Redes de Computadores

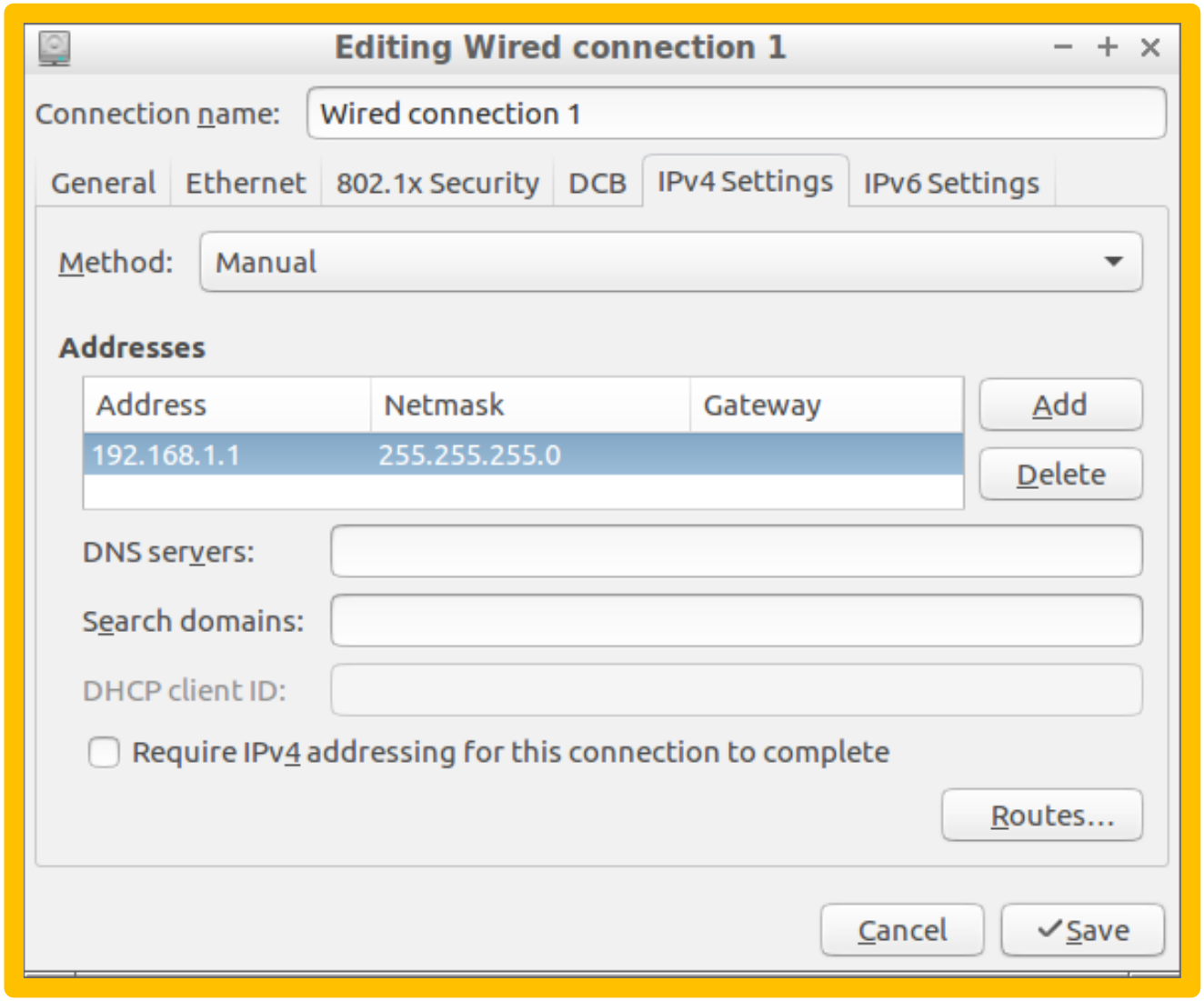
Grado en Ingeniería Informática en Tecnologías de la Información

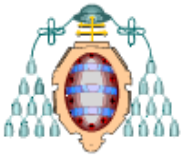
Curso 2022-2023

Práctica 3: Conectividad

Francisco González Bulnes
Pelayo Nuño Huergo
Pablo Alonso García
Área de Ingeniería Telemática
Universidad de Oviedo

Configuración de Parámetros de Red





Configuración de Parámetros de Red

Ejemplo 1: envío a un destino en la misma red

IP emisor: 166.45.161.2
Máscara emisor: 255.255.255.0

IP receptor: 166.45.161.1

Configuración de Parámetros de Red

Ejemplo 1: envío a un destino en la misma red

	10100110. 00101101. 10100001. 00000010	IP emisor: 166.45.161.2
AND	11111111. 11111111. 11111111. 00000000	Máscara emisor: 255.255.255.0
	10100110. 00101101. 10100001. 00000000	Dirección de red: 166.45.161.0
		IP receptor: 166.45.161.1

Configuración de Parámetros de Red

Ejemplo 1: envío a un destino en la misma red

	10100110. 00101101. 10100001. 00000010	IP emisor: 166.45.161.2
AND	11111111. 11111111. 11111111. 00000000	Máscara emisor: 255.255.255.0
	10100110. 00101101. 10100001. 00000000	Dirección de red: 166.45.161.0
	10100110. 00101101. 10100001. 00000001	IP receptor: 166.45.161.1
AND	11111111. 11111111. 11111111. 00000000	Máscara emisor: 255.255.255.0
	10100110. 00101101. 10100001. 00000000	Dirección de red: 166.45.161.0

Configuración de Parámetros de Red

Ejemplo 2: envío a un destino perteneciente a otra red

IP emisor: 166.45.161.2
Máscara emisor: 255.255.255.0

IP receptor: 166.45.160.1

Configuración de Parámetros de Red

Ejemplo 2: envío a un destino perteneciente a otra red

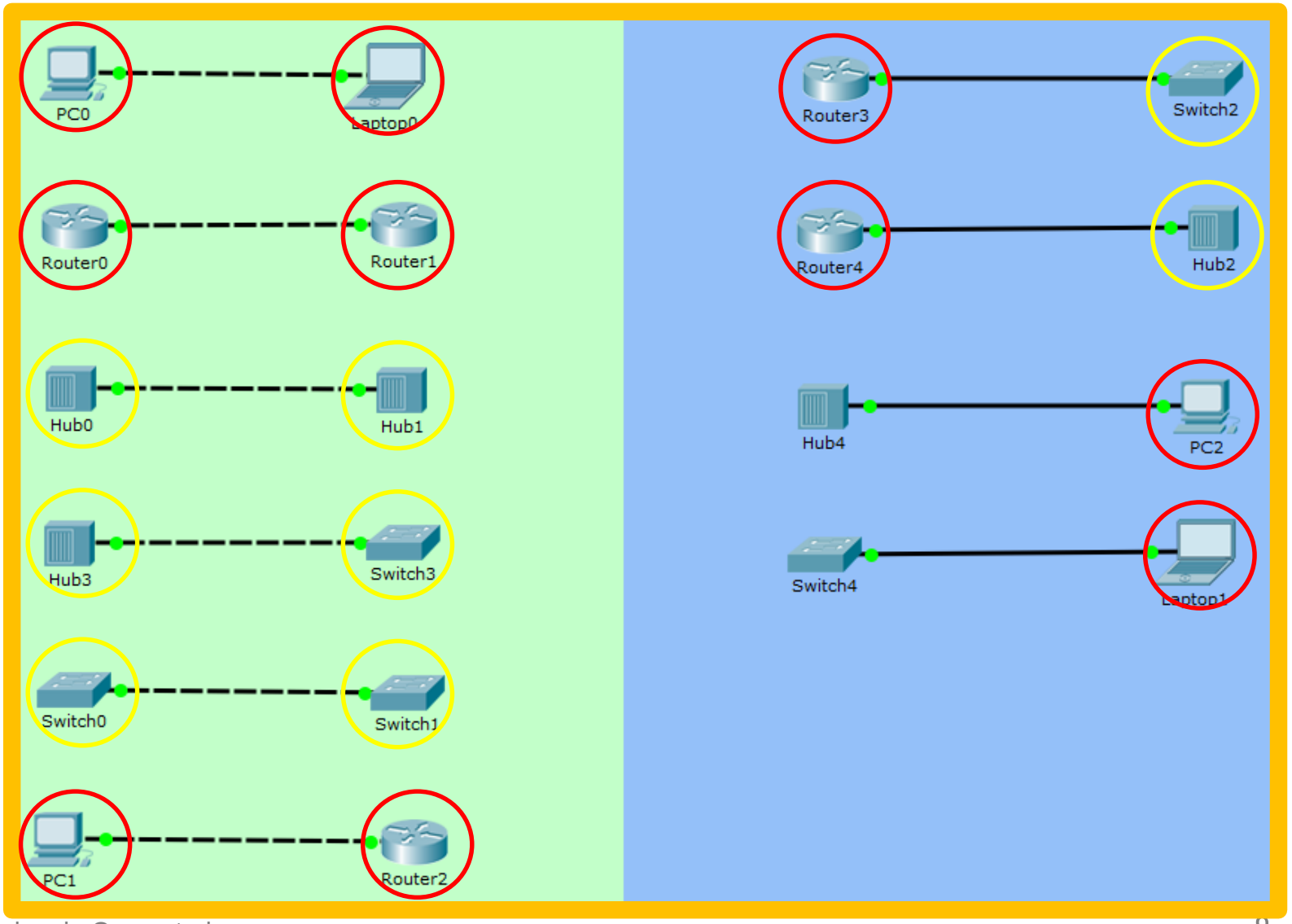
	10100110. 00101101. 10100001. 00000010	IP emisor: 166.45.161.2
AND	11111111. 11111111. 11111111. 00000000	Máscara emisor: 255.255.255.0
	10100110. 00101101. 10100001. 00000000	Dirección de red: 166.45.161.0
		IP receptor: 166.45.160.1

Configuración de Parámetros de Red

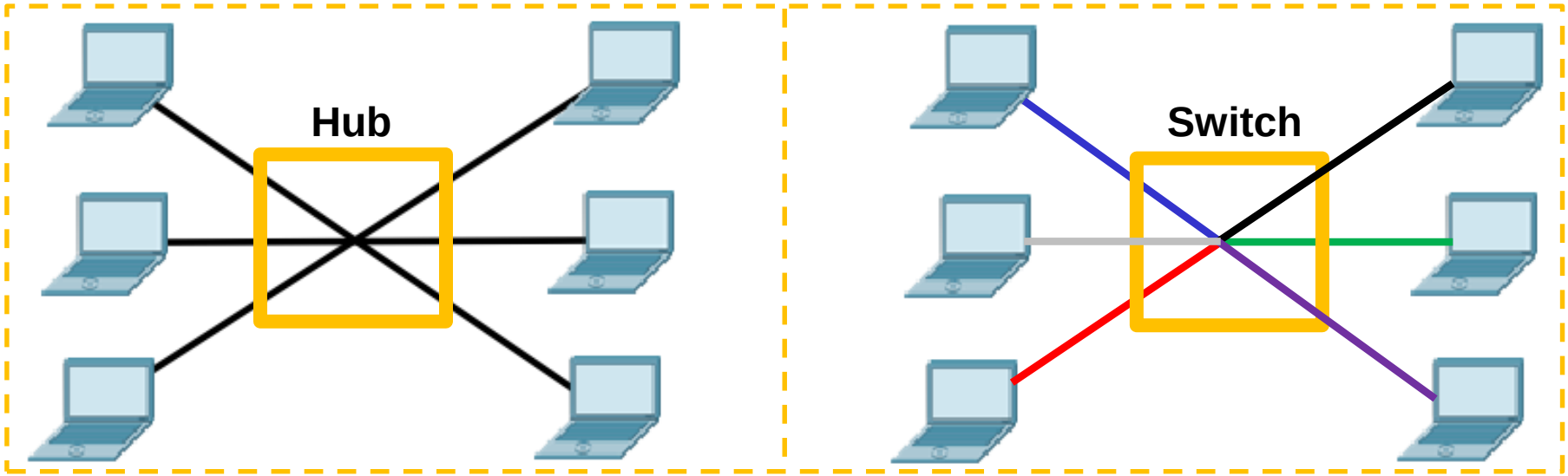
Ejemplo 2: envío a un destino perteneciente a otra red

	10100110. 00101101. 10100001. 00000010	IP emisor: 166.45.161.2
AND	11111111. 11111111. 11111111. 00000000	Máscara emisor: 255.255.255.0
	10100110. 00101101. 10100001. 00000000	Dirección de red: 166.45.161.0
	10100110. 00101101. 10100000. 00000001	IP receptor: 166.45.160.1
AND	11111111. 11111111. 11111111. 00000000	Máscara emisor: 255.255.255.0
	10100110. 00101101. 10100000. 00000000	Dirección de red: 166.45.160.0

Cables Cruzados y Directos

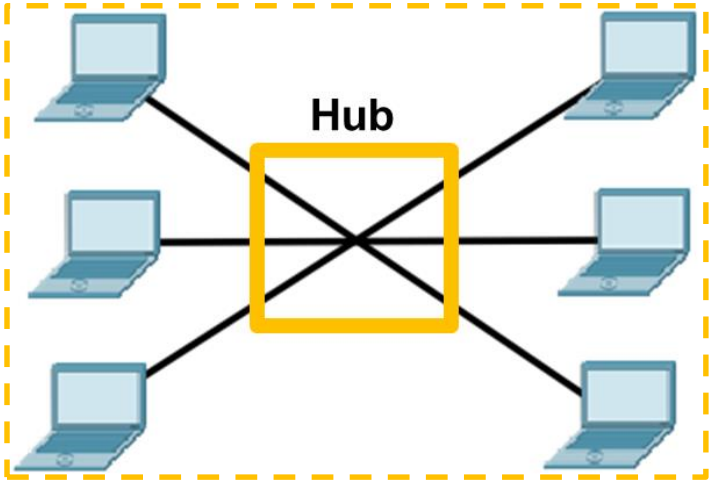


Ejemplo de configuraciones LAN



Cómo funciona un Hub

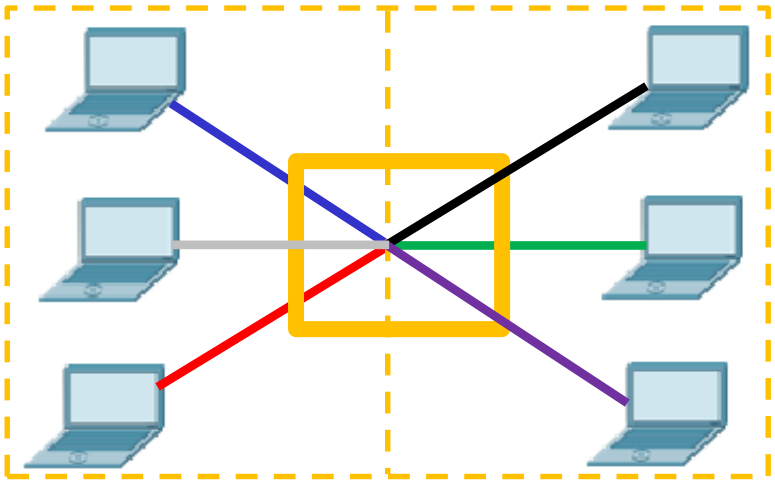
- Hubs (concentradores LAN)
  - Permiten incrementar el tamaño (longitud) de la red
  - La señal que recibe del medio físico la propaga por todos sus puertos
  - Todos los cables conforman un único dominio de transmisión



- Limitaciones:
  - Half dúplex → Existen colisiones en la red
  - Ancho de banda compartido → Uso ineficiente de los recursos
  - No es posible actuar en base al tráfico que circula

Cómo funciona un Switch

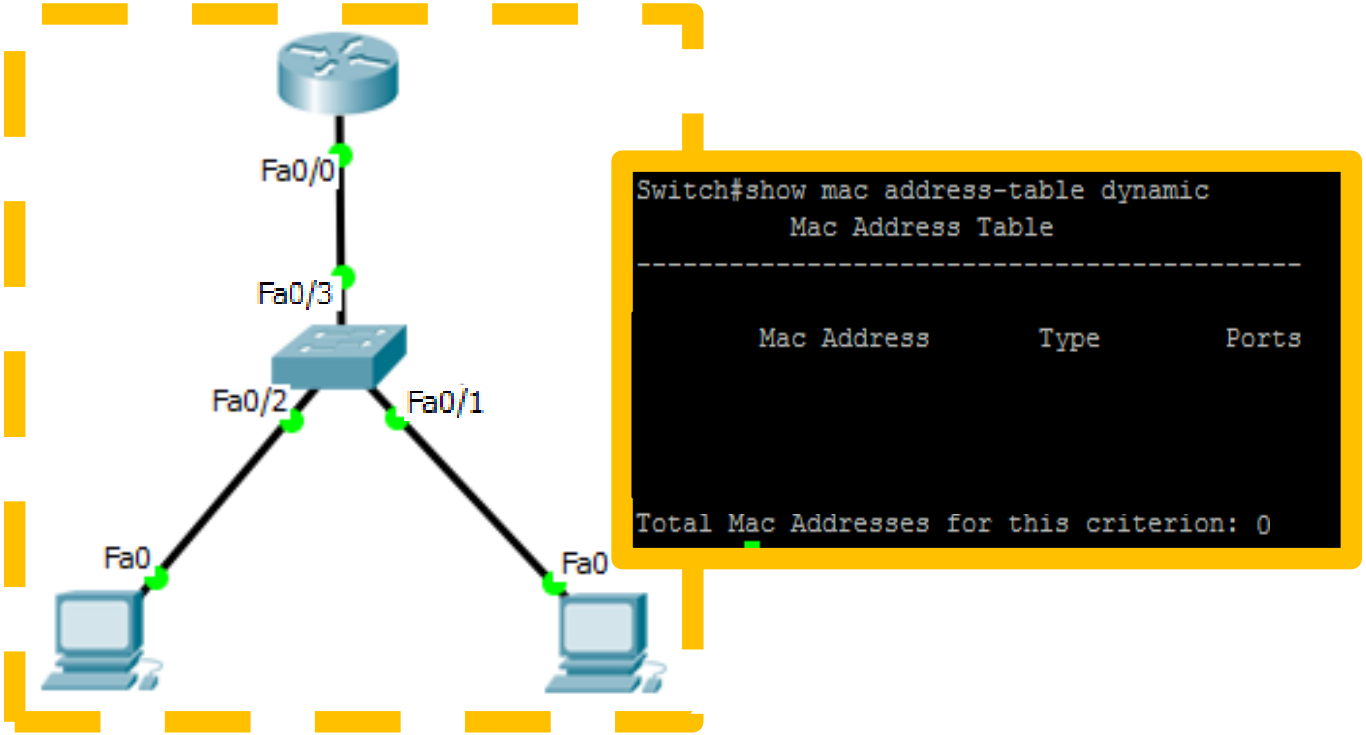
- Switches (conmutadores LAN)
  - Full Dúplex → Eliminan las colisiones en una red local
  - Uso eficiente del ancho de banda
  - Implementan funcionalidades avanzadas de seguridad



- Funcionalidades básicas comunes en la mayoría de switches
  - Aprendizaje de direcciones MAC (Aprendizaje hacia atrás)
  - Decisiones de reenvío/filtrado de tramas
  - Algoritmos para evitar bucles en la red

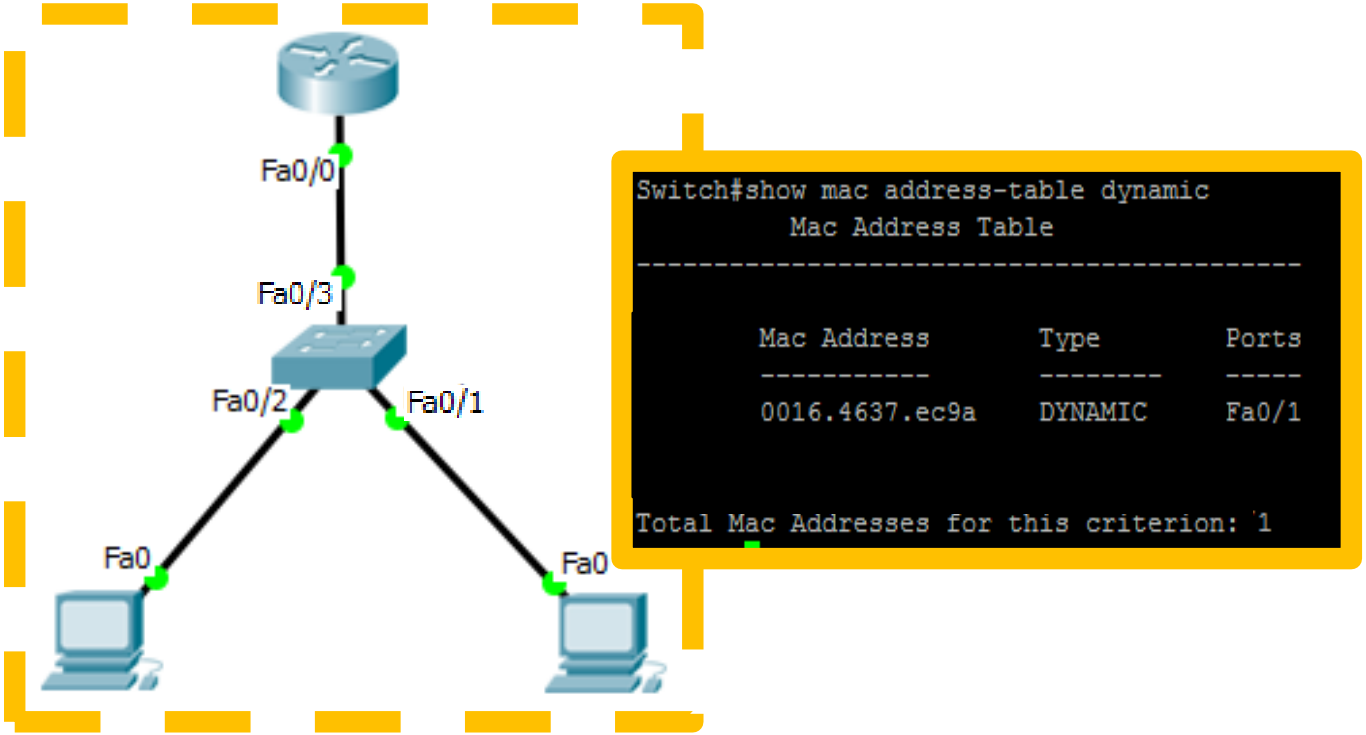
Cómo funciona un Switch

- Al principio el switch no conoce la posición de ningún equipo LAN en la red
- Mecanismo de aprendizaje hacia atrás
  - A medida que recibe tramas actualiza tabla
- Conmutación mediante tabla de direcciones MAC



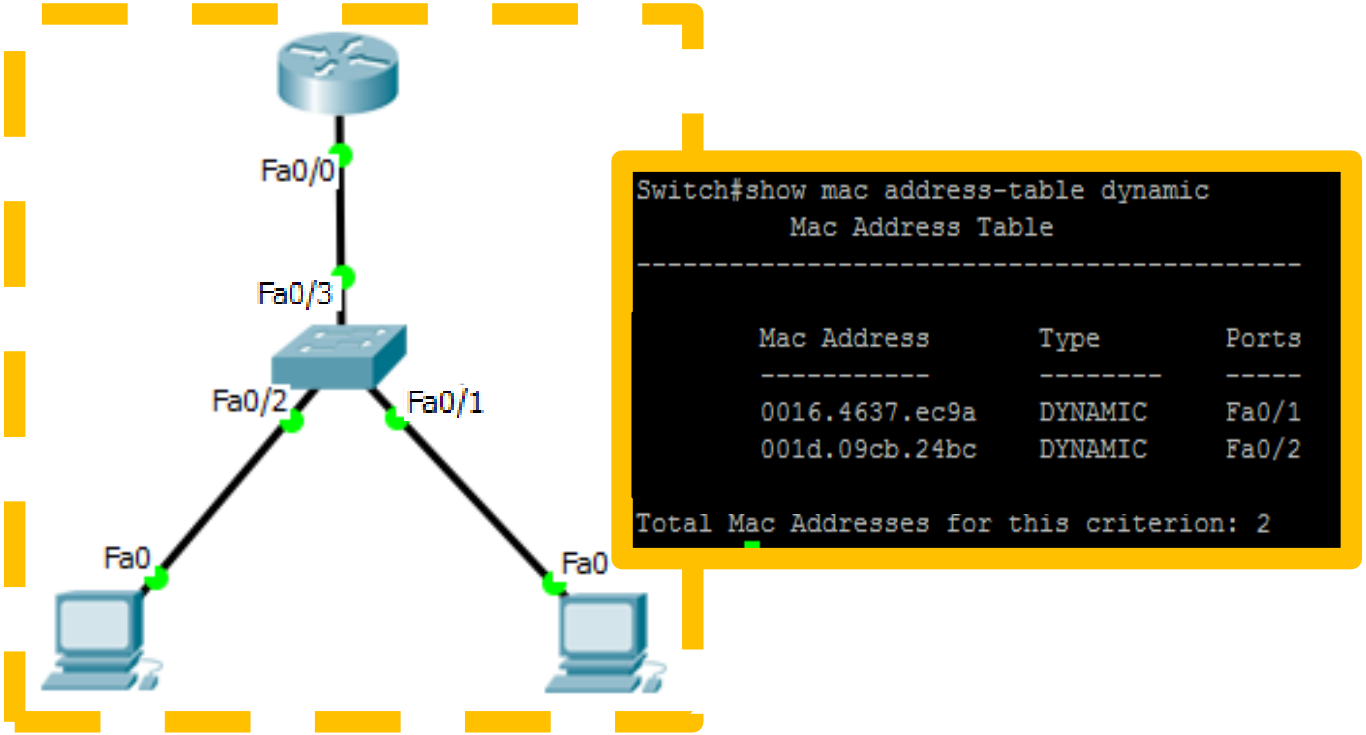
Cómo funciona un Switch

- Al principio el switch no conoce la posición de ningún equipo LAN en la red
- Mecanismo de aprendizaje hacia atrás
  - A medida que recibe tramas actualiza tabla
- Conmutación mediante tabla de direcciones MAC



Cómo funciona un Switch

- Al principio el switch no conoce la posición de ningún equipo LAN en la red
- Mecanismo de aprendizaje hacia atrás
  - A medida que recibe tramas actualiza tabla
- Conmutación mediante tabla de direcciones MAC



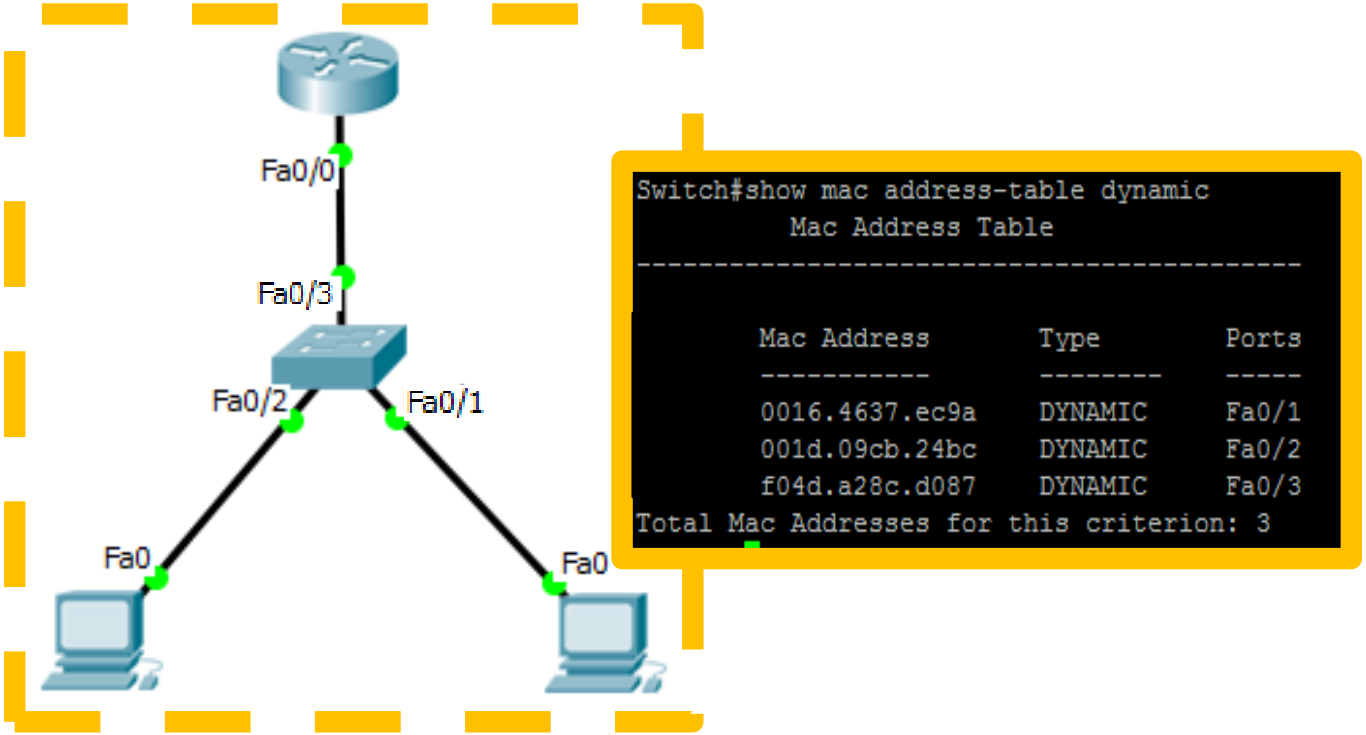

```
Switch#show mac address-table dynamic
          Mac Address Table
-------------------------------------------

        Mac Address       Type        Ports
        -----------       --------    -----
        0016.4637.ec9a    DYNAMIC     Fa0/1
        001d.09cb.24bc    DYNAMIC     Fa0/2

Total Mac Addresses for this criterion: 2
```

Cómo funciona un Switch

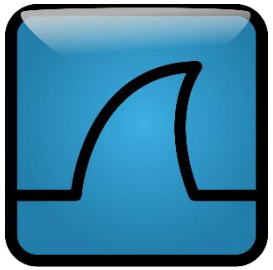
- Al principio el switch no conoce la posición de ningún equipo LAN en la red
- Mecanismo de aprendizaje hacia atrás
  - A medida que recibe tramas actualiza tabla
- Conmutación mediante tabla de direcciones MAC



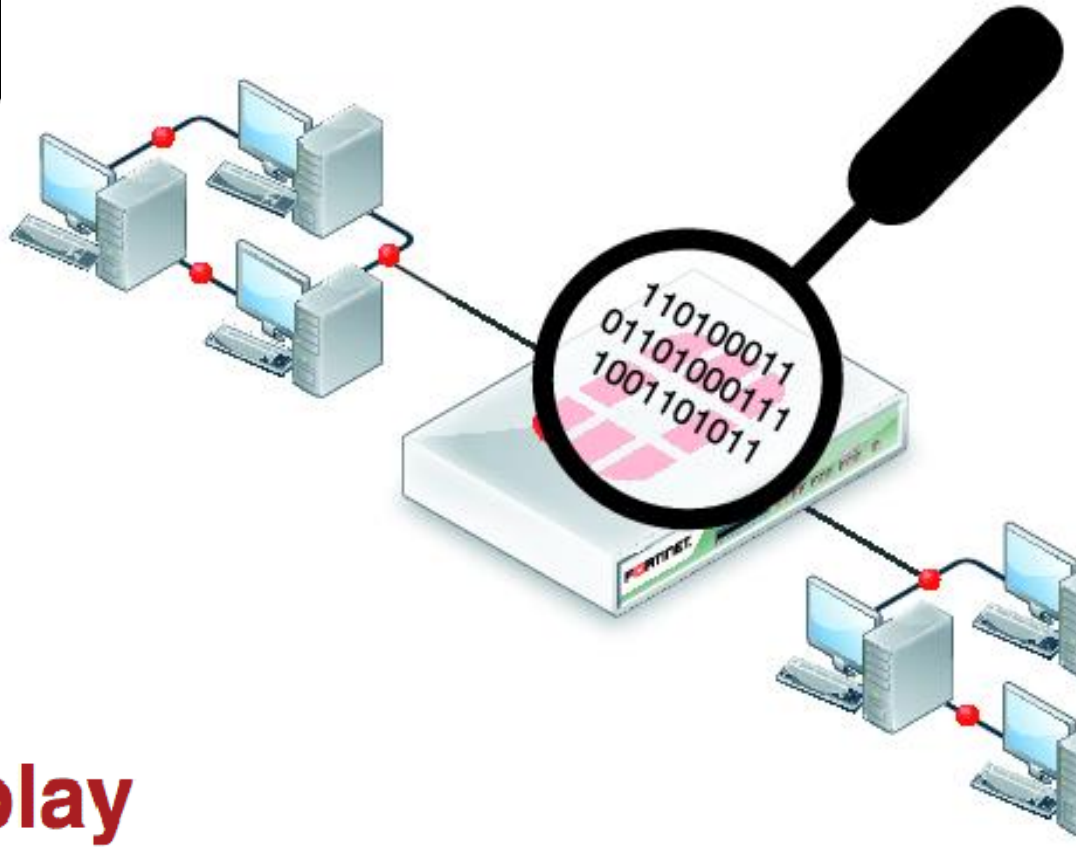

```
Switch#show mac address-table dynamic
          Mac Address Table
-------------------------------------------

        Mac Address       Type        Ports
        -----------       --------    -----
        0016.4637.ec9a    DYNAMIC     Fa0/1
        001d.09cb.24bc    DYNAMIC     Fa0/2
        f04d.a28c.d087    DYNAMIC     Fa0/3
Total Mac Addresses for this criterion: 3
```

Captura, análisis y generación de Tráfico



TCPDUMP

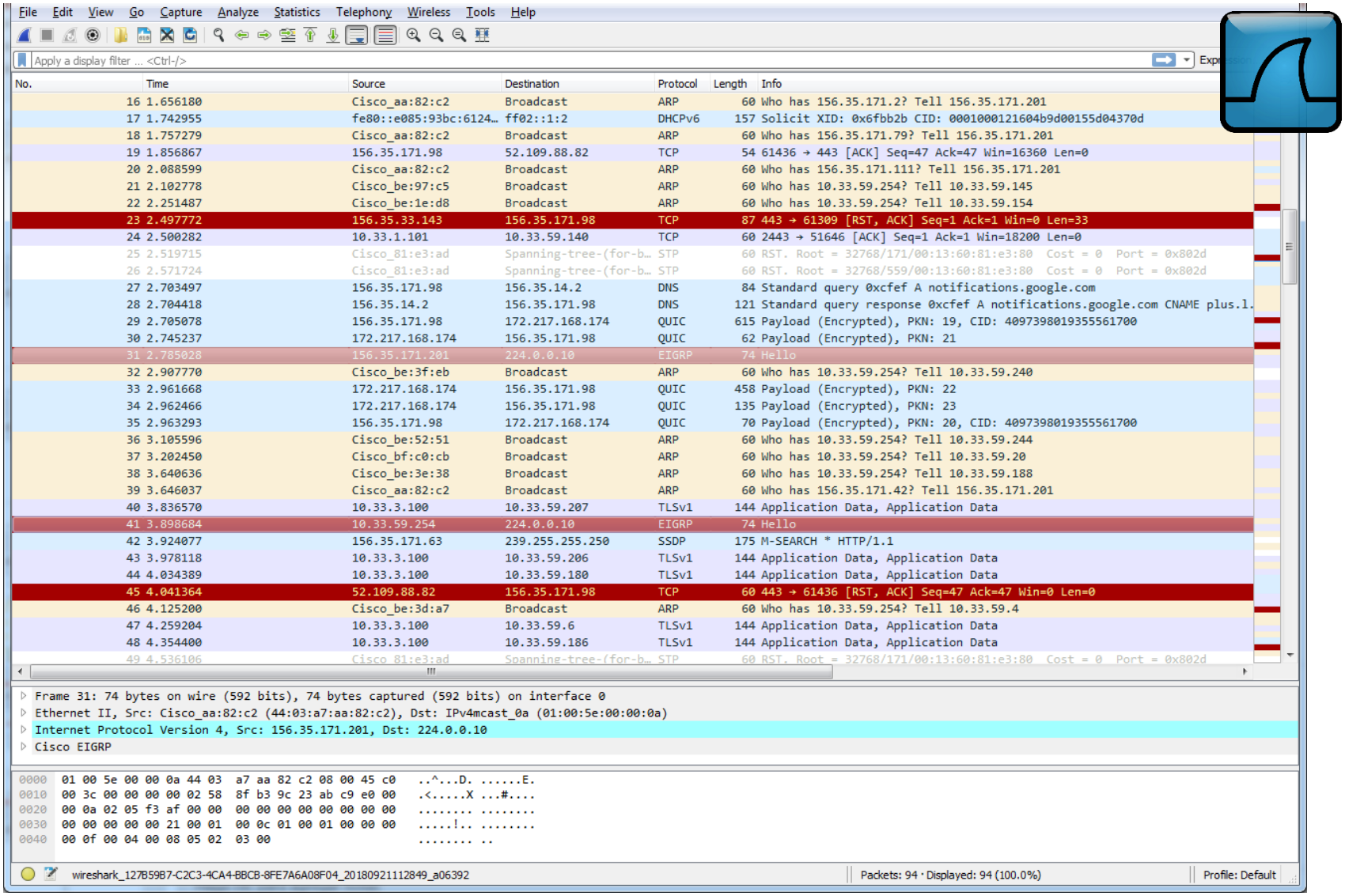


Tcpreplay

scapy

Wireshark

Wireshark


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
16 1.656180 Cisco_aa:82:c2 Broadcast ARP 60 Who has 156.35.171.2? Tell 156.35.171.201
17 1.742955 fe80::e085:93bc:6124… ff02::1:2 DHCPv6 157 Solicit XID: 0x6fbb2b CID: 0001000121604b9d00155d04370d
18 1.757279 Cisco_aa:82:c2 Broadcast ARP 60 Who has 156.35.171.79? Tell 156.35.171.201
19 1.856867 156.35.171.98 52.109.88.82 TCP 54 61436 → 443 [ACK] Seq=47 Ack=47 Win=16360 Len=0
20 2.088599 Cisco_aa:82:c2 Broadcast ARP 60 Who has 156.35.171.111? Tell 156.35.171.201
21 2.102778 Cisco_be:97:c5 Broadcast ARP 60 Who has 10.33.59.254? Tell 10.33.59.145
22 2.251487 Cisco_be:1e:d8 Broadcast ARP 60 Who has 10.33.59.254? Tell 10.33.59.154
23 2.497772 156.35.33.143 156.35.171.98 TCP 87 443 → 61309 [RST, ACK] Seq=1 Ack=1 Win=0 Len=33
24 2.500282 10.33.1.101 10.33.59.140 TCP 60 2443 → 51646 [ACK] Seq=1 Ack=1 Win=18200 Len=0
25 2.519715 Cisco_81:e3:ad Spanning-tree-(for-b… STP 60 RST. Root = 32768/171/00:13:60:81:e3:80 Cost = 0 Port = 0x802d
26 2.571724 Cisco_81:e3:ad Spanning-tree-(for-b… STP 60 RST. Root = 32768/559/00:13:60:81:e3:80 Cost = 0 Port = 0x802d
27 2.703497 156.35.171.98 156.35.14.2 DNS 84 Standard query 0xcfef A notifications.google.com
28 2.704418 156.35.14.2 156.35.171.98 DNS 121 Standard query response 0xcfef A notifications.google.com CNAME plus.l.
29 2.705078 156.35.171.98 172.217.168.174 QUIC 615 Payload (Encrypted), PKN: 19, CID: 4097398019355561700
30 2.745237 172.217.168.174 156.35.171.98 QUIC 62 Payload (Encrypted), PKN: 21
31 2.785028 156.35.171.201 224.0.0.10 EIGRP 74 Hello
32 2.907770 Cisco_be:3f:eb Broadcast ARP 60 Who has 10.33.59.254? Tell 10.33.59.240
33 2.961668 172.217.168.174 156.35.171.98 QUIC 458 Payload (Encrypted), PKN: 22
34 2.962466 172.217.168.174 156.35.171.98 QUIC 135 Payload (Encrypted), PKN: 23
35 2.963293 156.35.171.98 172.217.168.174 QUIC 70 Payload (Encrypted), PKN: 20, CID: 4097398019355561700
36 3.105596 Cisco_be:52:51 Broadcast ARP 60 Who has 10.33.59.254? Tell 10.33.59.244
37 3.202450 Cisco_bf:c0:cb Broadcast ARP 60 Who has 10.33.59.254? Tell 10.33.59.20
38 3.640636 Cisco_be:3e:38 Broadcast ARP 60 Who has 10.33.59.254? Tell 10.33.59.188
39 3.646037 Cisco_aa:82:c2 Broadcast ARP 60 Who has 156.35.171.42? Tell 156.35.171.201
40 3.836570 10.33.3.100 10.33.59.207 TLSv1 144 Application Data, Application Data
41 3.898684 10.33.59.254 224.0.0.10 EIGRP 74 Hello
42 3.924077 156.35.171.63 239.255.255.250 SSDP 175 M-SEARCH * HTTP/1.1
43 3.978118 10.33.3.100 10.33.59.206 TLSv1 144 Application Data, Application Data
44 4.034389 10.33.3.100 10.33.59.180 TLSv1 144 Application Data, Application Data
45 4.041364 52.109.88.82 156.35.171.98 TCP 60 443 → 61436 [RST, ACK] Seq=47 Ack=47 Win=0 Len=0
46 4.125200 Cisco_be:3d:a7 Broadcast ARP 60 Who has 10.33.59.254? Tell 10.33.59.4
47 4.259204 10.33.3.100 10.33.59.6 TLSv1 144 Application Data, Application Data
48 4.354400 10.33.3.100 10.33.59.186 TLSv1 144 Application Data, Application Data
49 4.536106 Cisco_81:e3:ad Spanning-tree-(for-b… STP 60 RST. Root = 32768/171/00:13:60:81:e3:80 Cost = 0 Port = 0x802d
Frame 31: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
Ethernet II, Src: Cisco_aa:82:c2 (44:03:a7:aa:82:c2), Dst: IPv4mcast_0a (01:00:5e:00:00:0a)
Internet Protocol Version 4, Src: 156.35.171.201, Dst: 224.0.0.10
Cisco EIGRP
0000 01 00 5e 00 00 0a 44 03 a7 aa 82 c2 08 00 45 c0 ..^...D. ......E.
0010 00 3c 00 00 00 00 02 58 8f b3 9c 23 ab c9 e0 00 .<.....X ...#....
0020 00 0a 02 05 f3 af 00 00 00 00 00 00 00 00 00 00 ....
0030 00 00 00 00 00 21 00 01 00 0c 01 00 01 00 00 00 .....!.. ....
0040 00 0f 00 04 00 08 05 02 03 00 ....
wireshark_127B59B7-C2C3-4CA4-BBCB-8FE7A6A08F04_20180921112849_a06392
Packets: 94 · Displayed: 94 (100.0%)
Profile: Default

Wireshark

- Panel superior:
  - Listado de las tramas que se reciben a través de la interfaz de red

30	2.745237	172.217.168.174	156.35.171.98	QUIC	62 Payload (Encrypted), PKN: 21
31	2.785028	156.35.171.201	224.0.0.10	EIGRP	74 Hello
32	2.907770	Cisco_be:3f:eb	Broadcast	ARP	60 Who has 10.33.59.254? Tell 10.33.59.240
33	2.961668	172.217.168.174	156.35.171.98	QUIC	458 Payload (Encrypted), PKN: 22
34	2.962466	172.217.168.174	156.35.171.98	QUIC	135 Payload (Encrypted), PKN: 23
35	2.963293	156.35.171.98	172.217.168.174	QUIC	70 Payload (Encrypted), PKN: 20, CID: 4097398019355561700

- Panel intermedio:
  - Detalles a nivel de protocolo de aquellos que componen la trama seleccionada

```
▷ Frame 31: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
▷ Ethernet II, Src: Cisco_aa:82:c2 (44:03:a7:aa:82:c2), Dst: IPv4mcast_0a (01:00:5e:00:00:0a)
▷ Internet Protocol Version 4, Src: 156.35.171.201, Dst: 224.0.0.10
▷ Cisco EIGRP
```

- Panel inferior:
  - Los bytes en crudo de la trama recibida

```
0000   01 00 5e 00 00 0a 44 03   a7 aa 82 c2 08 00 45 c0   ..^...D. ......E.
0010   00 3c 00 00 00 00 02 58   8f b3 9c 23 ab c9 e0 00   .<.....X ...#....
0020   00 0a 02 05 f3 af 00 00   00 00 00 00 00 00 00 00   ........ ........
0030   00 00 00 00 00 21 00 01   00 0c 01 00 01 00 00 00   .....!.. ........
0040   00 0f 00 04 00 08 05 02   03 00                     ........ ..
```

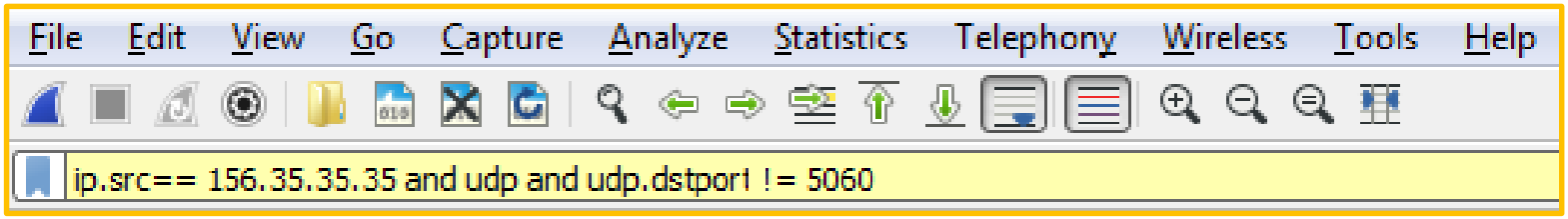
Wireshark

- Campos que conforman el panel superior:
  - **No.:** Número de orden de la trama. La primera trama que se capturó tiene el nº 1, y así, sucesivamente.
  - **Time:** Es una referencia al instante de tiempo en el que se capturó la trama.
  - **Source:** Dirección de origen de la trama. Puede ser una dirección IP o MAC.
  - **Destination:** Dirección de destino de la trama. Puede ser una dirección IP o MAC.
  - **Protocol:** Protocolo superior que contiene la trama capturada. Wireshark muestra el protocolo de más alto nivel que logra identificar.
  - **Info:** Información que contiene la trama. Cuando son datos de usuario, muestra unos pocos bytes nada más.

Wireshark

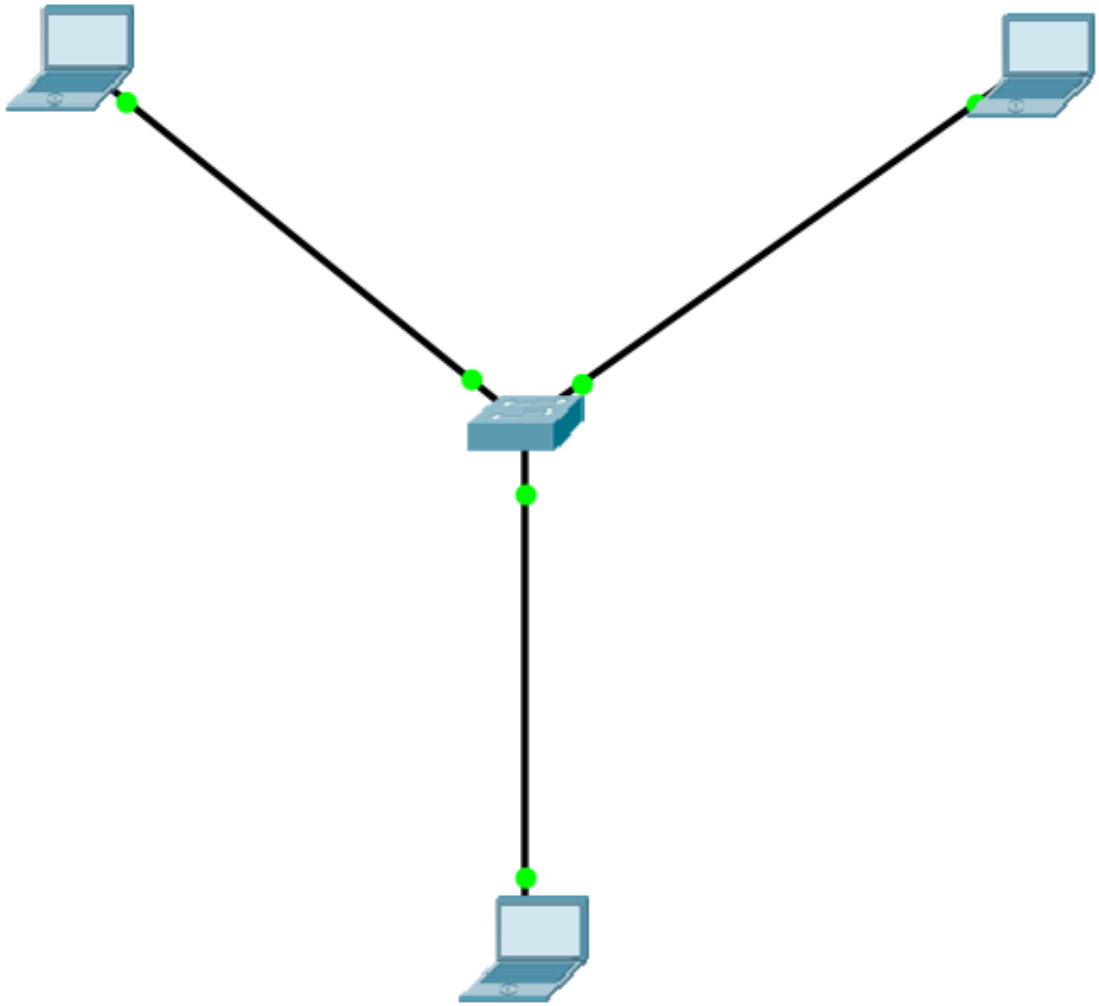
- Se pueden filtrar las tramas por el valor de cualquier campo que se muestran en el panel superior e intermedio.

- Pueden crearse a mano:



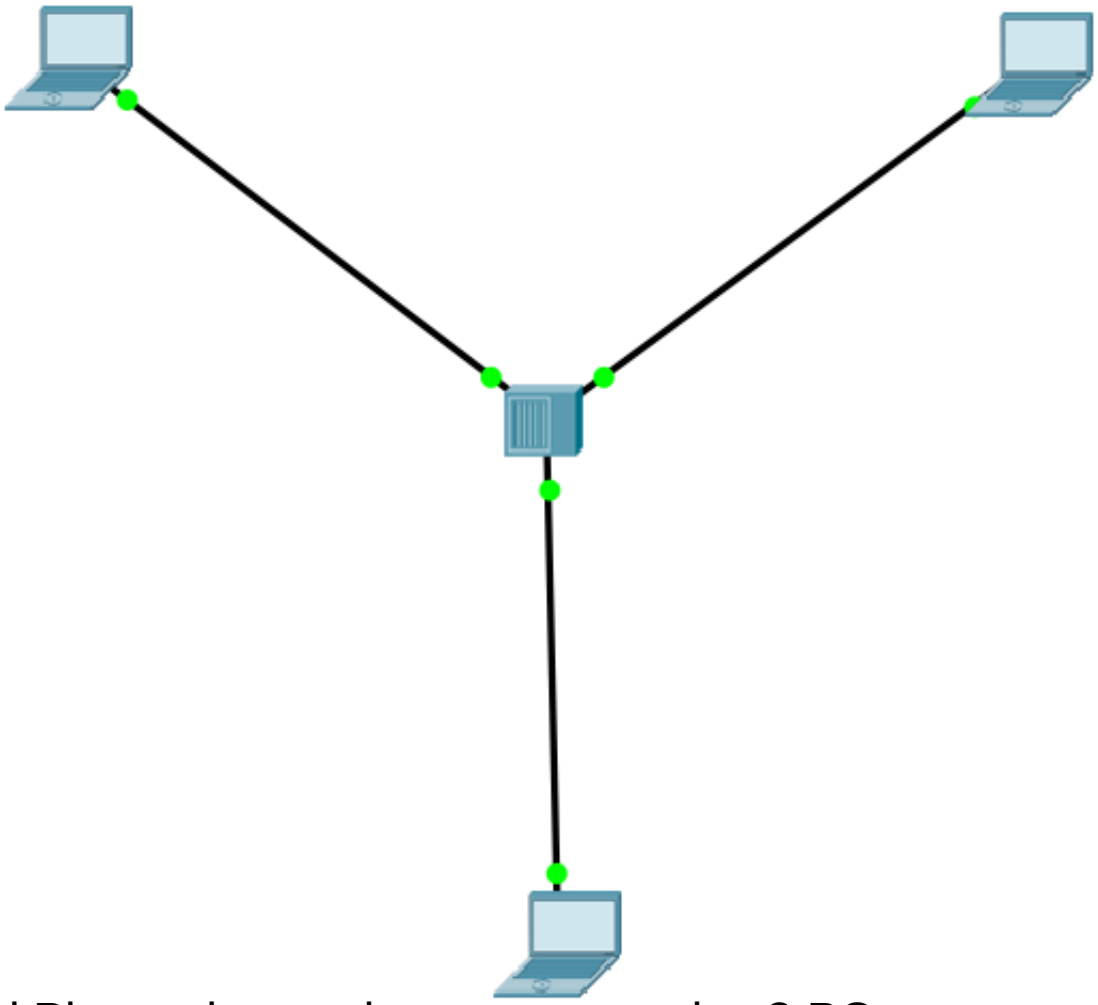
- Además, tres tipos de opciones filtro:
  - Aplicar Filtro
  - Preparar filtro
  - Filtro conversacional

Wireshark en Red con Switch



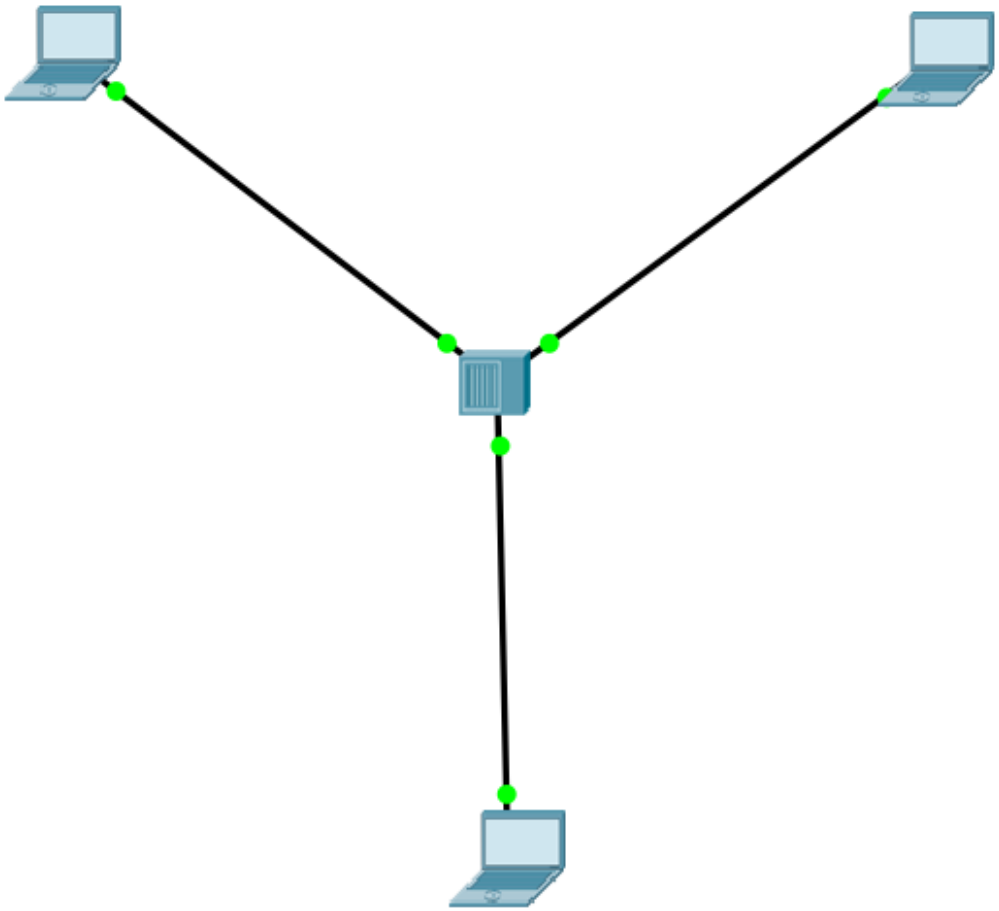
- Haz un Ping entre máquinas cualesquiera y observa lo que pasa en los 3 PCs.

Wireshark en Red con Hub



- Repite el Ping y observa lo que pasa en los 3 PCs.

Wireshark en Red con Hub



- Captura solo el tráfico de salida de la máquina en la que te encuentras.
- Captura solo el tráfico ICMP recibido desde uno de los dos vecinos.

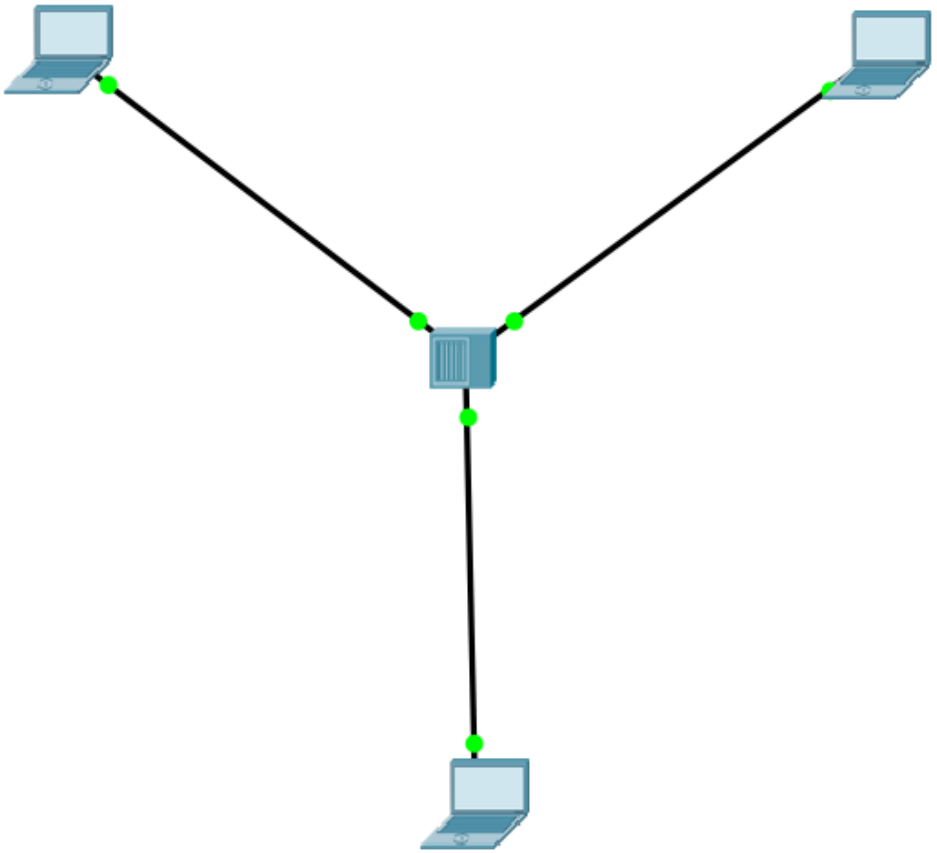
TCPDump

TCPDump

- Herramienta de captura de tráfico por línea de comandos de Linux:

  - **tcpdump -i any** Captura tráfico en todas las interfaces. Ver toda la red
  - **tcpdump -i *eth0*** Capturar en una interfaz exclusiva (**enp2s0 en Lubuntu**)
  - **tcpdump [host | src host | dst host] *Dir_IP*** Capturar el tráfico con origen o destino en dicha IP, solo si es origen o solo si es destino respectivamente
  - **tcpdump [port | src port | dst port] *80*** Capturar el tráfico con origen o destino en dicho puerto, solo si es origen o solo si es destino respectivamente
  - **tcpdump *proto*** Capturar tráfico en función del protocolo especificado (icmp, udp, tcp, http, etc.). Para concatenar con criterios de puerto o host usar **and**.
  - **tcpdump [< | <= | > | >=] *64*** Capturar paquetes en función del tamaño en bytes
  - **tcpdump port *23* -w *nombre_ficher*o** Salvar la captura a disco
  - **tcpdump –r *nombre_fichero*** Mostrar la captura desde disco

TCPDump en red con Hub



- Captura solo el tráfico de salida de la máquina en la que te encuentras.
- Captura solo el tráfico ICMP recibido desde uno de los dos vecinos.

TCPReplay

Tcpreplay

- Permite volcar a la red tráfico capturado previamente
- Resulta de gran utilidad para realizar pruebas de rendimiento sobre la red y de seguridad frente a ataques
- Se pueden modificar parámetros de los paquetes antes de ser emitidos con la herramienta TCPRewrite

TCPReplay

Tcpreplay

- TCPRewrite - Modo de uso (1):

```
lubuntu:~$ sudo tcprewrite --infile=A --outfile=B --srcipmap=C
--enet-smac=D --dstipmap=E --enet-dmac=F
```

A. Fichero de entrada con la captura inicial:

   Ejemplo.pcap

B. Fichero de salida con los cambios aplicados:

   EjemploRetocado.pcap

C. Modificación de la dirección IP de origen:

   Una IP concreta por otra → 172.16.2.4/32:172.16.2.7
   Todas las IPs por otra → 0.0.0.0/0:172.16.2.7

D. Modificación de la dirección MAC de origen:

   Una MAC concreta por otra → FA:BA:DA:AA:AA:AA,CA:FE:EE:EE:EE:EE
   Todas las MACs por otra → CA:FE:EE:EE:EE:EE

E. Modificación de la dirección IP de destino

F. Modificación de la dirección MAC de destino

TCPReplay

- TCPRewrite - Modo de uso (2):

Tcpreplay

En ocasiones puede resultar necesario modificar el checksum de los paquetes tras modificarlos, que sirve como firma de los mismos, para que el destino no los considere erróneos o corruptos

```
lubuntu:~$ sudo tcprewrite --infile=A --outfile=B --fixcsum
```

A. Fichero de entrada con la captura inicial:

   EjemploRetocado.pcap

B. Fichero de salida con los cambios aplicados:

   EjemploRetocadoCorregido.pcap

TCPReplay

- TCPReplay - Modo de uso (1):

Tcpreplay

```
lubuntu:~$  sudo tcpreplay OPCIONES --intf1=NombreInterfaz ejemploRetocado.pcap
```

- Opciones:

A. Número de veces que se desea repetir el envío del fichero de entrada:

   --loop=666 → 666 veces
   --loop=0 → infinitas veces hasta pulsar CTRL-C

B. Cargar en memoria RAM el fichero de entrada para recudir el tiempo de lectura respecto a hacerlo desde disco:

   --enable-file-cache

C. Configuración de la velocidad de envío:

   --mpbs=2.0 → Usa un ancho de banda de 2 MegaBytes por segundo
   --pps=300 → Envía 300 paquetes por segundo
   --topspeed → Envía al límite de la capacidad de la interfaz de red

TCPReplay

Tcpreplay

- Ejercicio básico:

1. Descarga el fichero VoIP.pcap del campus virtual. Dicho fichero se corresponde con una llamada VoIP real compuesta de 6127 paquetes.

2. Interconecta tu portátil con el de un compañero a través de un switch

3. Copia mediante USB el fichero descargado en uno de los portátiles, y usa TCPReplay para enviarlo hacia la red

4. Observa con Wireshark cómo se recibe el tráfico. ¿Cómo es posible este hecho si la IP y MAC de destino no coinciden con las del portátil receptor?

TCPReplay

Tcpreplay

- Ejercicio complejo:

1. Usa como fichero de entrada la captura con solo tráfico ICMP recibido desde un vecino. Da igual que sea hubiese sido hecha con Wireshark o TCPDump

2. Modifica las IPs de origen y de destino para que sean tu dirección IP y la de tu vecino respectivamente

3. Modifica las MACs de origen y de destino para que sean tu dirección MAC y la de tu vecino respectivamente

   Para saber la MAC de un PC, escribe **ifconfig** en el terminal y busca la dirección física (**HWaddr**)

4. Realiza un “DoS” sobre tu vecino usando el fichero resultante y la herramienta TCPreplay

5. Comprueba el éxito “del ataque” en el PC del vecino usando Wireshark

Scapy

- Herramienta de ciberseguridad, entre otro tipo de aplicaciones
- Permite generar tráfico y enviarlo a través de la red
- También permite construir sniffers de red personalizados.

Scapy

- sudo scapy
- Ejemplo: Envío de un Ping

Composición del mensaje en Scapy:

```
>>> object_Ethernet = Ether()
>>> object_IP = IP()
>>> object_IP.display()
>>> object_IP.dst="192.168.1.5"
>>> object_ICMP = ICMP()
>>> sendp(object_Ethernet/object_IP/object_ICMP)
```

- Observa en una captura de wireshark en el PC de tu compañero como llega el tráfico ICMP. Usa el filtro correspondiente.

Scapy

- Ejercicios simples. Genera los siguientes paquetes:

❑ Modifica la IP de origen para que el Ping no se envíe con la IP de tu portátil

❑ Modifica la MAC de origen para que no sea la de tu portátil

- Ejemplos:
  FA:BA:DA:AA:AA:AA
  CA:FE:EE:EE:EE:EE
  CE:BA:DA:AA:AA.AA
  AC:AB:AB:AA:AA:AA
  CA:BE:CE:AB:AA:AA

❑ Modifica la MAC de origen para que parezca que se generó desde un dispositivo Huawei

- MAC Address vendor Huawei en Google

Scapy

- Hacking mode: Envía un Ping de manera infinita

Composición del mensaje en Scapy:

>>> while (1<2):

… instrucción **(OJO A LA INDENTACIÓN)**

… *intro*

- Observa a través de wireshark en el PC de tu compañero el resultado
- Modifica el código anterior para que cada paquete se envíe con una dirección IP de origen diferente y aleatoria:
  - RandIP()
  - En lugar de un bucle while, usa la opción loop=1 de la función sendp: sendp(……..,loop=1)
- Observa el resultado en el PC de tu compañero a través de Wireshark

Scapy

- Construcción de un scanner de red:

```
#!/usr/bin/env python2
from scapy.all import *
from netaddr import *

ip_range = IPNetwork('192.168.1.0/24')

Crea capa Ethernet
Crea capa IP
Crea capa ICMP

for address in ip_range:
        ip.dst=str(address)
        envía el paquete
```

Scapy

- Modifica el script anterior para que reciba la dirección de red y la máscara desde la línea de comandos

```
import sys

if len(sys.argv) != ¿?:
    mensaje de error indicando los argumentos del script
    quit()

red=sys.argv[¿?]
mask=sys.argv[¿?]

ip_range = IPNetwork(red+'/'+mask)
```

Scapy

- Creación de un *sniffer* de red:

```
#!/usr/bin/env python2
from scapy.all import *
def paqueteRecibido(paquete):
        if paquete.haslayer(Ether):
                print paquete[Ether].dst

sniff  ( iface=”Nombre”,
        prn=”paqueteRecibido”,
        filter=”protocolo”,
        store=0)
```

- Pruébalo lanzando un Ping entre máquinas

Scapy

- Ejercicio complejo:

1. Crea un *sniffer* de red que espere solo por tráfico ICMP
2. Tras cada Ping recibido, se muestra un contador de Pings
3. Cuando la IP de origen sea igual a 192.168.1.66, habrá consecuencias
4. La consecuencia es lanzar un escaneo a la red 192.168.1.0 /28, aunque pueda especificarse una red y máscara diferentes desde línea de comandos
5. Muestra la IP de destino de cada paquete enviado por la consola