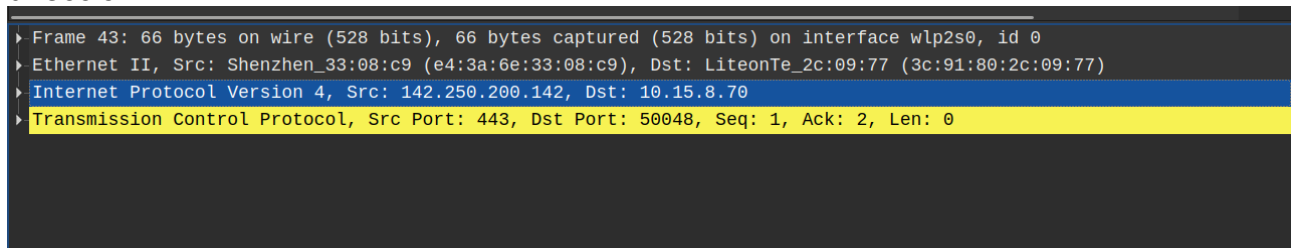


Ejercicio 1: Sistemas de direccionamiento (I).

a) Usando Wireshark, captura una pantalla de un paquete donde aparezcan en la misma imagen los siguientes seis campos a la vez, e indica la longitud en bytes y bits de cada dirección:



- MAC origen y destino: 6 bytes (o 48 bits) cada MAC
- IPv4 origen y destino: 4 bytes (o 32 bits) cada IPv4
- Puerto origen y destino: 2 bytes (o 16 bits) cada puerto

b) Indica cuál sería la dirección mínima y máxima posible de cada sistema de direccionamiento:

- Primera MAC: 00:00:00:00:00:00
- Última MAC: ff:ff:ff:ff:ff:ff
- Primera IPv4: 0.0.0.0
- Última IPv4: 255.255.255.255
- Primer n.º de puerto: 0
- Último n.º de puerto: 65535

c) Asocia cada sistema de direccionamiento (direcciones MAC, direcciones IP, puertos) con la razón adecuada:

Necesitamos **MAC** para distinguir cada elemento de una LAN

Necesitamos **puertos** para distinguir entre aplicaciones en un mismo equipo

Necesitamos **direcciones IP** para distinguir entre equipos tanto en la LAN como en la WAN

Ejercicio 2: Encapsulamiento en los niveles de red y transporte.

a) Escribe el tamaño en bytes típico de cada cabecera en estos dos paquetes:

Ethernet (14 bytes)	IPv4 (>=20 bytes)	TCP (>=20 bytes)	Aplicación
----------------------	--------------------	-------------------	------------

Ethernet (14 bytes)	IPv4 (>=20 bytes)	UDP (8 bytes)	Aplicación
----------------------	--------------------	----------------	------------

b) Aquí tienes el contenido en hexadecimal de un paquete que usa UDP en la capa de transporte (no aparece el checksum final):

84 9c a6 58 cf ae b8 76 3f 0b 24 57 08 00 45 00 00 44 78 79 00 00 80 11 e7 e4 c0 a8 00
c7 4a 7d ce 5e f6 9c 01 bb 00 30 21 21 0c 47 bf 88 cd 47 ec 25 3c 04 03 08 c5 62 10 1f b5
b7 72 e5 b8 7b 44 9d 99 f3 b8 09 3b 5f 21 97 00 12 77 27 47 e1 de 39

Colorea cada cabecera del paquete usando los siguientes colores de fondo: **enlace**, **red**, **transporte**, **aplicación**.

¿Cuántos bytes ocupa la capa de aplicación? 40 bytes

Ejercicio 3: Análisis de cabeceras IPv4.

a) Aquí tienes resaltadas las cabeceras IPv4 de 3 paquetes. Completa las tablas con los valores adecuados y en el formato correcto (por ejemplo, escribe un 0 ó 1 en el valor de los flags ya que cada flag es un bit, las IP en decimal, el checksum en hexadecimal, etc):

```
0000  b8 76 3f 0b 24 57 10 62 d0 8c a4 95 08 00 45 00  ·v?·$W·b·····E·
0010  01 31 a8 89 00 00 79 06 19 9c ac d9 11 14 c0 a8  ·1·····y·······
0020  00 0c 00 50 c0 e8 a3 13 77 58 ea 02 76 3a 50 18  ···P·····wX··v:P·
```

Campo	Longitud en bits	Valor
Versión	4 bits	4
Longitud cabecera	4 bits	5
DSCP	6 bits	0
ECN	2 bits	0
Longitud paquete	16 bits	01 31 = 305
Identificación	16 bits	a8 89
Flag DF	1 bits	0
Flag MF	1 bits	0
Offset	13 bits	0
TTL	8 bits	79 = 121
Siguiente Protocolo	8 bits	06 = TCP (6)
Checksum	16 bits	19 9c
IP origen	32 bits	172.217.17.20
IP destino	32 bits	192.168.0.12

```

0000  b8 76 3f 0b 24 57 10 62 d0 8c a4 95 08 00 45 00  ·v?·$W·b ······E·
0010  00 35 fa 9a 40 00 31 11 e0 bd b9 1e f4 8c c0 a8  ·5··@·1· ······
0020  00 0c 1b 47 37 2d 00 21 fc e0 01 00 13 19 67 5b  ··G7-·! ······g[

```

Campo	Longitud en bits	Valor
Versión	4 bits	4
Longitud cabecera	4 bits	5
DSCP	6 bits	0
ECN	2 bits	0
Longitud paquete	16 bits	00 35 = 53
Identificación	16 bits	fa 9a
Flag DF	1 bits	1
Flag MF	1 bits	0
Offset	13 bits	0
TTL	8 bits	31 = 49
Siguiente Protocolo	8 bits	11 = UDP (17)
Checksum	16 bits	e0 bd
IP origen	32 bits	185.30.244.140
IP destino	32 bits	192.168.0.12

```

0000  b8 76 3f 0b 24 57 10 62 d0 8c a4 95 08 00 45 20  ·v?·$W·b ······E·
0010  00 30 5a 8c 00 00 31 11 fa a8 31 90 42 24 c0 a8  ·0Z···1· ··1·8$··
0020  00 0c 58 05 37 2d 00 1c 29 db 21 00 3a 12 c7 41  ··X·7-· )·!·:···A·

```

Campo	Longitud en bits	Valor
Versión	4 bits	4
Longitud cabecera	4 bits	5
DSCP	6 bits	0010 00 = 8
ECN	2 bits	0 = 00
Longitud paquete	16 bits	00 30 = 48
Identificación	16 bits	5a 8c
Flag DF	1 bits	0
Flag MF	1 bits	0
Offset	13 bits	0
TTL	8 bits	31 = 49
Siguiente Protocolo	8 bits	11 = UDP (17)
Checksum	16 bits	fa a8
IP origen	32 bits	49.144.66.36
IP destino	32 bits	192.168.0.12

b) Ahora escribe el contenido completo en hexadecimal de cuatro cabeceras IPv4, sabiendo que cada una cumple los siguientes requisitos especificados. Si no se te proporciona el valor de alguno de los campos, usa el valor por defecto más común o valores aleatorios si es necesario:

- Paquete enviado por 25.46.98.124 (un servidor web) a 192.168.0.13, con una longitud total de 1000 bytes y un TTL de 80

Contenido cabecera: 45 00 03 e8 af 09 00 00 50 06 ef c7 19 2e 62 7c c0 a8 00 0d

- Paquete enviado por un servidor FTP cuya IP es 212.15.68.93 a 10.0.0.5, con longitud total de 400 bytes y un TTL de 50. Es el segundo y último fragmento de un paquete. El primer fragmento tenía una longitud de 700 bytes de datos

Contenido cabecera: 45 00 01 90 00 40 00 55 32 06 b5 04 d4 0f 44 5d 0a 00 00 05

- Paquete de un cliente HTTPS enviado desde 43.21.203.65 a 180.12.12.4 con un TTL de 5 y longitud de 400 bytes

Contenido cabecera: 45 00 01 90 d2 5a 00 00 05 06 5f a7 2b 15 cb 41 b4 0c 0c 04

- Paquete DNS que ha de ser entregado de manera inmediata desde 8.8.8.8 a tu ordenador 192.168.0.20, con un TTL de 19 y longitud total de 60 bytes. Este paquete debe reaccionar si encuentra congestión en la ruta

Contenido cabecera: 45 83 00 3c f9 a9 00 00 13 11 f2 c4 08 08 08 08 c0 a8 00 14

Puede serte útil rellenar una tabla para cada cabecera (como la del apartado anterior), y cuando tengas todos los campos, escribe entonces todo el contenido completo en hexadecimal, de manera que el resultado sea una secuencia de bytes.

Ejercicio 4: Práctica: Filtros IP en Wireshark.

Averigua primero los datos de tu conexión, adjuntando los pantallazos adecuados:

```

> ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp1s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
    link/ether e8:6a:64:d4:ce:4f brd ff:ff:ff:ff:ff:ff
3: wlp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 3c:91:80:2c:09:77 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.20/24 brd 192.168.0.255 scope global dynamic noprefixroute wlp2s0
        valid_lft 70505sec preferred_lft 70505sec
    inet6 fe80::621a:76c8:ecba:49fe/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
  
```

- Mi MAC es: 3c:91:80:2c:09:77
- Mi IPv4 es: 192.168.0.20
- Mi máscara es: 255.255.255.0
- Mi puerta de enlace es: 192.168.0.1

Con ayuda de los datos obtenidos, escribe el filtro Wireshark de cada apartado:

- Paquetes enviados a tu IP
ip.dst == 192.168.0.20
- Paquetes enviados por tu IP a tu puerta de enlace
ip.src == 192.168.0.20 && ip.dst == 192.168.0.1
- Paquetes que tengan parte opcional al final de la cabecera IP
ip.hdr_len > 20
- Paquetes cuya longitud total de la zona de IP esté entre 500 y 1000 bytes
ip.len >= 500 && ip.len <= 1000
- Paquetes con TTL mayor que 10
ip.ttl > 10
- Paquetes que no usen TCP en la capa de transporte
ip.proto != 6
- Paquetes que no puedan fragmentarse
ip.flags.df == 1
- Paquetes enviados por tu IP y tu MAC, con TTL menor que 20 y que usen UDP
ip.src == 192.168.0.20 &&
eth.src == 3c:91:80:2c:09:77 &&
ip.ttl < 20 &&
ip.proto == 17

Ejercicio 5: Fragmentación en IP.

a) Supón que quieres enviar un paquete de 5000 bytes de datos a través de una red Ethernet. Indica, para cada fragmento, el valor de los siguientes campos de las cabeceras:

Fragmento n.º 1 : Identificador= 25, MF= 1, offset= 0
 Fragmento n.º 2 : Identificador= 25, MF= 1, offset= 185
 Fragmento n.º 3 : Identificador= 25, MF= 1, offset= 370
 Fragmento n.º 4 : Identificador= 25, MF= 0, offset= 555

(Utiliza tantos fragmentos como necesites. Como identificador de todos, puedes usar un número aleatorio)

b) En una red se capturan los siguientes paquetes, correspondientes a los dos últimos fragmentos de una transmisión. Solo se muestran los datos necesarios de su cabecera IPv4:

- Penúltimo: Identificador=1352, MF=1, offset=375, longitud total=620
- Último: Identificador=1352, MF=0, offset=450, longitud total=120

Averigua:

- La MTU de la red: 620 bytes
- El nº total de fragmentos en los que se dividió el paquete inicial: 7 fragmentos
- La longitud de los datos del paquete inicial sin fragmentar: 3700 bytes

Ejercicio 8: Sistemas de direccionamiento (II).

Completa la siguiente tabla teniendo en cuenta que:

- La MAC origen/destino puede ser: “mi MAC”, “la MAC del router”, “la MAC del servidor” o “difusión”
- La IP origen/destino puede ser: “mi IP”, “la IP del router”, “la IP del servidor” o “difusión”
- El puerto origen/destino puede ser: “aleatorio” o un número concreto (consultar tabla de puertos en el PDF o las diapositivas de la unidad)

Colorea en amarillo las filas que usan TCP en el nivel de transporte y en verde las que usan UDP.

	MAC origen	MAC destino	IP origen	IP destino	Puerto origen	Puerto destino
Petición para ver una web externa	mi MAC	la MAC del router	mi IP	la IP de servidor	39008 aleatorio	443 o 80
Petición para usar el servidor FTP de la LAN	mi MAC	la MAC del servidor	mi IP	la IP del servidor	4563 aleatorio	21
Respuesta de un servidor DNS externo	la MAC del router	mi MAC	la IP del servidor	mi IP	53	55848 aleatorio
Respuesta del servidor web de Google	la MAC del router	mi MAC	la IP del servidor	mi IP	443 o 80	47108 aleatorio
Petición para obtener una IP	mi MAC	difusión	0.0.0.0	difusión	68	67

Ejercicio 9: Análisis de cabeceras TCP y UDP.

a) Aquí tienes una serie de paquetes en los que aparece resaltada la cabecera UDP en cada uno de ellos. Para cada paquete, indica si se trata de una petición o una respuesta, e indica también de qué servicio (HTTPS, DNS, etc).

```

0000 10 62 d0 8c a4 95 b8 76 3f 0b 24 57 08 00 45 00
0010 00 3c 04 b6 00 00 80 11 cd 9b c0 a8 00 0c d4 a6
0020 d3 04 e9 31 00 35 00 28 4c 32 ce 01 01 00 00 01
0030 00 00 00 00 00 00 03 77 77 77 06 65 6c 70 61 69
0040 73 03 63 6f 6d 00 00 01 00 01

```

- Es una petición del servicio DNS

```

0000 ff ff ff ff ff ff b8 76 3f 0b 24 57 08 00 45 00
0010 01 48 14 12 00 00 80 11 25 94 00 00 00 00 ff ff
0020 ff ff 00 44 00 43 01 34 c8 e6 01 01 06 00 ec ef
0030 f5 84 00 00 00 00 00 00 00 00 00 00 00 00 00
0040 00 00 00 00 00 00 b8 76 3f 0b 24 57 00 00 00 00
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

- Es una petición del servicio DHCP

```

0000 b8 76 3f 0b 24 57 10 62 d0 8c a4 95 08 00 45 00
0010 00 74 04 ce 00 00 fc 11 51 4b d4 a6 d3 04 c0 a8
0020 00 0c 00 35 e7 b4 00 60 23 f6 ef 1a 81 80 00 01
0030 00 03 00 00 00 00 08 70 72 69 73 61 63 6f 6d 02
0040 73 63 06 6f 6d 74 72 64 63 03 6e 65 74 00 00 01
0050 00 01 c0 0c 00 01 00 01 00 00 00 8e 00 04 6c 80
0060 82 e0 c0 0c 00 01 00 01 00 00 00 8e 00 04 34 31
0070 64 bd c0 0c 00 01 00 01 00 00 00 8e 00 04 34 1f
0080 be 3a

```

- Es una respuesta del servicio DNS

b) Aquí tienes resaltadas las cabeceras TCP de 3 paquetes. Completa las tablas con los valores adecuados y en el formato correcto (por ejemplo, escribe un 0 ó 1 en el valor de los flags ya que cada flag es un bit, escribe los puertos en decimal, el checksum en hexadecimal, etc):

```

0000 10 62 d0 8c a4 95 b8 76 3f 0b 24 57 08 00 45 00
0010 00 28 b5 d3 40 00 80 06 72 a9 c0 a8 00 0c 0d 6b
0020 04 34 cb e8 00 50 29 bb 1e e3 32 d9 5d 30 50 10
0030 02 05 36 9c 00 00

```

Campo	Longitud en bits	Valor
Puerto origen	16	52200 = cb e8
Puerto destino	16	80 = 00 50
Flag ACK	1	1
Flag SYN	1	0
Checksum	16	36 9c

```

0000  b8 76 3f 0b 24 57 10 62 d0 8c a4 95 08 00 45 00
0010  00 ff ab 51 40 00 32 06 25 f3 a2 7d 13 83 c0 a8
0020  00 0c 01 bb cb df f7 47 7f ed 95 d1 e7 94 50 18
0030  ae 60 ea 3e 00 00 17 03 03 00 d2 00 00 00 00 00
0040  00 00 02 79 e2 67 73 f0 6b 33 1a 1f 0b 55 20 c5

```

Campo	Longitud en bits	Valor
Puerto origen	16	443 = 01 bb
Puerto destino	16	52191 = cb df
Flag ACK	1	1
Flag SYN	1	0
Checksum	16	ea 3e

```

0000  b8 76 3f 0b 24 57 10 62 d0 8c a4 95 08 00 45 00
0010  00 34 80 75 40 00 77 06 b0 fb 0d 6b 04 34 c0 a8
0020  00 0c 00 50 cb e8 32 d9 5d 2f 29 bb 1e e3 80 12
0030  ff ff f7 e1 00 00 02 04 05 a0 01 03 03 08 01 01
0040  04 02

```

Campo	Longitud en bits	Valor
Puerto origen	16	80 = 00 50
Puerto destino	16	52200 = cb e8
Flag ACK	1	1
Flag SYN	1	1
Checksum	16	f7 e1

Ejercicio 10: Práctica: Filtros TCP y UDP en Wireshark.

a) Escribe el filtro Wireshark para obtener...

- Respuestas DNS enviadas por tu ordenador
udp.srcport == 53 && ip.src == 192.168.0.20
- Peticiones HTTPS enviadas por tu ordenador
tcp.dstport == 443 && ip.src == 192.168.0.20
- Paquetes SYN+ACK enviados por cualquier servidor web HTTPS
tcp.flags.syn == 1 && tcp.flags.ack == 1 && tcp.srcport == 443
- Respuestas DHCP enviadas por tu servidor DHCP (tu router)
ip.src == 192.168.0.1 && udp.srcport == 67
- Paquetes FTP (tanto preguntas como respuestas) con TTL mayor que 10
tcp.port == 21 && ip.ttl > 10

b) Captura paquetes y guarda en un fichero de Wireshark únicamente estos 4 paquetes:

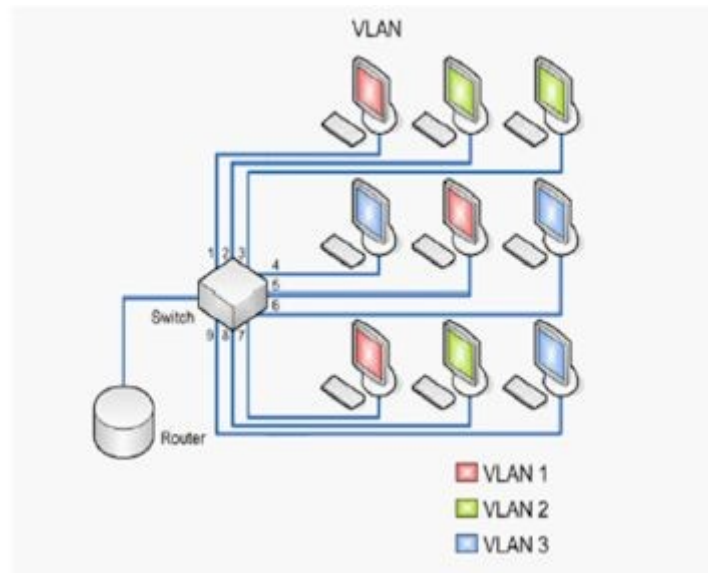
- Un paquete SYN
- Un paquete SYN+ACK
- Una petición DNS
- Una respuesta DNS

➔ Adjunta el fichero de Wireshark a tu entrega

Ejercicio 11: VLAN.

Una VLAN (Virtual LAN) es una manera de crear grupos de dispositivos en una LAN, aunque físicamente no estén en la misma área de trabajo. De ahí el nombre de redes “virtuales”, ya que están formadas por equipos independientemente de su posición en la LAN.

Por ejemplo, en el siguiente esquema puedes ver una red con 9 equipos, todos conectados al mismo switch, pero divididos en 3 VLAN diferentes dependiendo del color:



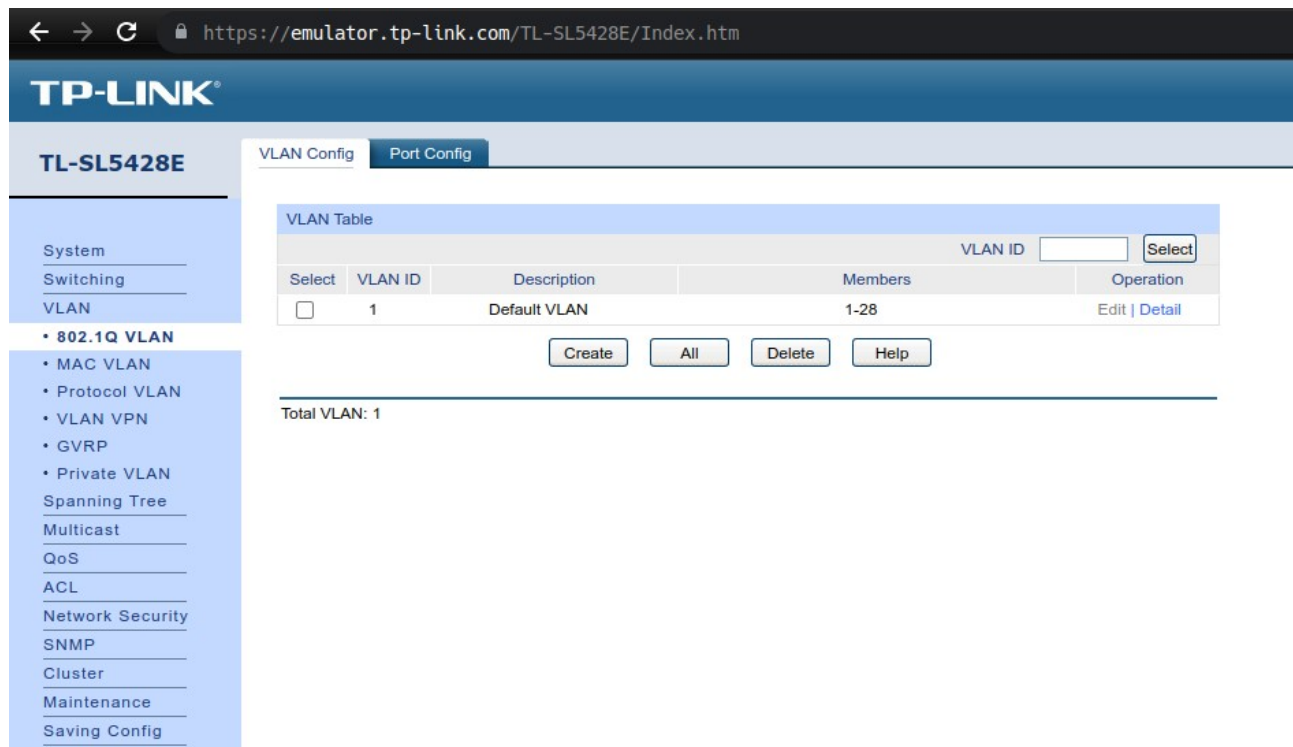
De esta forma, podemos organizar los equipos en grupos y, por ejemplo, aplicarles reglas de seguridad diferentes sin pensar dónde están situados. Los equipos pertenecientes a una VLAN creerán que están juntos, y separados de los equipos de otra VLAN (como si cada grupo tuviera un switch dedicado). Además de ayudar a gestionar la seguridad y simplificar la administración de la red, minimiza las molestias producidas por el tráfico por difusión. Sin VLAN, cuando un equipo envía un paquete por broadcast, llega a todos los demás equipos. Con VLAN, la difusión solo llega a los equipos de su mismo grupo. Es una manera de establecer dominios de difusión, que propagan los paquetes de broadcast solo por algunos puertos del switch, no por todos.

La creación de varias redes virtuales en los equipos conectados al mismo switch de una LAN ha de hacerse accediendo a la configuración del switch. Para ello, basta con definir cada grupo e indicar los n.º de los puertos del switch donde se conectan los equipos de cada grupo o VLAN.

a) Accede a la configuración del switch de la unidad anterior (ejercicio 6 de la unidad 5) y busca la sección para configurar una VLAN. Aparece ya creada una VLAN por defecto.

- ¿Cómo se llama la VLAN ya creada? Default VLAN
- ¿Cuál es su número identificador? 1
- ¿Qué puertos del switch pertenecen a esa VLAN? 2, 5, 9, 11, 13, 15, 16, 18 y 20

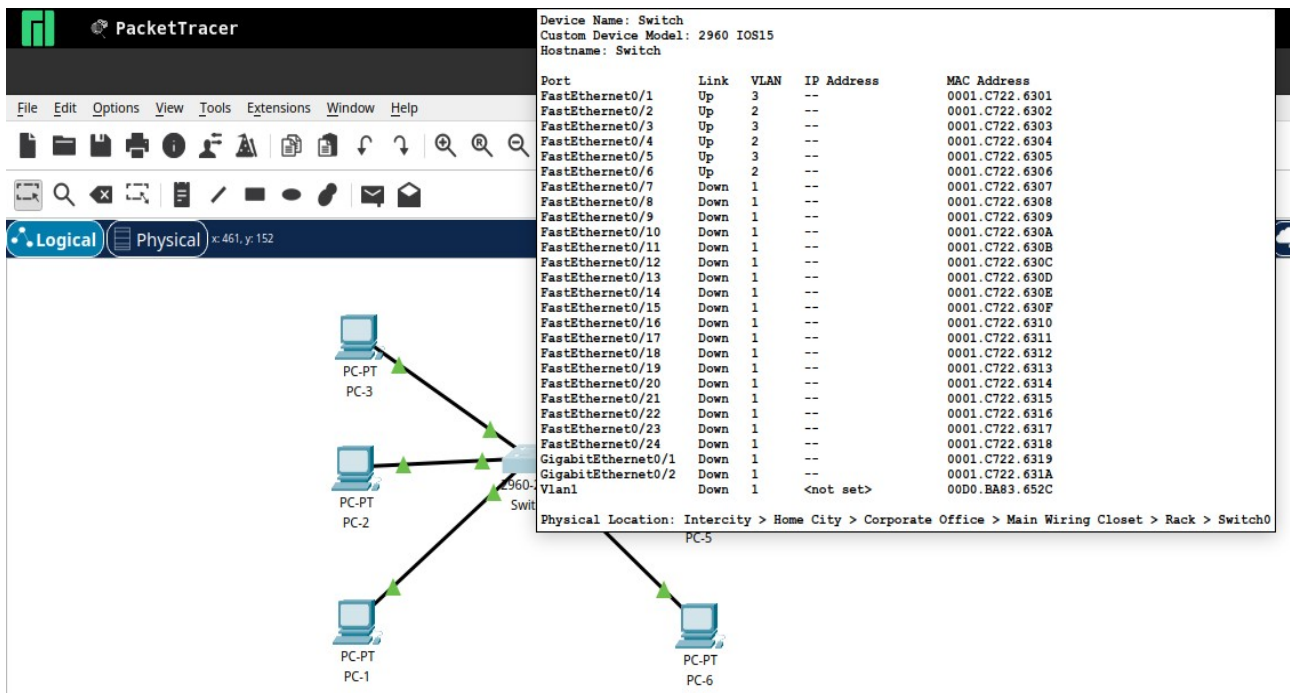
Adjunta una captura de la pantalla donde aparezca la configuración de VLAN en el switch.



b) Para crear una VLAN en PT, sigue los siguientes pasos:

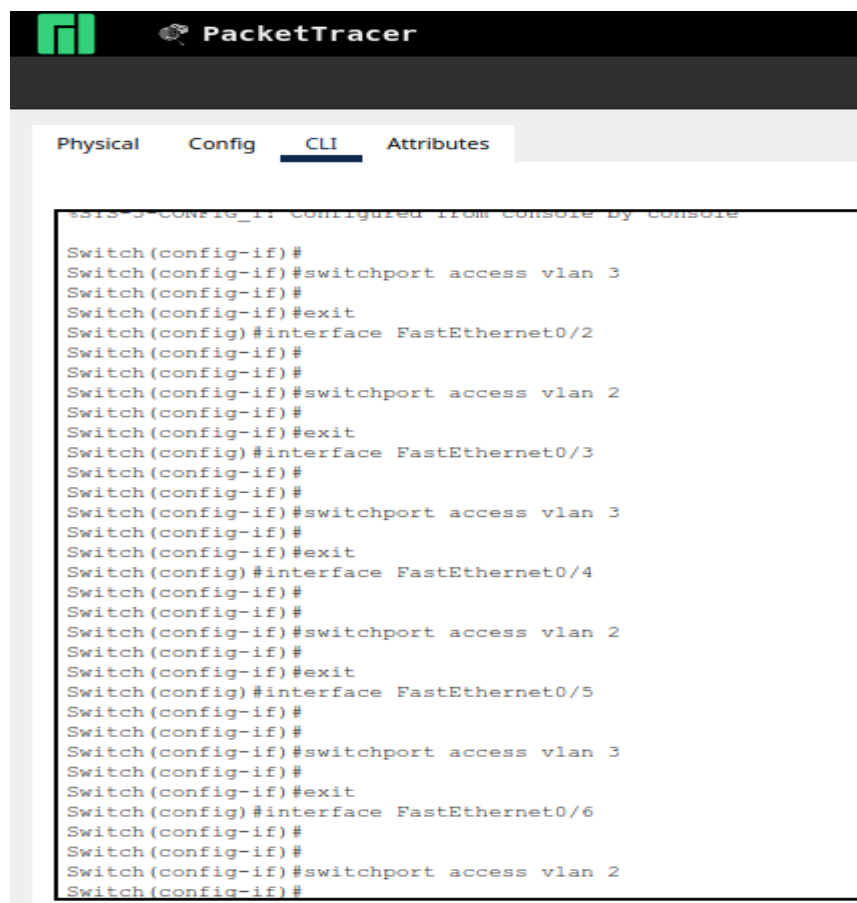
- Crea una LAN con un switch 2960 y 6 PC conectados al switch
- Asigna a los PC las IP estáticas de la 192.168.0.1 a la 6, y la misma máscara (255.255.255.0) en todos
- Ve a la configuración del switch, Config, VLAN Database
- Por defecto, aparecen ya creadas varias VLAN. Crea dos VLAN nuevas: una con n.º identificador 2 y nombre "Pares" y otra con n.º identificador 3 y nombre "Impares"
- Aún en el switch, ve en Config a cada uno de sus puertos (FastEthernet0/1, FastEthernet0/2, etc) y asigna la VLAN 2 a los que tienen IP par y la VLAN 3 a los que tienen IP impar

Adjunta una captura de pantalla donde aparezca la distribución de los equipos en las dos VLAN. Para ello, cuando termines todos los pasos, deja el ratón quieto sobre el switch y aparecerá una tabla con la VLAN asociada a cada puerto.



Port	Link	VLAN	IP Address	MAC Address
FastEthernet0/1	Up	3	--	0001.C722.6301
FastEthernet0/2	Up	2	--	0001.C722.6302
FastEthernet0/3	Up	3	--	0001.C722.6303
FastEthernet0/4	Up	2	--	0001.C722.6304
FastEthernet0/5	Up	3	--	0001.C722.6305
FastEthernet0/6	Up	2	--	0001.C722.6306
FastEthernet0/7	Down	1	--	0001.C722.6307
FastEthernet0/8	Down	1	--	0001.C722.6308
FastEthernet0/9	Down	1	--	0001.C722.6309
FastEthernet0/10	Down	1	--	0001.C722.630A
FastEthernet0/11	Down	1	--	0001.C722.630B
FastEthernet0/12	Down	1	--	0001.C722.630C
FastEthernet0/13	Down	1	--	0001.C722.630D
FastEthernet0/14	Down	1	--	0001.C722.630E
FastEthernet0/15	Down	1	--	0001.C722.630F
FastEthernet0/16	Down	1	--	0001.C722.6310
FastEthernet0/17	Down	1	--	0001.C722.6311
FastEthernet0/18	Down	1	--	0001.C722.6312
FastEthernet0/19	Down	1	--	0001.C722.6313
FastEthernet0/20	Down	1	--	0001.C722.6314
FastEthernet0/21	Down	1	--	0001.C722.6315
FastEthernet0/22	Down	1	--	0001.C722.6316
FastEthernet0/23	Down	1	--	0001.C722.6317
FastEthernet0/24	Down	1	--	0001.C722.6318
GigabitEthernet0/1	Down	1	--	0001.C722.6319
GigabitEthernet0/2	Down	1	--	0001.C722.631A
Vlan1	Down	1	<not set>	00D0.BA83.652C

Adjunta también otra captura de pantalla con los comandos CLI que se han usado internamente para configurar las VLAN del switch.



```

Switch(config-if) #
Switch(config-if) #switchport access vlan 3
Switch(config-if) #
Switch(config-if) #exit
Switch(config) #interface FastEthernet0/2
Switch(config-if) #
Switch(config-if) #
Switch(config-if) #switchport access vlan 2
Switch(config-if) #
Switch(config-if) #exit
Switch(config) #interface FastEthernet0/3
Switch(config-if) #
Switch(config-if) #
Switch(config-if) #switchport access vlan 3
Switch(config-if) #
Switch(config-if) #exit
Switch(config) #interface FastEthernet0/4
Switch(config-if) #
Switch(config-if) #
Switch(config-if) #switchport access vlan 2
Switch(config-if) #
Switch(config-if) #exit
Switch(config) #interface FastEthernet0/5
Switch(config-if) #
Switch(config-if) #
Switch(config-if) #switchport access vlan 3
Switch(config-if) #
Switch(config-if) #exit
Switch(config) #interface FastEthernet0/6
Switch(config-if) #
Switch(config-if) #
Switch(config-if) #switchport access vlan 2
Switch(config-if) #

```