



**udp** UNIVERSIDAD  
DIEGO PORTALES

Facultad de Ingeniería  
Escuela de Informática y Telecomunicaciones

---

## 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

10 de Abril de 2016

# Índice general

<b>1. Introducción</b>	<b>2</b>
<b>2. Contenido</b>	<b>3</b>
2.1. Creación de Paquetes . . . . .	3
2.2. Envío de un paquete de datos a FF:FF:FF:FF:FF:FF . . . . .	4
2.2.1. Switch . . . . .	4
2.2.2. Hub . . . . .	4
2.3. Envío de un paquete de datos con MAC específica . . . . .	4
2.3.1. Switch . . . . .	4
2.3.2. Hub . . . . .	6
2.4. Envío de un paquete de datos con una MAC fuera de la red . . . . .	6
2.4.1. Switch . . . . .	6
2.4.2. Hub . . . . .	8
<b>3. Conclusión</b>	<b>9</b>

# 1. Introducción

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

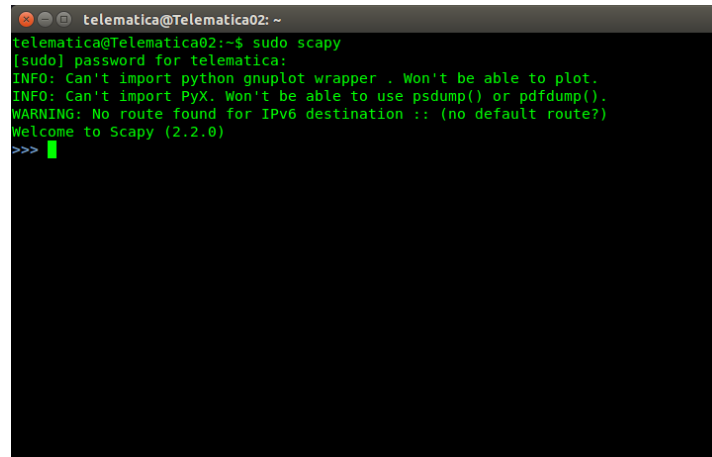
Los paquetes, en este experimento, varían principalmente en la dirección MAC, lo que hace que sean recibidos por distintos equipos. Para esto se ocupa 'Wireshark', programa con el que se puede capturar los paquetes enviados por la red. Una vez creados y enviados los paquetes a través del Switch, se repite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

Este comando iniciara el programa donde se podrá usar sus funciones para la creación de los paquetes. 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.



```
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.

[...] 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()`

## 2.2. 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' ...

```
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/instanciamos, aunque esto no es necesario 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 al 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()
Traceback (most recent call last):
  File "<console>", line 1, in <module>
NameError: name 'raw' is not defined
>>> rawlin=Raw()
>>> rawlin.load="este es un paquetea FF:FF:FF:FF:FF:FF"
>>> packet=link/ip/icmp/rawlin
>>> sendp(packet)
.
Sent 1 packets.
```

### 2.2.1. 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, si estamos usando Wireshark en modo Promiscuo podemos ver que el paquete se envía.

### 2.2.2. Hub

## 2.3. Envío de un paquete de datos con MAC específica

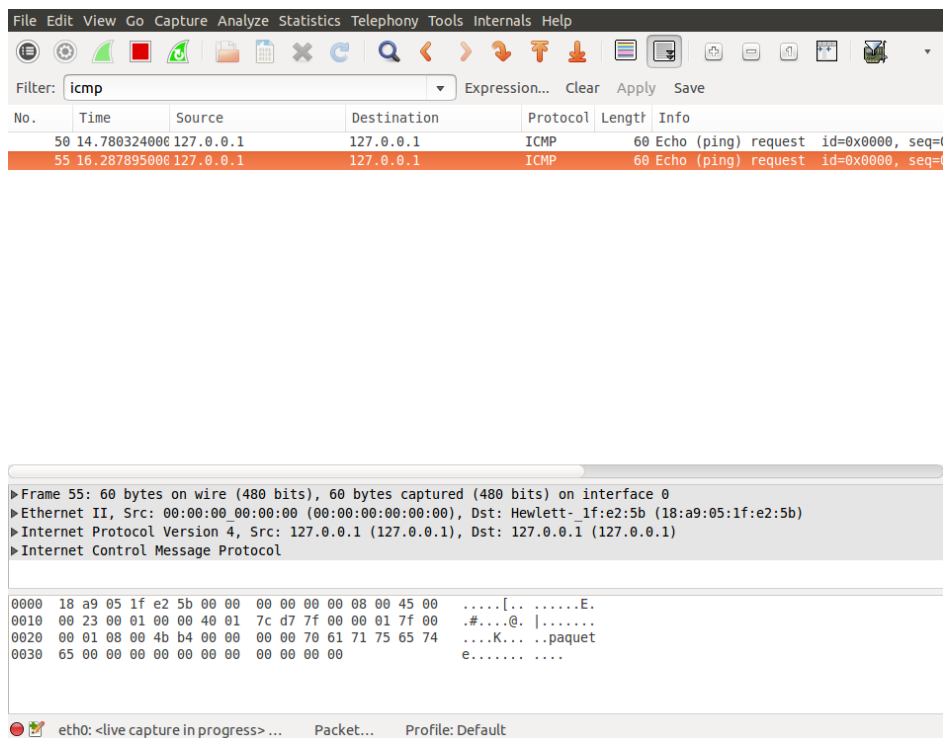
### 2.3.1. Switch

Para este segundo experimento, esta vez usamos la siguiente dirección MAC '18:a9:05:1f:e2:5b' 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 facilmente nuestro paquete.

```
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 envio correctamente y luego a destino.



### 2.3.2. Hub

## 2.4. Envío de un paquete de datos con una MAC fuera de la red

### 2.4.1. Switch

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, esta dirección fue '78:e4:00:bb:23:23'. 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.

## 2.4. ENVÍO DE UN PAQUETE DE DATOS CON UNA MAC FUERA DE LA RED

```
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: '\n'
>>> 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.
>>> █
```

Capturing from eth0 [Wireshark 1.10.6 (v1.10.6 from master-1.10)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: icmp Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
136	122.85314406	127.0.0.1	127.0.0.1	ICMP	62	Echo (ping) request id=0x0000, seq=0/0, ttl=64

► Frame 136: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface 0  
► Ethernet II, Src: 00:00:00\_00:00:00 (00:00:00:00:00:00), Dst: HonHaiPr\_bb:23:23 (78:e4:00:bb:23:23)  
► 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 78 e4 00 bb 23 23 00 00 00 00 00 00 00 45 00 x...##.. ..E.  
0010 00 30 00 01 00 00 40 01 7c ca 7f 00 00 01 7f 00 .0...@. |.....  
0020 00 01 08 00 3d 11 00 00 00 00 65 73 74 65 20 65 .......este e  
0030 73 20 6f 74 72 6f 20 70 61 71 75 65 74 65 s otro p aquete

eth0: <live capture in progress> ... Packets: 7285 · Displayed: 1 (0,0%) Profile: Default

**2.4.2. Hub**



### 3. Conclusión