D:D2:4F:33@wlan2: reassociating
08:13:27 wireless,info 00:15:6D:D2:4F:33@wlan2: disconnected, ok
08:13:27 wireless,info 00:15:6D:D2:4F:33@wlan2: connected
08:16:14 wireless,info 00:15:6D:D2:4F:33@wlan2: reassociating
08:16:14 wireless,info 00:15:6D:D2:4F:33@wlan2: disconnected, ok
...
```

Se pueden consultar las acciones que se están registrando en el apartado /system logging

```
[admin@CS-UJI-BiblAP] /system logging> /system logging print
Flags: X - disabled, I - invalid
 #   TOPICS                       ACTION
PREFIX
 0   info                         memory

 1   error                        memory

 2   warning                      memory

 3   critical                     echo
```

Para añadir que se registren nuevas acciones se puede ejecutar la orden

```
[admin@CS-UJI-BiblAP] /system logging> /system logging add
action=memory topics=ospf
[admin@CS-UJI-BiblAP] /system logging> /system logging print
Flags: X - disabled, I - invalid
 #   TOPICS                       ACTION
PREFIX
 0   info                         memory

 1   error                        memory
```

```
 2   warning                     memory

 3   critical                    echo

 4   ospf                        memory
```

y para eliminarlas

```
/system logging remove 4
```

Linux

Como se ha comentado en el tema 5 GNU/Linux dispone de un potente sistema de logs que almacena los mensajes generados por las aplicaciones. Con cada mensaje se almacena qué programa lo generó, la prioridad y la fecha y hora en que se produjo.

Los ficheros de log en un sistema linux, se encuentran habitualmente en el directorio `/var/log` o en algún directorio dentro de éste.

El sistema de logs arranca con el script `/etc/init.d/sysklogd`, el duende que realiza los registros es `syslogd` que se configura mediante el fichero `/etc/syslog.conf`, donde se indica qué se quiere registrar y dónde se deben enviar los logs.

Los archivos más importantes son:

- `/var/log/messages`: donde se almacenan todos los mensajes con prioridad `info` (información), `notice` (notificación) o `warn` (aviso). Es uno de los ficheros en los que primero se mira cuando hay algún problema.
- `/var/log/kern.log`: almacena los logs del kernel.
- `/var/log/dmesg`: almacena la información que genera el kernel durante el arranque del sistema. Se puede ver su contenido con la orden `dmesg`.

Para ver el contenido (total o parcial) de alguno de estos ficheros se pueden utilizar alguna de las siguientes ordenes:

```
cat /var/log/messages
```

muestra el contenido completo del fichero `/var/log/messages`.

```
less /var/log/messages
```

muestra el contenido completo del fichero `/var/log/messages` paginando la salida (la tecla q permite finalizar sin mostrar todo el fichero).

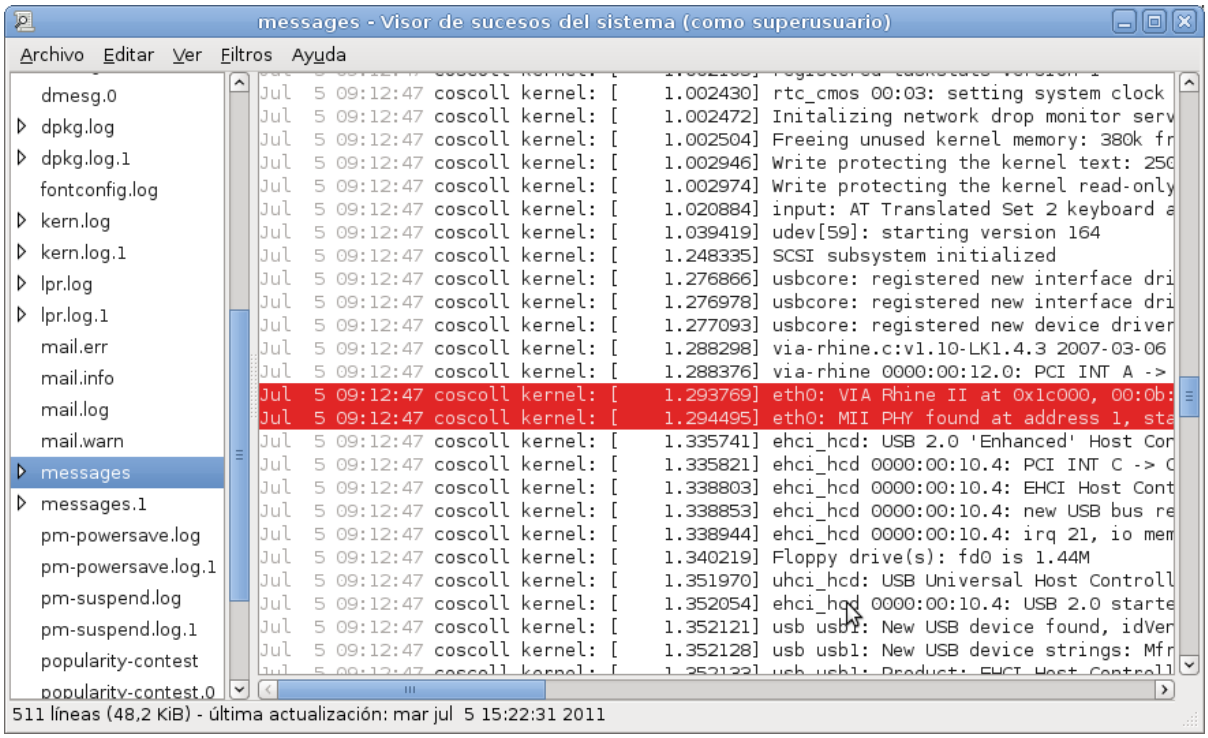
```
tail -50 /var/log/messages
```

muestra las últimas 50 líneas del fichero `/var/log/messages`.

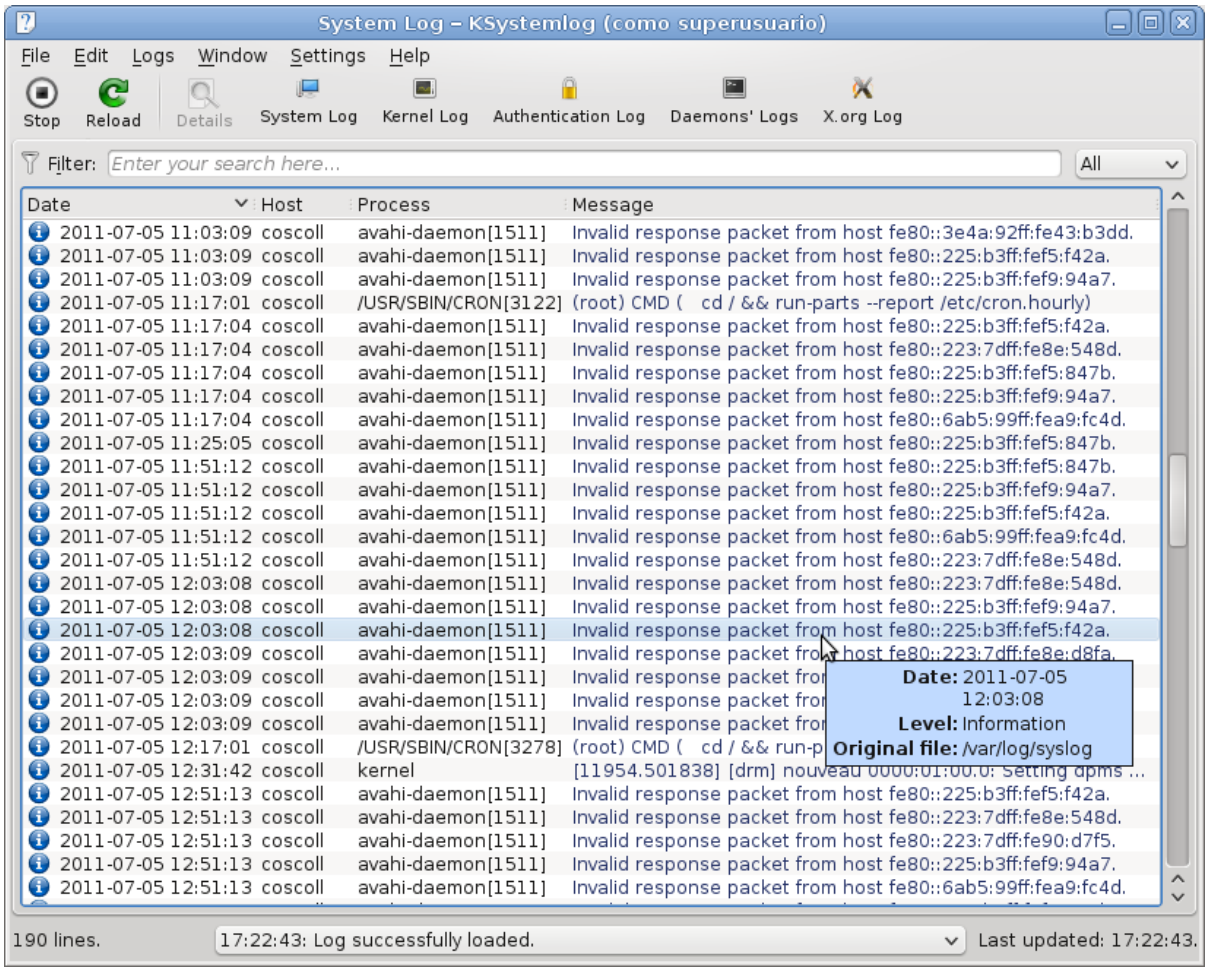
Los archivos de log suelen crecer mucho ya que en ellos se está guardando información continuamente. Por ello, existe una aplicación, `logrotate` (que se puede configurar a través del fichero en `/etc/logrotate`) que, si los ficheros de log son muy grandes, los comprime y aplica una rotación a los archivos (añadiéndoles la extensión .1.gz, .2.gz, etc.), volviendo a crear uno vacío (cuanto mayor es el número más antiguo es el log).

Existen aplicaciones gráficas para supervisar los logs, por ejemplo GNOME-System-Log, KSystemLog, Xlogmaster y Xwatch.

GNOME-System-Log



KSystemLog



Enlaces

http://www.aboutdebian.com/syslog.htm

TCPDump

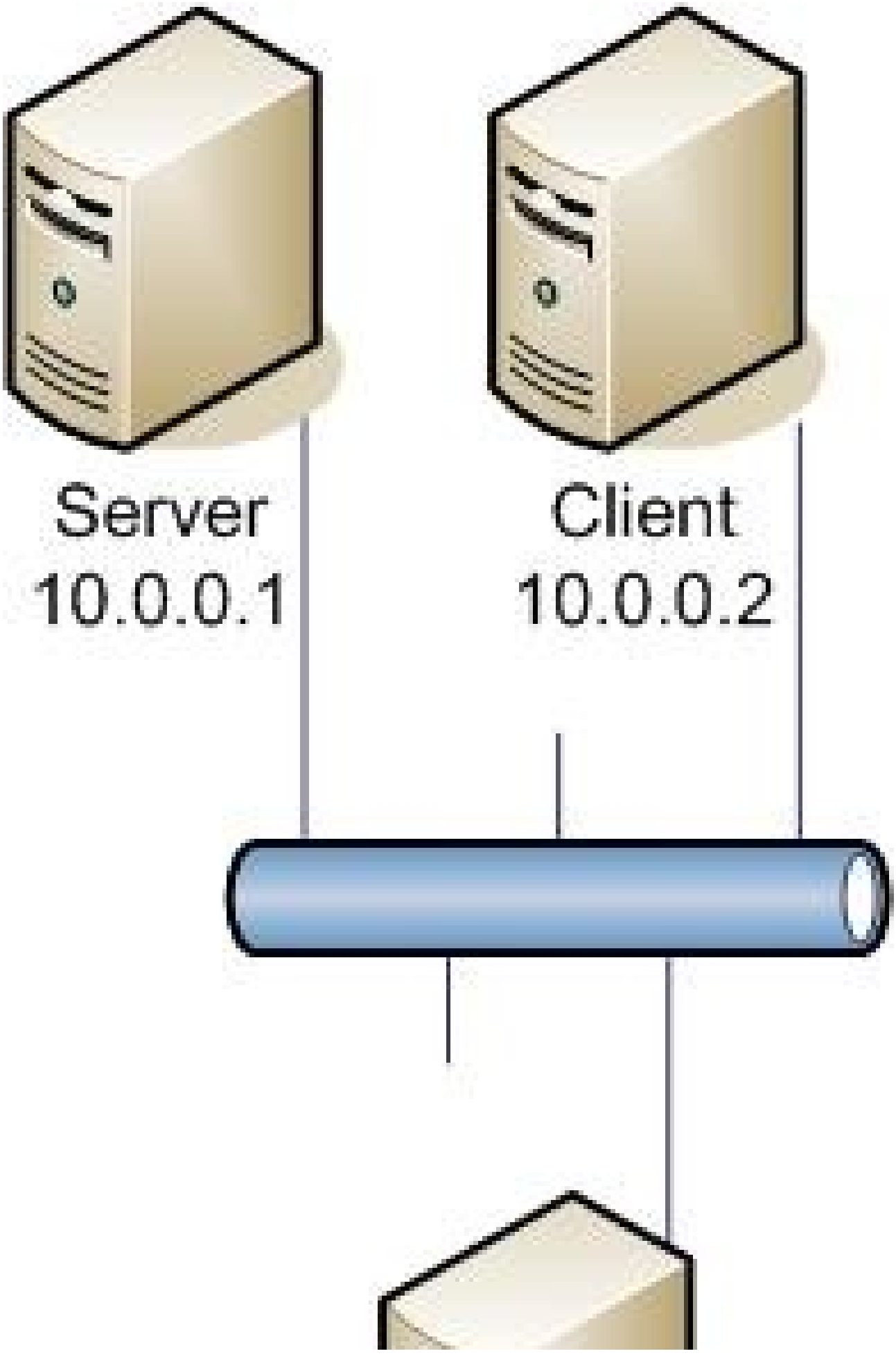
- http://es.wikipedia.org/wiki/Tcpdump

TCPDump es una utilidad de análisis de tráfico de red en línea de comandos. Pese a su complejidad inicial, una vez aprendido su funcionamiento básico será la herramienta que nos resultará más útil para saber qué está pasando en la red. Nos va a permitir examinar en bruto el tráfico que está pasando por la red, así como filtrar por protocolo, IP, MAC, puerto, ect.

Mikrotik tiene también un *sniffer* de red, pero no es tan completo y cómodo como **tcpdump**.

Consideraciones iniciales

- TCPDump únicamente funciona cuando lo ejecutamos en modo **root**. Esto es así por motivos de seguridad.
- El concepto de TCPDump es lo que se muestra en esta imagen, poner en una red un cliente que “observa” el tráfico que se intercambia entre el resto de clientes.
- En las redes switcheadas actuales podemos observar el tráfico de los clientes directamente conectados al switch. En las redes inalámbricas podemos observar el tráfico que fluye por todo el enlace inalámbrico.
- En nuestras máquinas enrutadoras cobra sentido ya que hay mucho tráfico.
- Utilizarlo nos hará ser consciente de lo importante que es cuidar la seguridad en nuestras comunicaciones.


Server
10.0.0.1
Client
10.0.0.2

Manual y ejemplos

```
# man tcpdump
```

La mejor manera de entender tcpdump es ponerlo en práctica

Capturar tráfico cuya dirección IP de origen sea 10.228.130.1.

```
# tcpdump -i wlan0 src host 10.228.130.1
```

```
root@lateralus:~# tcpdump -n -i wlan0 src host 10.228.130.1
tcpdump: verbose output suppressed, use -v or -vv for full
protocol decode
listening on wlan0, link-type EN10MB (Ethernet), capture size
65535 bytes
14:56:06.291867 IP 10.228.130.1.80 > 192.168.1.222.37271: Flags
[S.], seq 2510715723, ack 98586484, win 5792, options [mss
1460,sackOK,TS val 119619640 ecr 146592,nop,wscale 2], length 0
14:56:06.294414 IP 10.228.130.1.80 > 192.168.1.222.37271: Flags
[.], ack 376, win 1716, options [nop,nop,TS val 119619640 ecr
146592], length 0
14:56:06.299367 IP 10.228.130.1.80 > 192.168.1.222.37271: Flags
[.], seq 1:1449, ack 376, win 1716, options [nop,nop,TS val
119619640 ecr 146592], length 1448
14:56:06.302374 IP 10.228.130.1.80 > 192.168.1.222.37271: Flags
[.], seq 1449:2897, ack 376, win 1716, options [nop,nop,TS val
119619640 ecr 146592], length 1448
```

Capturar tráfico con destino a la dirección MAC 50:43:A5:AE:69:55

```
# tcpdump -i eth0 ether dst 50:43:A5:AE:69:55
```

Capturar tráfico con red destino 10.228.0.0

```
# tcpdump -i wlan0 dst net 10.228.0.0/16
```

Capturar tráfico con puerto origen o destino el 53 (DNS)

```
# tcpdump -i wlan0 port 53
```

Capturar los paquetes de tipo ICMP

```
# tcpdump -i eth1 ip proto \\icmp
```

Capturar los paquetes de tipo UDP

```
# tcpdump -i tunnel0 ip proto \\udp
```

Capturar todo el tráfico TCP excepto el de SSH

```
# tcpdump -i tunnel0 ip proto \\tcp and not port 22
```

Capturar tráfico para examinarlo posteriormente con wireshark:

```
# tcpdump -i <interface> -s 65535 -w <some-file>
```

Mikrotik. Packet sniffer

Muy similar a la funcionalidad de tcpdump, aunque un tanto incómodo y menos potente. Nos permite analizar paquetes que pasan por el router y filtrar por interfaz, ip origen/destino o protocolo.

```
#Para acceder:
/tool sniffer

# arrancarlo
/tool sniffer start

# pararlo
/tool sniffer stop

# Examinar el tráfico
/tool sniffer packet print
#    TIME INTERFACE SRC-ADDRESS
DST-ADDRESS                                                  IP-
PROTOCOL  SIZE
0   0.001 bridge1   10.228.144.161:22 (ssh)
10.228.144.170:48615                                         tcp
180
1   0.001 ether3    10.228.144.161:22 (ssh)
10.228.144.170:48615                                         tcp
180
2   0.027 ether3    10.228.144.163:3128 (squid)
10.228.145.58:50946                                          tcp
46
3   0.027 bridge1   10.228.144.163:3128 (squid)
10.228.145.58:50946                                          tcp
46
4   0.027 wlan2     10.228.144.163:3128 (squid)
10.228.145.58:50946                                          tcp
40
...
17   0.043 ether3    10.228.144.170:48615
10.228.144.161:22 (ssh)                                      tcp
52
18   0.043 bridge1   10.228.144.170:48615
10.228.144.161:22 (ssh)                                      tcp
52
19   0.077 wlan2     10.228.145.58:50994
10.228.144.163:3128 (squid)                                  tcp
46
20   0.077 bridge1   10.228.145.58:50994
10.228.144.163:3128 (squid)                                  tcp
40
21   0.077 ether3    10.228.145.58:50994
10.228.144.163:3128 (squid)                                  tcp
40
22   0.249 wlan2     10.228.145.58:50946
10.228.144.163:3128 (squid)                                  tcp
46
23   0.249 bridge1   10.228.145.58:50946
10.228.144.163:3128 (squid)                                  tcp
40
```

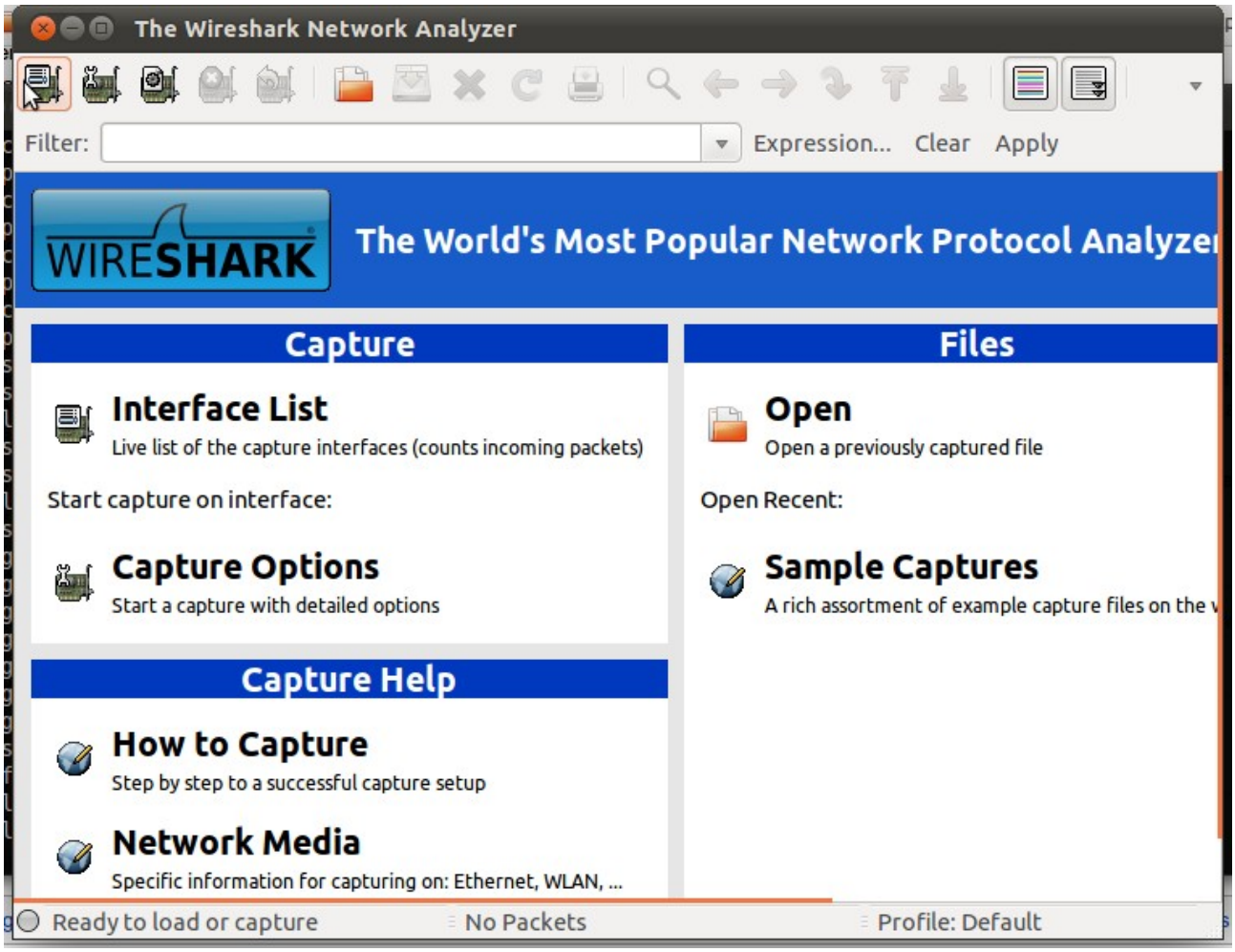
La documentación de esta herramienta la podemos ver aquí:

http://wiki.mikrotik.com/wiki/Manual:Tools/Packet_Sniffer

Wireshark

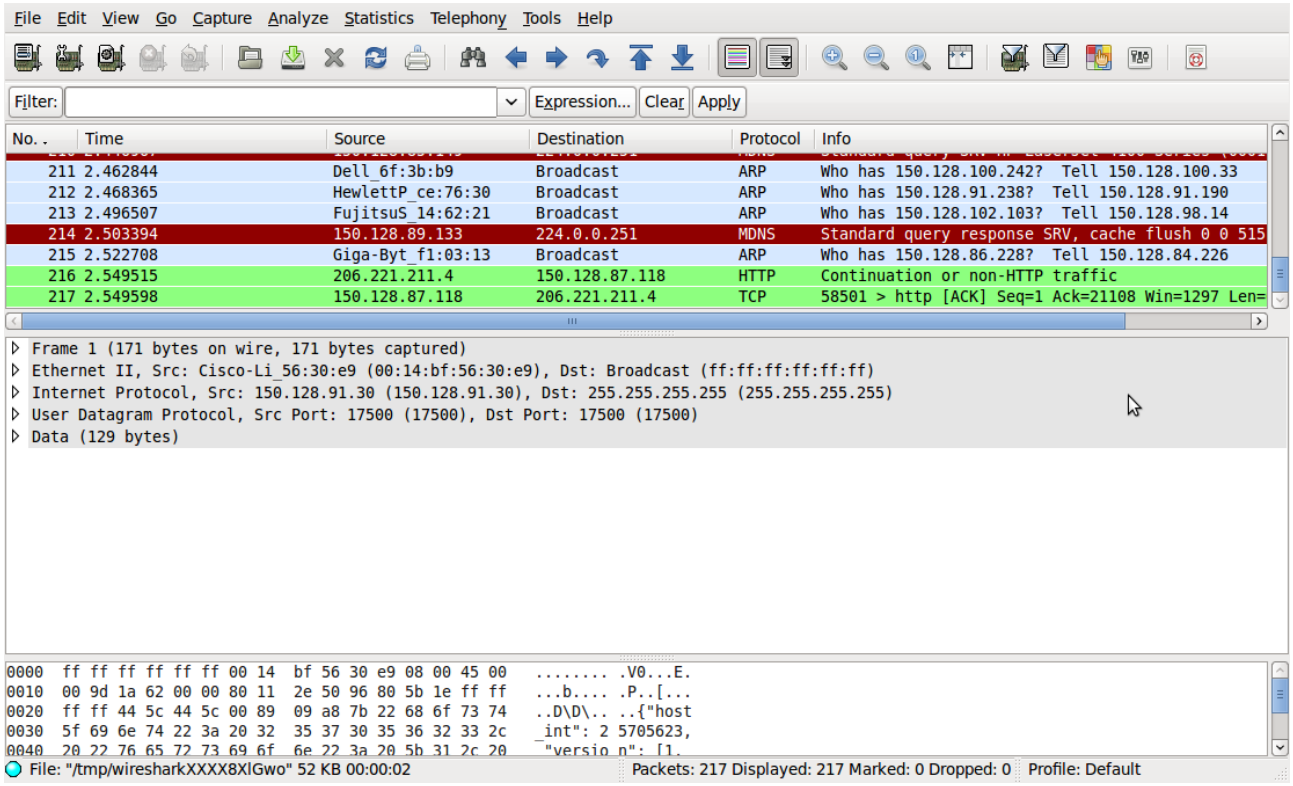
- http://es.wikipedia.org/wiki/Wireshark

Wireshark es, al igual que TCPDump, una herramienta de análisis del tráfico de red vía interfaz gráfica. Nos permite *escuchar* lo que está pasando en la red en un momento determinado, para posteriormente analizarlo gracias a funcionalidades como el agrupamiento, organización y filtrado de información.

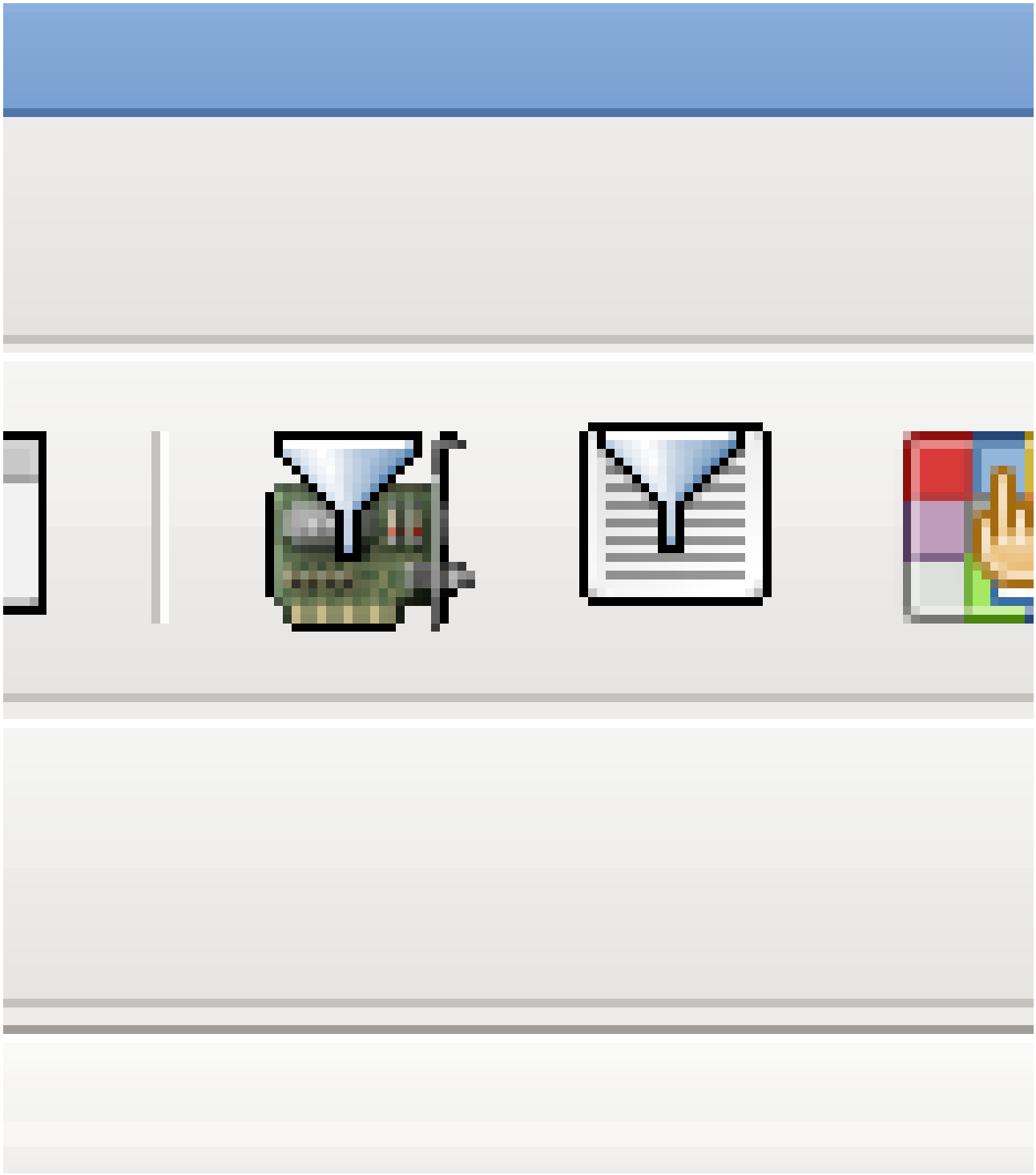


Debemos ejecutar Wireshark como administrador ya que se necesita acceso a los dispositivos de red. La captura realizada se puede guardar para analizarla posteriormente.

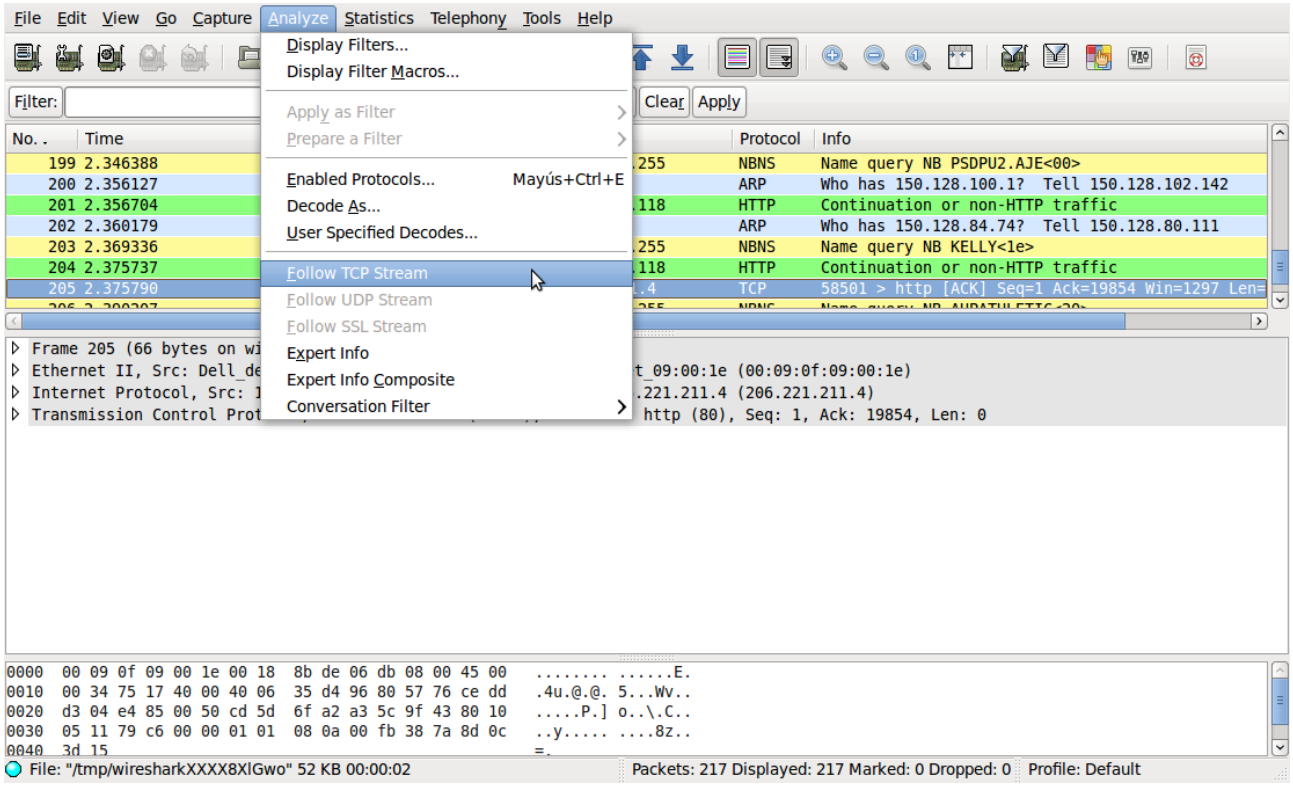
La ventana de la aplicación tiene tres partes. La primera nos muestra los mensajes capturados, la segunda, para el mensaje que seleccionemos, nos permite desplegar información sobre cada nivel de la pila de protocolos; esta parte es muy interesante porque podemos ver datos de la cabera de cada protocolo. La tercera nos muestra en hexadecimal todos los bits del mensaje.



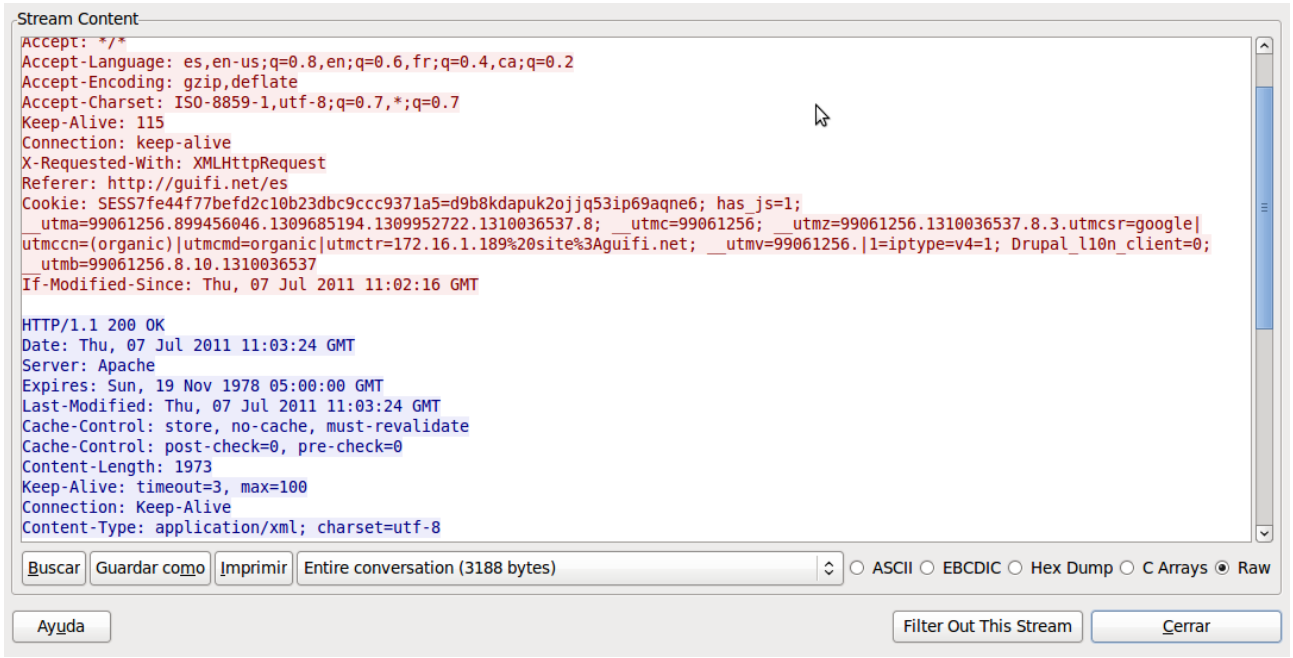
Sobre una captura se pueden aplicar filtros para focalizarnos en algún tipo de tráfico. Hay filtros de captura (para limitar el tamaño de los datos recogidos) y filtros que se aplican sobre una captura ya realizada (si quitamos el filtro volveremos a ver todos los mensajes de la captura).



Una utilidad interesante de Wireshark es *Follow tcp stream*. Con esta utilidad, si seleccionamos un mensaje de una conexión TCP, veremos los mensajes que se intercambian en los dos sentidos de esa conexión. El diálogo se ve con un color para interlocutor.



El resultado:



ipcalc

Aunque no se trata de una herramienta de monitorización, es muy útil para averiguar el rango de IPs de una subred.

```
ipcalc 10.228.130.3/27
```

devuelve

```
Address:   10.228.130.3          00001010.11100100.10000010.000
00011
Netmask:   255.255.255.224 = 27  11111111.11111111.11111111.111
```

```
00000
Wildcard:  0.0.0.31             00000000.00000000.00000000.000
11111
=>
Network:   10.228.130.0/27      00001010.11100100.10000010.000
00000
HostMin:   10.228.130.1         00001010.11100100.10000010.000
00001
HostMax:   10.228.130.30        00001010.11100100.10000010.000
11110
Broadcast: 10.228.130.31        00001010.11100100.10000010.000
11111
Hosts/Net: 30                    Class A, Private Internet
```

Netdiscover

Netdiscover busca ordenadores en una red de forma pasiva (por el tráfico que ve pasar) o mediante peticiones arp (activamente consultado con el protocolo ARP). También puede ser útil para inspeccionar el tráfico arp de una red o para encontrar direcciones de red (en el modo auto scan).

`netdiscover -i eth0`

busca ordenadores en la interfaz `eth0`, realiza un scan de las subredes privadas definidas en el RFC 1918 (`192.168.0.0/16`, `172.16.0.0/16`, `172.26.0.0/16`, ... , `10.0.0.0/8`). Es decir, que si no indicamos el rango de IPs que debe buscar, irá probando activamente dentro de los rangos de IPs privadas.

`netdiscover -i ath0`

busca ordenadores en la interfaz `ath0`

`netdiscover -i eth0 -r 150.128.2.0/27`

busca ordenadores en la interfaz `eth0` dentro del rango `150.128.2.0/27`

`netdiscover -p`

busca ordenadores en modo pasivo, sólo «mira» la información que pasa por la red.

`netdiscover -P`

muestra la salida de forma que se pueda almacenar en un fichero.

Traceroute/ping/MTR

ping

- http://es.wikipedia.org/wiki/Traceroute
- http://es.wikipedia.org/wiki/Ping
- http://en.wikipedia.org/wiki/MTR_(Software)

Ping es una herramienta de diagnóstico que comprueba el estado de una conexión mandando y recibiendo paquetes ICMP. Útil para comprobar

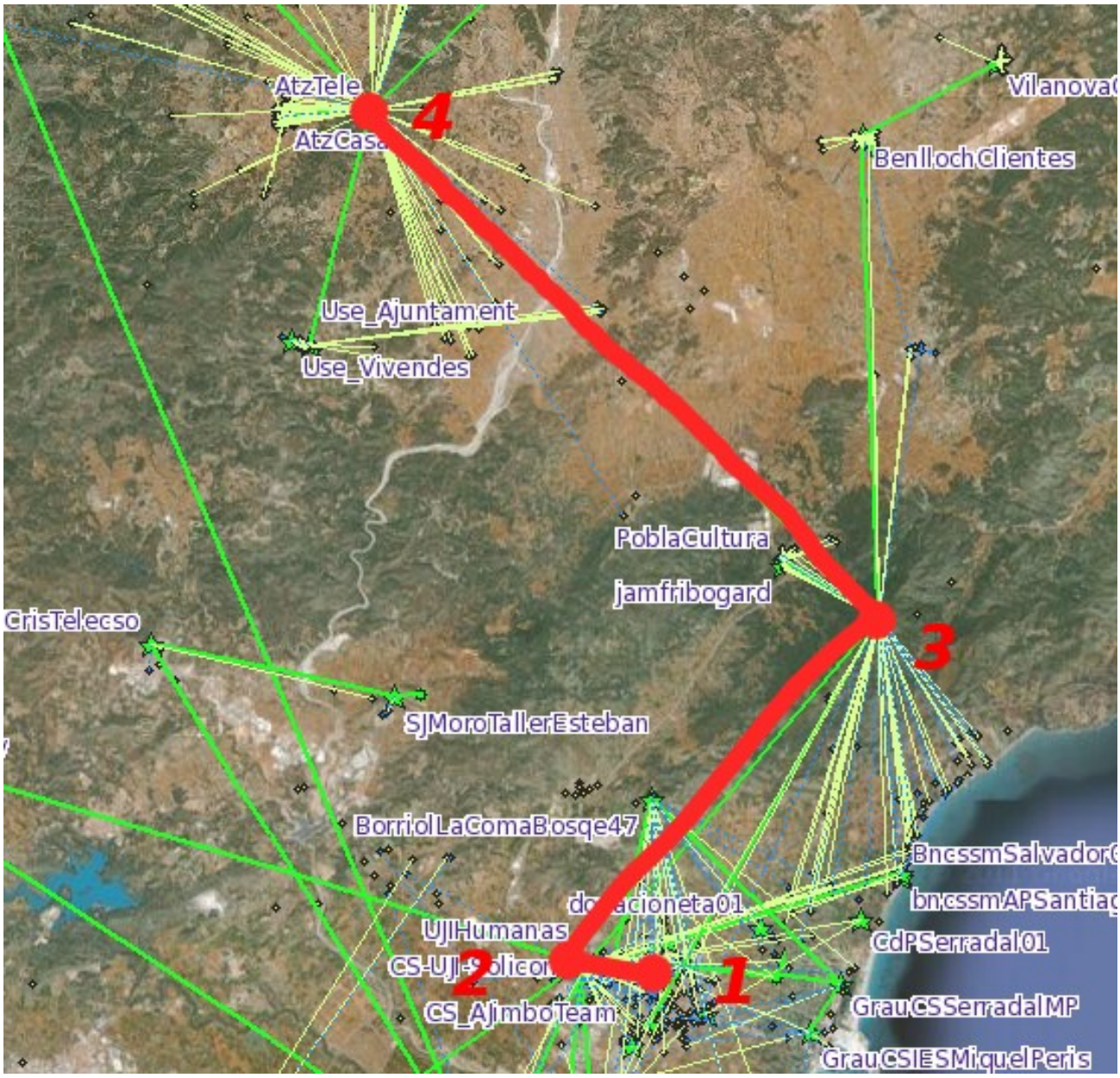
rápidamente el estado de la red, pero parco en datos.

Podemos utilizar un par de parámetros útiles (sólo para Linux):

* `-i time`: podemos especificar el tiempo entre cada envío de trama ICMP.
* `-t ttl`: Especificamos el número máximo de hosts por los que puede pasar el paquete.

Ejemplo:

```
$ ping -i .2 -t 10 10.228.145.1
```



En routers que tengan varias IPs asignadas puede ser necesario especificar la IP origen del ping porque si no sabemos la que se envía puede que no obtengamos contestación. Por ejemplo, si hacemos ping desde un punto donde tenemos una ip `172.x.y.z` y una `10.x.y.z` si el ping da más de un salto y sale con la IP origen `172.x.y.z` no recibiremos respuesta puesto que estas direcciones no se activan en el encaminamiento dinámico de guifi.net.

```
$ ping -I 10.228.130.1 10.228.132.33
```

MTR

Esta herramienta es la fusión de las dos anteriores, dándonos en un único comando lo mejor de **traceroute** y **ping**.

```
boots> mtr 10.228.131.1
HOST: boots                      Loss%   Snt   Last   Avg  Best
Wrst StDev
  1.|-- 10.228.144.189            0.0%    10    0.2   0.2   0.2
0.3   0.0
  2.|-- 172.16.1.78               0.0%    10    1.1   1.2   1.1
1.4   0.1
  3.|-- 172.16.1.189              0.0%    10    1.3   1.5   1.3
2.1   0.3
  4.|-- 172.16.107.69             0.0%    10    4.4   5.1   3.9
7.1   1.0
  5.|-- 10.228.131.1              0.0%    10    4.3   6.6   4.0
16.9   3.9
```

Al igual que con ping, si se quiere forzar la IP origen, se debe utilizar la opción `-a IP`.

Pruebas de prestaciones

Para realizar pruebas de velocidad de uno o varios enlaces se puede utilizar la utilidad `iperf`. Está disponible tanto para GNU/Linux como para AirOs. Por el momento no está disponible en RouterOS que tiene sus propios test de velocidad.

La forma más sencilla de ejecutarlo es la siguiente: En uno de los extremos se ejecuta `iperf` en modo servidor

```
iperf -s
```

en el otro extremo se ejecuta el test contra el servidor

```
iperf -c 10.228.134.212
```

donde `10.228.134.212` es la IP del servidor.

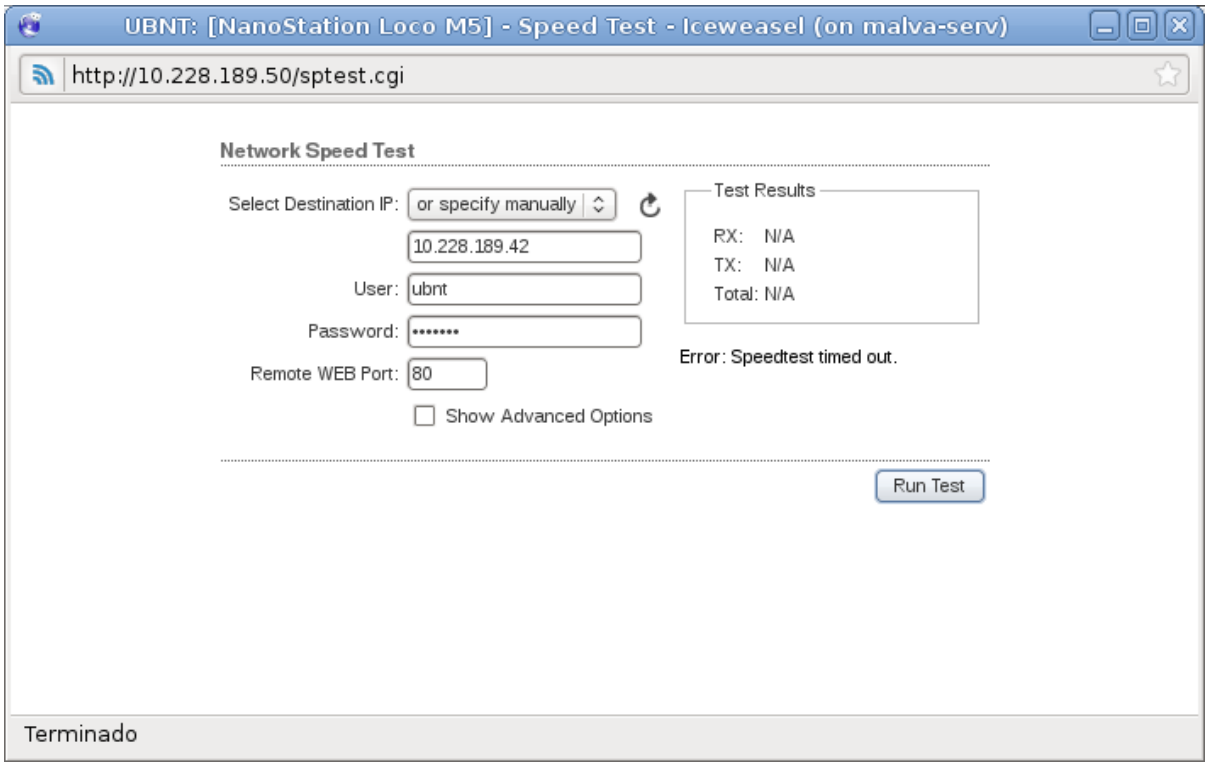
En castello.guifi.net por ejemplo, hay un servidor iperf funcionando continuamente. De esta forma se pueden realizar tests de velocidad fácilmente

```
XM.v5.3# iperf -c 10.228.130.162 -P 5
------------------------------------------------------------
Client connecting to 10.228.130.162, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  5] local 10.228.170.3 port 2798 connected with 10.228.130.162
port 5001
[  7] local 10.228.170.3 port 2799 connected with 10.228.130.162
port 5001
[ 10] local 10.228.170.3 port 2802 connected with 10.228.130.162
port 5001
[  8] local 10.228.170.3 port 2800 connected with 10.228.130.162
port 5001
[  9] local 10.228.170.3 port 2801 connected with 10.228.130.162
```

```
port 5001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.0 sec  8.82 MBytes  7.40 Mbits/sec
[ ID] Interval       Transfer     Bandwidth
[  7]  0.0-10.0 sec  8.97 MBytes  7.52 Mbits/sec
[ ID] Interval       Transfer     Bandwidth
[ 10]  0.0-10.0 sec  8.66 MBytes  7.26 Mbits/sec
[ ID] Interval       Transfer     Bandwidth
[  8]  0.0-10.0 sec  8.88 MBytes  7.44 Mbits/sec
[ ID] Interval       Transfer     Bandwidth
[  9]  0.0-10.0 sec  8.90 MBytes  7.46 Mbits/sec
[SUM]  0.0-10.0 sec  44.2 MBytes  37.1 Mbits/sec
```

con la opción `-P 5` indicamos que se realicen 5 pruebas de velocidad en paralelo de forma que el ancho de banda total es la suma de todos ellos.

Si se quiere realizar un test de velocidad entre dos dispositivos de ubiquity se puede utilizar el iperf tal y como se ha comentado anteriormente (entrando a los dispositivos por ssh), para ello deben tener IPs pertenecientes a la misma subred (o se debe poder llegar de una a la otra). La interfaz web del AirOs también dispone de una herramienta para medir la velocidad pero solo de otros dispositivos en la misma subred y se debe indicar el usuario y el password.



También cabe la posibilidad de hacer una descarga utilizando wget:

```
mperez@coscoll:~$ wget http://roure.act.uji.es/v

--2011-07-07 15:41:35--  http://roure.act.uji.es/v
Resolviendo roure.act.uji.es... 150.128.97.53
Connecting to roure.act.uji.es|150.128.97.53|:80... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 146276352 (140M) [text/plain]
```

```
Saving to: `v'

100%[======================================>] 146.276.352 11,1M/s
in 13s

2011-07-07 15:41:48 (11,1 MB/s) - `v' saved [146276352/146276352]
```

Descarga el fichero v almacenado en el servidor web de `roure.act.uji.es`. Muestra en pantalla la evolución de la descarga y al final nos muestra el tamaño del fichero y la velocidad a la que se ha descargado.

`2011-07-07 15:41:48 (11,1 MB/s) - `v' saved [146276352/146276352]`

También la orden scp se puede utilizar para averiguar la velocidad entre dos ordenadores (o nodos):

```
jlopez@coscoll:~$ scp jlopez@10.228.130.14:f.tgz .
f.tgz                                         100%  140MB
11.6MB/s   00:12
```

descarga el fichero `f.tgz`desde el ordenador `10.228.130.14` hasta el ordenador local (se solicitara la contraseña). Como se observa, muestra el tamaño del fichero, la velocidad de la descarga y el tiempo utilizado. Al tratarse de una copia segura (scp utiliza SSL) se añade una pequeña sobrecarga para el cifrado de los datos, aunque puede despreciarse.

Análogamente se puede realizar una transferencia del ordenador local a uno remoto mediante:

```
mperez@coscoll:~$ scp f.tgz jlopez@10.228.130.14:
f.tgz                                         100%  140MB
11.6MB/s   00:12
```

En RouterOS se puede utilizar `bandwidth-server/bandwidth-test` del apartado `tools`. Es similar al `iperf` pero sólo se puede utilizar para realizar pruebas entre dos mikrotiks (http://wiki.mikrotik.com/wiki/Manual:Tools/Bandwidth_Test).

En uno de los mikrotiks se pone en marcha el servidor

```
/tool bandwidth-server set enables=yes
```

y se prueba en el otro mediante

```
/tool bandwidth-test protocol=tcp address=10.228.145.1
```

Tracedump:

```
newBaseSize: 12pt
newBaseSizeInPt: 12
```