

Informe Laboratorio III Redes de Datos

Arturo Mantinetti
Manuel Tobar
Diego Vilches
Nicolas Henriquez
`arturo.mantinetti@mail_udp.cl`
`manuel.tobar@mail_udp.cl`
`diego.vilches@mail_udp.cl`
`nicolas.henriquez@mail_udp.cl`

Profesor
Jaime Álvarez
Ayudante
Maximiliano Vega

14 de Abril de 2016

Índice general

1. Introducción	2
2. Contenido	3
2.1. Creación de Paquetes	3
2.2. Hardware utilizado	4
2.3. Envío de un paquete de datos a FF:FF:FF:FF:FF:FF	5
2.3.1. Switch	6
2.3.2. Hub	6
2.4. Envío de un paquete de datos con MAC específica	7
2.4.1. Switch	7
2.4.2. Hub	8
2.5. Envío de un paquete de datos con una MAC fuera de la red	9
2.5.1. Switch	9
2.5.2. Hub	10
3. Conclusión	11

1. Introducción

Este laboratorio consistió en crear paquetes de datos con diferentes parámetros para luego enviarlos por la red, con el fin lo lograr comprender como se conforman y comportan estos según sus características. Esto es posible gracias a un programa llamado 'Scapy' que nos da esas funcionalidades.

Los paquetes, en este experimento, varían principalmente en la dirección MAC, lo que hace que sean recibidos por distintos equipos. Para esto se ocupa 'Wireshark', programa con el que se puede capturar los paquetes enviados por la red. Una vez creados y enviados los paquetes a través del Switch, se repite el procedimiento, sólo que esta vez los equipos están conectados a un Hub.

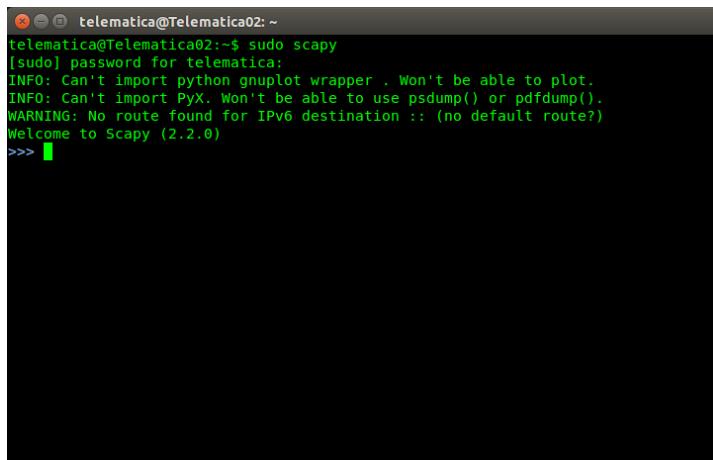
2. Contenido

2.1. Creación de Paquetes

Para crear un paquete con Scapy, este se tiene que ejecutar vía la consola el siguiente comando:

```
sudo scapy
```

Este comando iniciara el programa como super-user donde se podrá usar sus funciones para la creación de los paquetes y enviarlos por la red. Estos se crean en base a las capas del modelo OSI, sin necesidad de seguir un orden específico al crear las capas, las cuales el programa permite su uso desde la Capa 2 hasta la que se necesite para el paquete.

A screenshot of a terminal window titled "telematica@Telematica02: ~". The window shows the command "sudo scapy" being run, followed by a password prompt "[sudo] password for telematica:". The terminal then displays several informational messages from Scapy: "INFO: Can't import python gnuplot wrapper . Won't be able to plot.", "INFO: Can't import PyX. Won't be able to use psdump() or pdfdump()", and "WARNING: No route found for IPv6 destination :: (no default route?)". Finally, it shows the welcome message "Welcome to Scapy (2.2.0)" and a prompt ">>>".

```
telematica@Telematica02: ~
telematica@Telematica02:~$ sudo scapy
[sudo] password for telematica:
INFO: Can't import python gnuplot wrapper . Won't be able to plot.
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().
WARNING: No route found for IPv6 destination :: (no default route?)
Welcome to Scapy (2.2.0)
>>>
```

Iniciando con la Capa 2 esta el comando Ether(), este comando permite modificar los parámetros del enlace de datos, en especial las MACs de destino y origen, en este laboratorio se utiliza en demasía esta capa.

El siguiente comando, el cual se encarga de la Capa 3, es IP() el cual se encarga de los parámetros de enrutamiento incluyendo protocolos y direcciones lógicas del sistema, las direcciones de IP de origen y destino.

A continuación definimos ICMP(), o Internet Control Message Protocol, el cual es un protocolo como UDP o TCP, el cual se encarga de administrar la información relacionada con errores de los equipos en Red, este maneja mensajes de errores y control para los sistemas de la red, informando con ellos a la fuente original para que evite o corrija el problema detectado.

El ultimo comando a usar, el cual se encarga de la información a enviar, es Raw() este se tiene un String como parámetro para el envío de información a ser usada por el equipo de destino.

Una vez creado las capas a usar, con las capas que se estimen convenientes, estas son apiladas en orden ascendente separadas con un '/' para que estas formen un solo paquete que luego puede ser enviado, para el envío del paquete se utiliza el comando sendp(), este lo envía a través de la red hacia su destino dependiendo de como este configurado.

2.2. Hardware utilizado

Para este laboratorio debimos utilizar una red montada con un Switch, para esto fue utilizado un Catalyst 2690 fabricado por Cisco System, y una red montada con un Hub, siendo este un AdvanceStack Switching Hub-12R fabricado por Hawlett Packard.



Los equipos conectados al Switch para realizar las pruebas fueron los equipos del laboratorio de Informática, mientras que los equipos que fueron utilizados para realizar las pruebas con el Hub fueron notebooks.

2.3. Envío de un paquete de datos a FF:FF:FF:FF:FF:FF

Creamos el paquete con la dirección MAC 'FF:FF:FF:FF:FF:FF', ante eso fijamos el valor en el campo de destino, 'dst', con los valores dado por el ejercicio. El resto de los campos son innecesarios para la actividad por lo cual los dejamos por defecto.

```
telematica@Telematica02:~$ sudo scapy
[sudo] password for telematica:
INFO: Can't import python gnuplot wrapper . Won't be able to plot.
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().
WARNING: No route found for IPv6 destination :: (no default route?)
Welcome to Scapy (2.2.0)
>>> link
Traceback (most recent call last):
  File "<console>", line 1, in <module>
NameError: name 'link' is not defined
>>> link=Ether()
>>> ls(link)
WARNING: Mac address to reach destination not found. Using broadcast.
dst      : DestMACField      = 'ff:ff:ff:ff:ff:ff' (None)
src      : SourceMACField    = '00:00:00:00:00:00' (None)
type     : XShortEnumField   = 0           (0)
>>> link.dst="FF:FF:FF:FF:FF:FF"
>>> ls(link)
dst      : DestMACField      = 'FF:FF:FF:FF:FF:FF' (None)
src      : SourceMACField    = '00:00:00:00:00:00' (None)
type     : XShortEnumField   = 0           (0)
>>> 
```

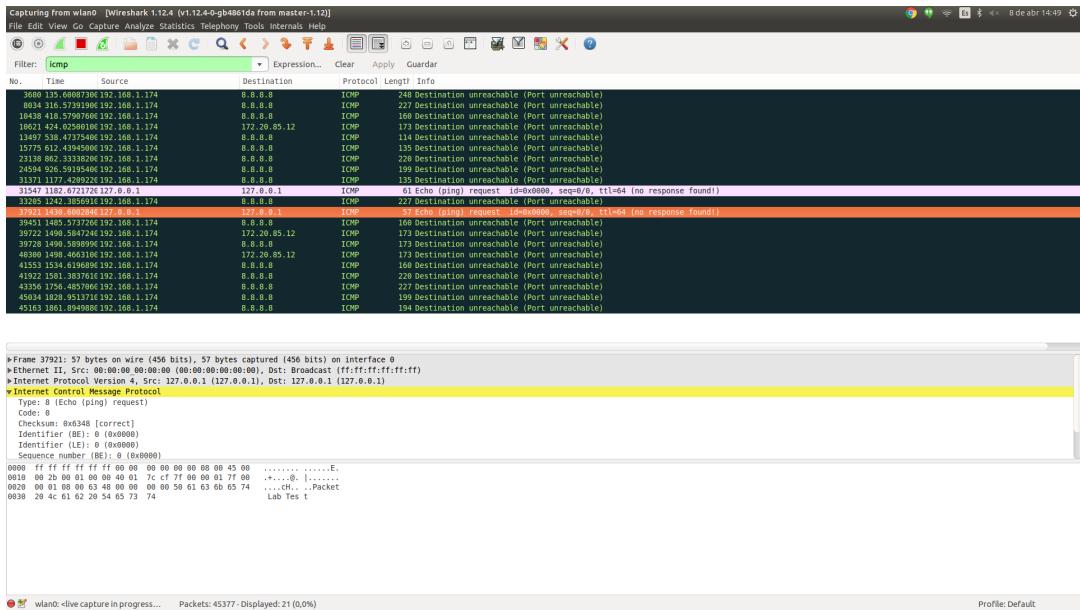
Al no necesitar ningún parámetro extra en las capas superiores solo las definimos, aunque estas no son necesarias para el funcionamiento del paquete . Solo modificamos el parámetro de la función Raw() para poder identificar el paquete que nosotros enviamos, una vez hecha la modificación a ese parámetro apilamos el paquete y procedemos el envío de este.

```
NameError: name 'link' is not defined
>>> link=Ether()
>>> ls(link)
WARNING: Mac address to reach destination not found. Using broadcast.
dst      : DestMACField      = 'ff:ff:ff:ff:ff:ff' (None)
src      : SourceMACField    = '00:00:00:00:00:00' (None)
type     : XShortEnumField   = 0           (0)
>>> link.dst="FF:FF:FF:FF:FF:FF"
>>> ls(link)
dst      : DestMACField      = 'FF:FF:FF:FF:FF:FF' (None)
src      : SourceMACField    = '00:00:00:00:00:00' (None)
type     : XShortEnumField   = 0           (0)
>>> ip=IP()
>>> icmp=ICMP()
>>> rawlin=raw()
>>> rawlin.load="este es un paquetea FF:FF:FF:FF:FF:FF"
>>> packet=link/ip/icmp/rawlin
>>> sendp(packet)
.
Sent 1 packets. 
```

2.3. ENVÍO DE UN PAQUETE DE DATOS A FF:FF:FF:FF:FF:FF

2.3.1. Switch

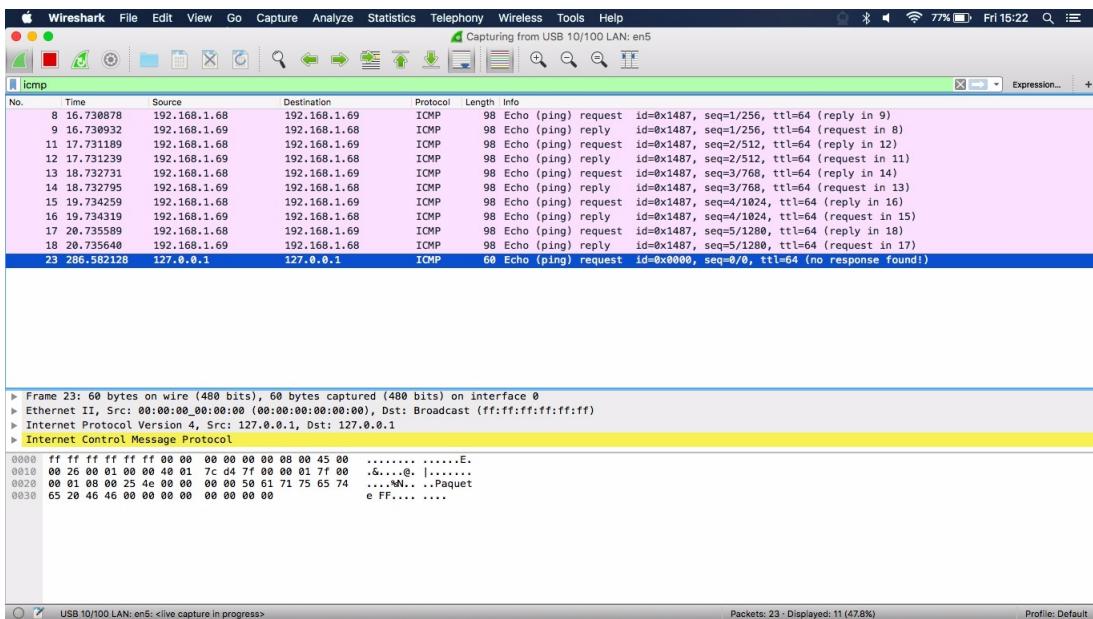
Al ser la dirección MAC 'FF:FF:FF:FF:FF:FF' el Switch no reenvía el paquete a ningún equipo que se encuentre dentro de la red. Adicionalmente a esto si estamos usando Wireshark en modo Promiscuo podemos capturar el paquete se a enviado.



Wireshark corriendo en Modo Promiscuo

2.3.2. Hub

El Hub reenvía el paquete a todos los equipos conectados a este. Pudiendo capturar este de cualquier equipo dentro de la red sin la necesidad de correr Wireshark en modo Promiscuo, debido a que les llega a todos los equipos dentro de esta.



2.4. Envío de un paquete de datos con MAC específica

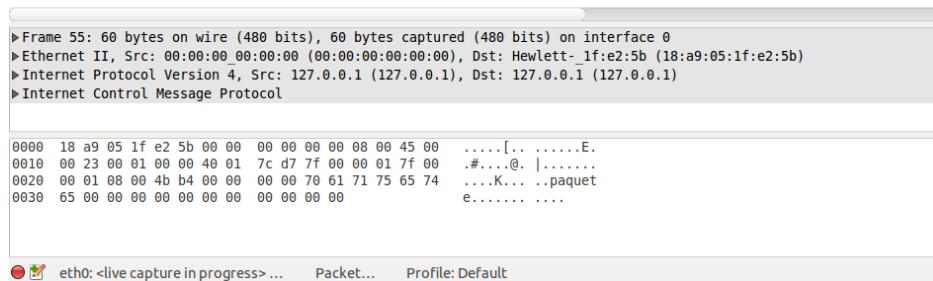
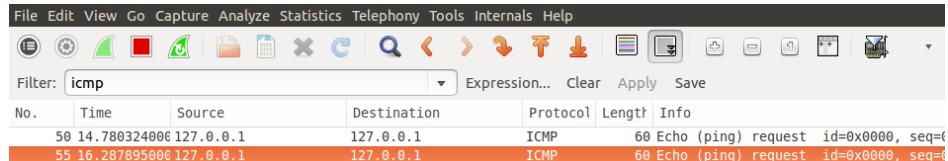
Para este segundo experimento, esta vez usamos una dirección MAC que pertenece a un equipo dentro de la red LAN, al igual que en el experimento anterior no necesitamos usar parámetros adicionales de las capas superiores a excepción de la ultima capa que la usamos para identificar fácilmente nuestro paquete.

2.4.1. Switch

Para este experimento, esta vez usamos la siguiente dirección MAC '18:A9:05:1F:E2:5B'.

```
telematica@Telematica03:~$ sudo scapy
INFO: Can't import python gnuplot wrapper . Won't be able to plot.
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().
WARNING: No route found for IPv6 destination :: (no default route?)
Welcome to Scapy (2.2.0)
>>> enlace=Ether()
>>> ls(enlace)
WARNING: Mac address to reach destination not found. Using broadcast.
dst      : DestMACField      = 'ff:ff:ff:ff:ff:ff' (None)
src      : SourceMACField    = '00:00:00:00:00:00' (None)
type     : XShortEnumField   = 0           (0)
>>> enlace.dst='18:a9:05:1f:e2:5b'
>>> ip=IP()
>>> icmp=ICMP()
>>> raw=Raw()
>>> raw.load='paquete'
>>> paquete=enlace/ip/icmp/raw
>>> sendp(paquete)
.
Sent 1 packets.
>>> 
```

Luego buscamos nuestro paquete en Wireshark para ver si se envió correctamente y llegó a destino.



2.4.2. Hub

Para el experimento con el Hub utilizamos la dirección MAC 'A8:B3:CC:4E:DE:08'

```
amantinetti@Manti-Buntu-Note: ~
TFTP_WRQ : TFTP Write Request
UDP : UDP
UDPPerror : UDP in ICMP
USER_CLASS_DATA : user class data
VENDOR_CLASS_DATA : vendor class data
VENDOR_SPECIFIC_OPTION : vendor specific option data
VRRP : None
X509Cert : None
X509RDN : None
X509v3Ext : None
DHCPoGuessPayload : None
DHCPo6GuessPayload : None
ICMPv6 : ICMPv6 dummy class
ICMPv6Error : ICMPv6 errors dummy class
ICMPv6ML : ICMPv6 dummy class
IPoption_HDR : None
IPv6extHdr : Abstract IPv6 Option Header
_MobilityHeader : Dummy IPv6 Mobility Header
>>> enlace=Ether()
>>> ls(enlace)
WARNING: Mac address to reach destination not found. Using broadcast.
dst : DestMACField      = 'ffff:ffff:ffff:ff:ff' (None)
src : SourceMACField    = '00:00:00:00:00:00' (None)
type : XShortEnumField   = 0          (0)
>>> enlace.dst='A8:B3:CC:4E:DE:08'
>>> ls(enlace)
dst : DestMACField      = 'A8:B3:CC:4E:DE:08' (None)
src : SourceMACField    = '00:00:00:00:00:00' (None)
type : XShortEnumField   = 0          (0)
>>> ip=IP()
>>> icmp=ICMP()
>>> mensaje=Raw()
>>> ls(mensaje)
load : StrField          = ''           ('')
>>> mensaje.load='Mac Henry'
>>> ls(mensaje)
load : StrField          = 'Mac Henry' ('')
>>> paquete=enlace/ip/icmp/mensaje
>>> send(paquete)
Sent 1 packets.
>>> █
>>> █
```

Luego buscamos nuestro paquete en Wireshark para ver si se envío correctamente y llego a destino.

No.	Time	Source	Destination	Protocol	Length	Info
679	249.48214006127.0.0.1	127.0.0.1		ICMP	51	Echo (ping) request id=0x0000, seq=0/0, ttl=64 [no response found!]

>Frame 679: 51 bytes on wire (408 bits), 51 bytes captured (408 bits) on interface 8	
>Ethernet II, Src: HewlettP_4e:de:08 (a8:b3:cc:4e:de:08)	
>Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)	
>Internet Control Message Protocol	
0000	a8 b3 cc 4e de 08 00 00 00 00 00 00 00 00 45 00 ...N....E.
0010	00 25 00 01 00 00 40 01 7c d5 7f 00 00 01 7f 00 .%....@.
0020	00 01 08 00 17 a6 00 00 00 00 4d 61 63 20 48 65Mac He
0030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 nry

Confirmando que el paquete llego a destino como debería.

2.5. Envió de un paquete de datos con una MAC fuera de la red

En esta ocasión se utilizo una dirección MAC de destino escrita al azar que no coincidiera con la de ninguno de los equipos pertenecientes a la red LAN. Luego se creó un paquete tras haber agregado un mensaje en la última capa, sin ningún cambio adicional el paquete fue enviado a la dirección ya mencionada.

2.5.1. Switch

La dirección MAC elegida fuera de la red para enviar el paquete fue '78:E4:00:BB:23:23'

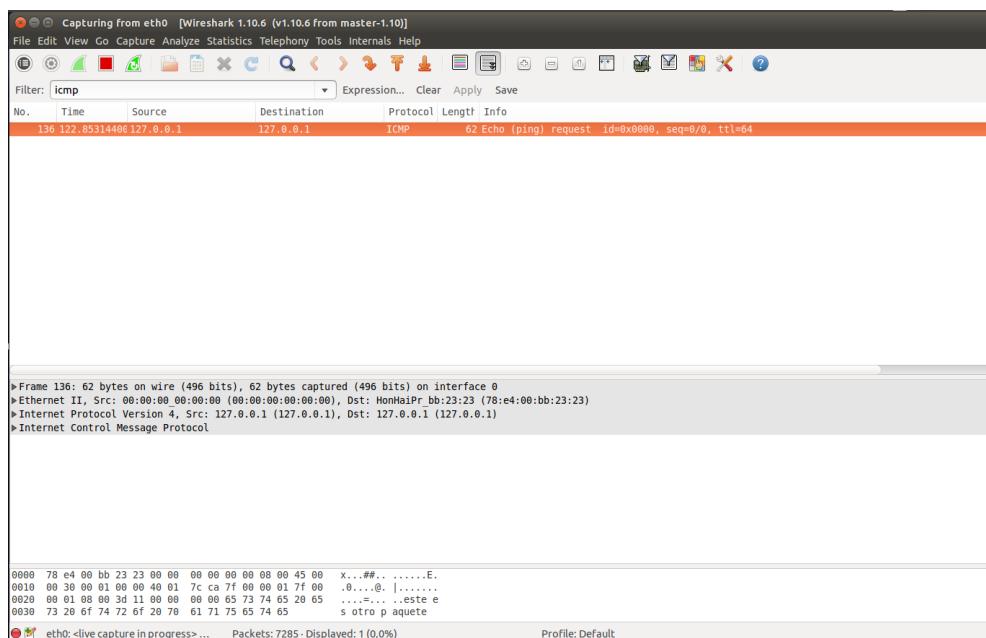
```

File "/usr/lib/python2.7/dist-packages/scapy/utils.py", line 244, in <lambda>
    return "".join(map(lambda x: chr(int(x,16)), mac.split(":")))
ValueError: invalid literal for int() with base 16: 'lf'
>>> clear()
Traceback (most recent call last):
  File "<console>", line 1, in <module>
NameError: name 'clear' is not defined
>>> clear
Traceback (most recent call last):
  File "<console>", line 1, in <module>
NameError: name 'clear' is not defined
>>>
telematica@Telematica02:~$ clear

telematica@Telematica02:~$ sudo scapy
INFO: Can't import python gnuplot wrapper . Won't be able to plot.
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().
WARNING: No route found for IPv6 destination :: (no default route?)
Welcome to Scapy (2.2.0)
>>> link3=Ether()
>>> link3.dst="78:e4:00:bb:23:23"
>>> ls(link3)
dst      : DestMACField      = '78:e4:00:bb:23:23' (None)
src      : SourceMACField    = '00:00:00:00:00:00' (None)
type     : XShortEnumField   = 0          (0)
>>> ip3=IP()
>>> icmp3=ICMP()
>>> rawlin3=Raw()
>>> rawlin3.load="este es otro paquete"
>>> packet=link3/ip3/icmp3/rawlin3
>>> sendp(packet)
.
Sent 1 packets.
>>> █

```

Descubrimos que el paquete no era recibido por ninguno de los equipos.



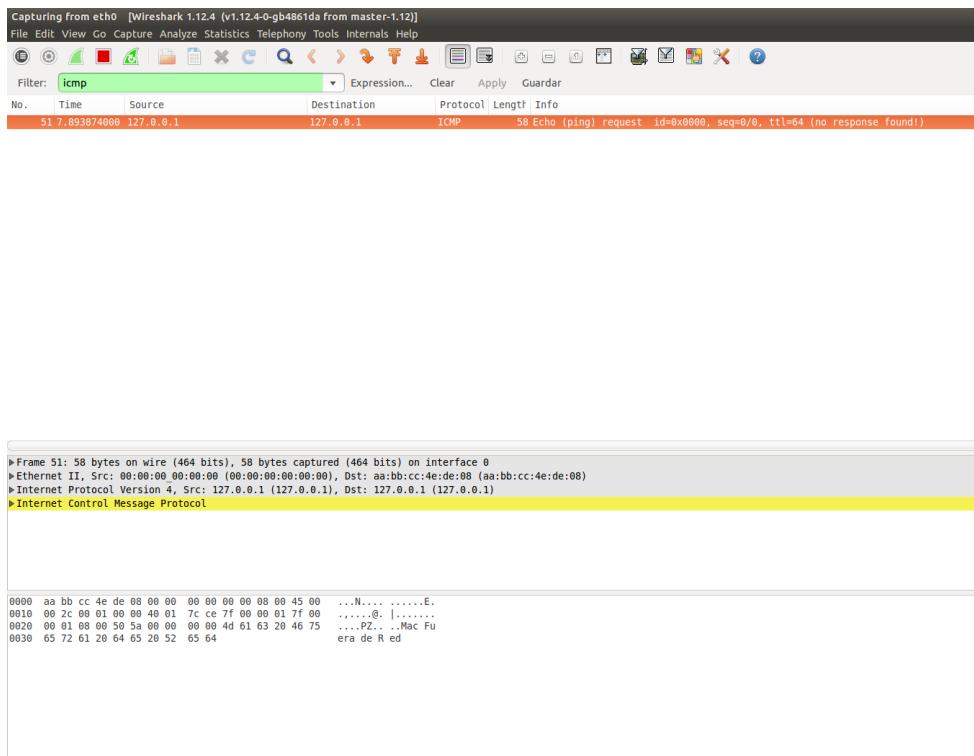
Wireshark corriendo en modo Promiscuo para comprobar el envío del paquete

2.5.2. Hub

La dirección MAC elegida fuera de la red para enviar el paquete fue 'AA:BB:CC:4E:DE:08'

```
root@amantinetti:~# ./script.py
ICMPv6Error : ICMPv6 errors dummy class
ICMPv6ML : ICMPv6 dummy class
IPoption_HDR : None
IPv6ExtHdr : Abstract IPv6 Option Header
MobilityHeader : Dummy IPv6 Mobility Header
>>> enlace=Ether()
>>> ls(enlace)
WARNING: Mac address to reach destination not found. Using broadcast.
dst : DestMACField      = 'ffff:ffff:ffff' (None)
src : SourceMACField    = '00:00:00:00:00:00' (None)
type : XshortEnumField  = 0          (0)
>>> enlace.dst='AA:BB:CC:4E:DE:08'
>>> ls(enlace)
dst : DestMACField      = 'AA:BB:CC:4E:DE:08' (None)
src : SourceMACField    = '00:00:00:00:00:00' (None)
type : XshortEnumField  = 0          (0)
>>> lpc=IP()
>>> lcp=ICMP()
>>> mensaje=Raw()
>>> ls(mensaje)
load : StrField         = ''          ('')
>>> mensaje.load='Mac Henry'
>>> ls(mensaje)
load : StrField         = 'Mac Henry' ('')
>>> paquet=enlace/lp/lcp/mensaje
>>> sendp(paquet)
.
Sent 1 packets.
>>> mensaje.load='Mac Fuerza de Red'
>>> paquet=enlace/lp/lcp/mensaje
>>> sendp(paquet)
.
Sent 1 packets.
>>> sendp(paquet)
.
Sent 1 packets.
>>> 
```

Este paquete era recibido por todos los equipos conectados al Hub



Wireshark corriendo normalmente

3. Conclusión