Universidad Nacional Autónoma de Nicaragua

UNAN-León

Facultad de Ciencias y Tecnología

Departamento de Computación

Ingeniería en Telemática

V año

Componente: Laboratorio de Redes de Area Extensa

Tema: Practica de MPLS

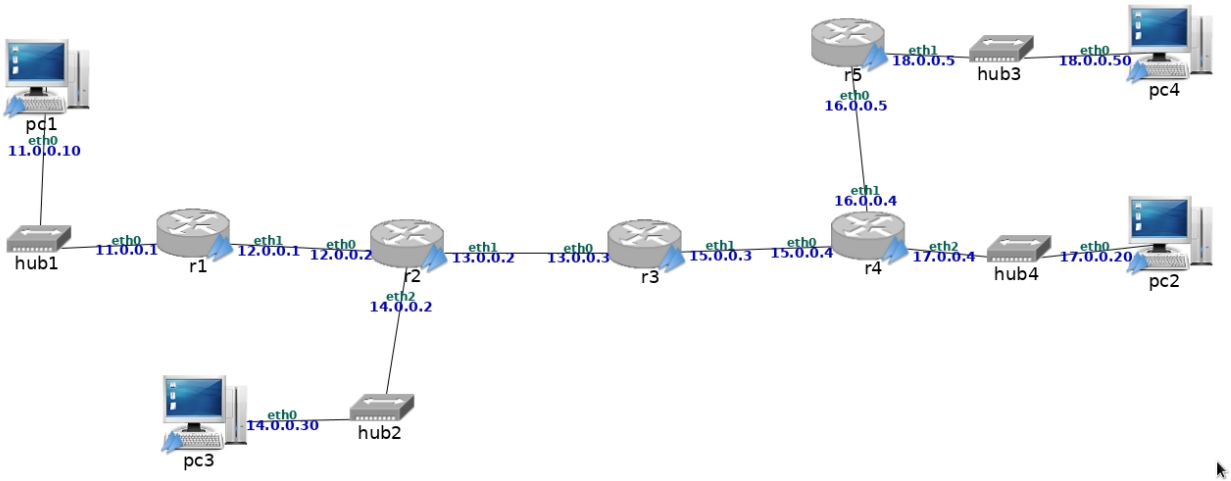
Realizado por:

Br. Jhonatan Uziel Espinoza Ortega Carnet: 15-00737-0

Dirigido a:

MSc. Aldo Martinez

León, Nicaragua lunes 8 de julio del 2019.



1. Comunicación entre pc1 y pc2 a través de MPLS

Comprueba que no funciona un ping desde pc1 a pc2, ya que los routers no tienen configurada ninguna ruta, salvo a las subredes a las que están directamente conectados.

```
pc1
pc1:~# ping 17.0.0.20
PING 17.0.0.20 (17.0.0.20) 56(84) bytes of data.
From 11.0.0.1 icmp_seq=1 Destination Net Unreachable
From 11.0.0.1 icmp_seq=2 Destination Net Unreachable
From 11.0.0.1 icmp_seq=3 Destination Net Unreachable
From 11.0.0.1 icmp_seq=4 Destination Net Unreachable
From 11.0.0.1 icmp_seq=5 Destination Net Unreachable
^C
--- 17.0.0.20 ping statistics ---
5 packets transmitted, 0 received, +5 errors, 100% packet loss, time 4035ms

pc1:~#
```

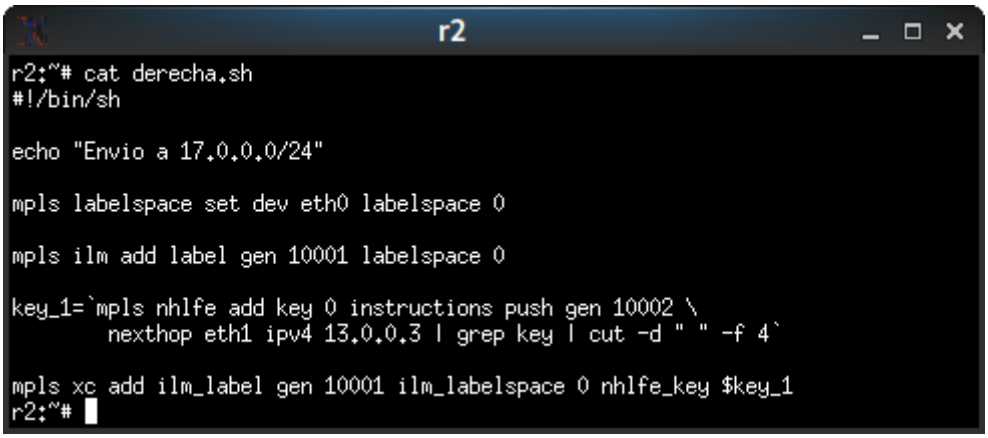
```
pc2
pc2:~# ping 11.0.0.10
PING 11.0.0.10 (11.0.0.10) 56(84) bytes of data.
From 17.0.0.4 icmp_seq=1 Destination Net Unreachable
From 17.0.0.4 icmp_seq=2 Destination Net Unreachable
From 17.0.0.4 icmp_seq=3 Destination Net Unreachable
From 17.0.0.4 icmp_seq=4 Destination Net Unreachable
From 17.0.0.4 icmp_seq=5 Destination Net Unreachable
^C
--- 11.0.0.10 ping statistics ---
5 packets transmitted, 0 received, +5 errors, 100% packet loss, time 4020ms

pc2:~#
```

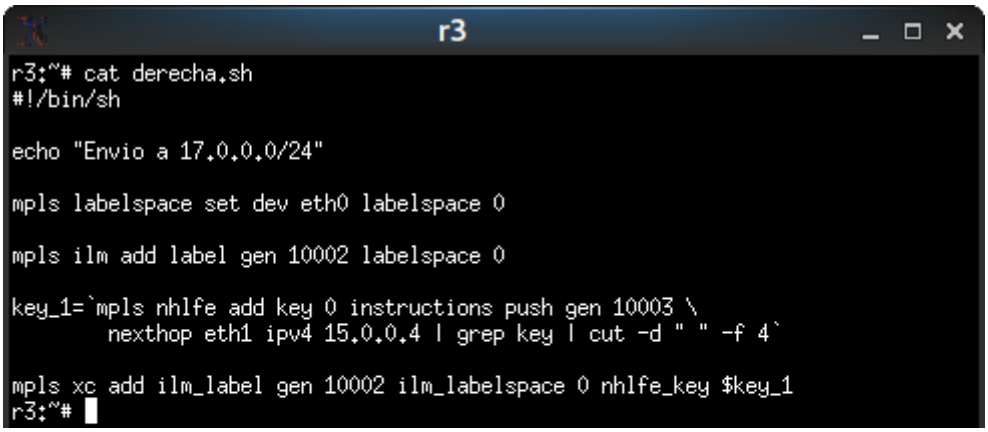
1.1 Configura los scripts MPLS para que pc1 y pc2 puedan intercambiarse tráfico utilizando una red MPLS a través de r1 - r2 - r3 - r4. En cada salto MPLS, los routers utilizarán una etiqueta diferente a la recibida. Todas las etiquetas de un router pertenecerán al mismo ámbito (labelspace=0).

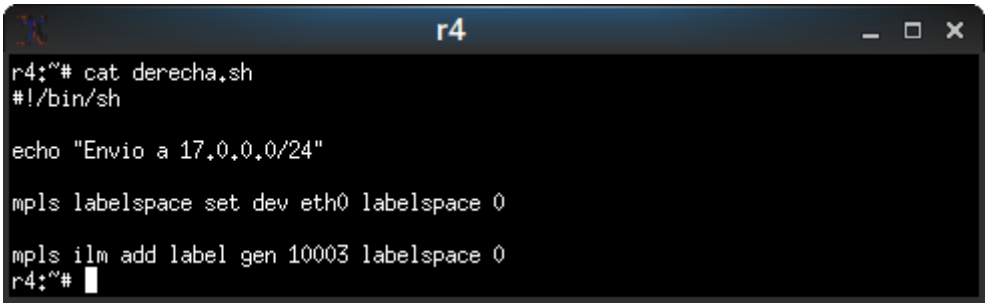
Incluye los scripts en la memoria, explicando los números de etiquetas que has utilizado en cada interfaz y sentido de la comunicación.

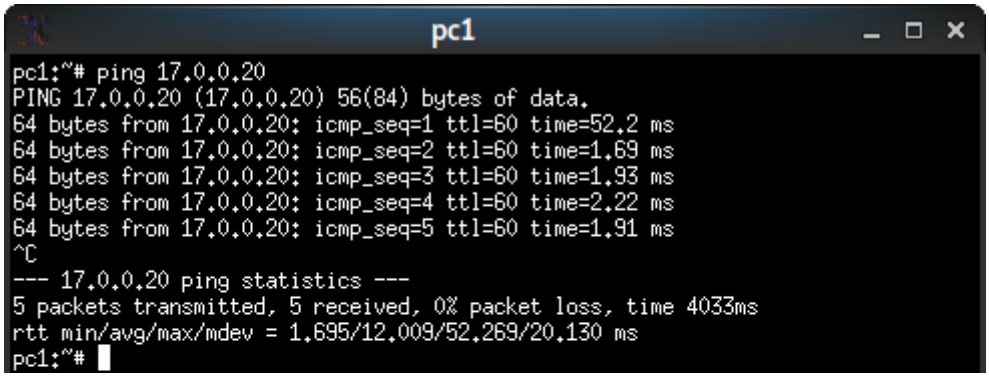
De PC1 a PC2

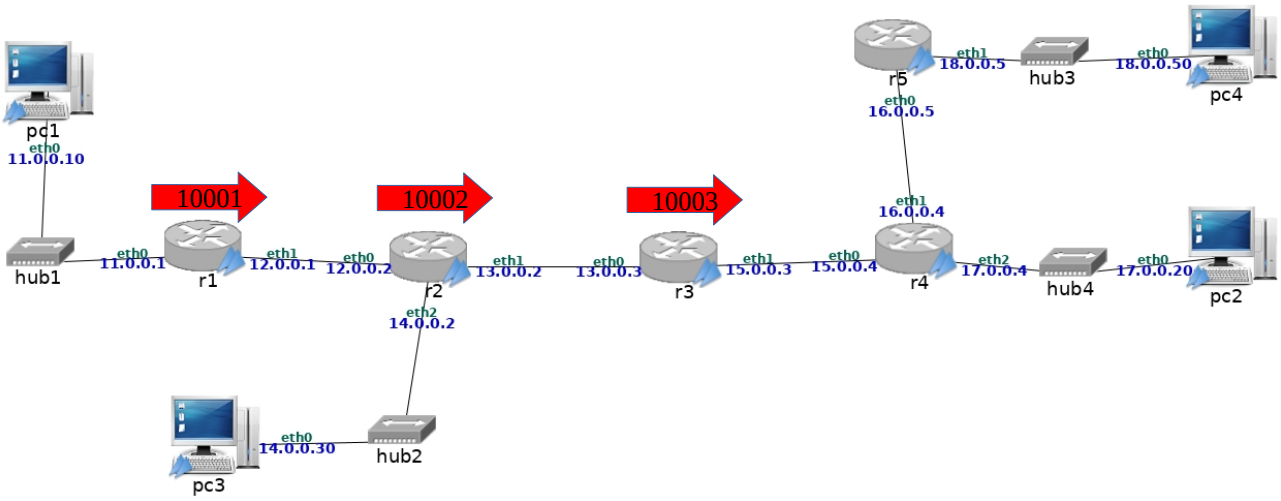
```
r1
r1:~# cat derecha.sh
#!/bin/sh

echo "Envio a 17.0.0.0/24"
key_1=`mpls nhlfe add key 0 instructions push gen 10001 \
        nexthop eth1 ipv4 12.0.0.2 | grep key | cut -d " " -f 4`

ip route add 17.0.0.0/24 via 12.0.0.2 mpls $key_1
r1:~#
```


r2
r2:~# cat derecha.sh
#!/bin/sh
echo "Envio a 17.0.0.0/24"
mpls labelspace set dev eth0 labelspace 0
mpls ilm add label gen 10001 labelspace 0
key_1=`mpls nhlfe add key 0 instructions push gen 10002 \
nexthop eth1 ipv4 13.0.0.3 | grep key | cut -d " " -f 4`
mpls xc add ilm_label gen 10001 ilm_labelspace 0 nhlfe_key $key_1
r2:~#


r3
r3:~# cat derecha.sh
#!/bin/sh
echo "Envio a 17.0.0.0/24"
mpls labelspace set dev eth0 labelspace 0
mpls ilm add label gen 10002 labelspace 0
key_1=`mpls nhlfe add key 0 instructions push gen 10003 \
nexthop eth1 ipv4 15.0.0.4 | grep key | cut -d " " -f 4`
mpls xc add ilm_label gen 10002 ilm_labelspace 0 nhlfe_key $key_1
r3:~#


r4
r4:~# cat derecha.sh
#!/bin/sh
echo "Envio a 17.0.0.0/24"
mpls labelspace set dev eth0 labelspace 0
mpls ilm add label gen 10003 labelspace 0
r4:~#


pc1
pc1:~# ping 17.0.0.20
PING 17.0.0.20 (17.0.0.20) 56(84) bytes of data.
64 bytes from 17.0.0.20: icmp_seq=1 ttl=60 time=52.2 ms
64 bytes from 17.0.0.20: icmp_seq=2 ttl=60 time=1.69 ms
64 bytes from 17.0.0.20: icmp_seq=3 ttl=60 time=1.93 ms
64 bytes from 17.0.0.20: icmp_seq=4 ttl=60 time=2.22 ms
64 bytes from 17.0.0.20: icmp_seq=5 ttl=60 time=1.91 ms
^C
--- 17.0.0.20 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4033ms
rtt min/avg/max/mdev = 1.695/12.009/52.269/20.130 ms
pc1:~#



De PC2 a PC1

r4

```
r4:~# cat izquierda.sh
#!/bin/sh

echo "Envio a 11.0.0.0/24"

key_2=`mpls nhlfe add key 0 instructions push gen 10011 \
       nexthop eth0 ipv4 15.0.0.3 | grep key | cut -d " " -f 4`

ip route add 11.0.0.0/24 via 15.0.0.3 mpls $key_2
r4:~#
```

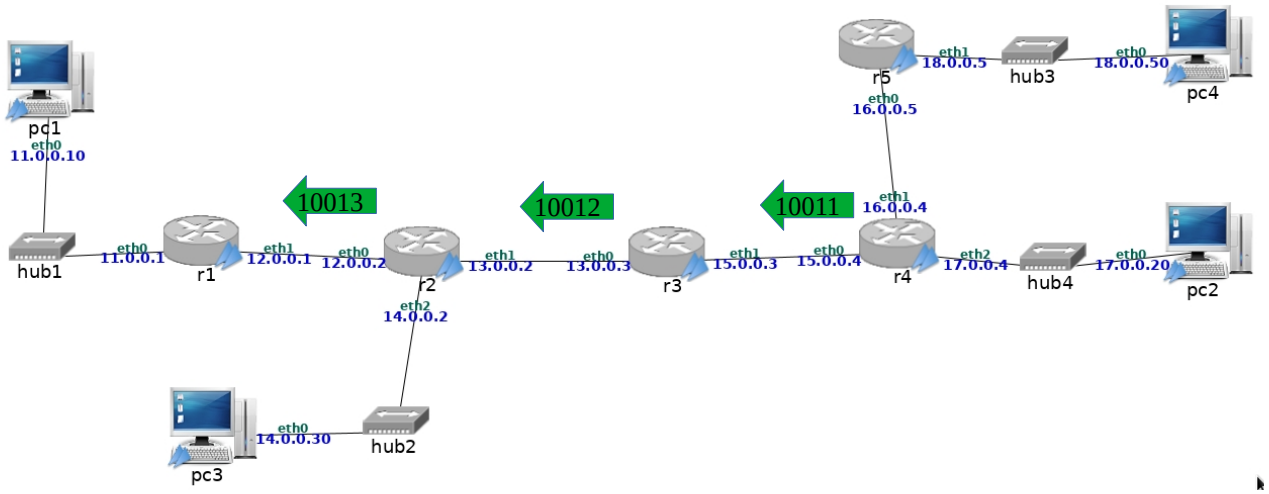
r3

```
r3:~# cat izquierda.sh
#!/bin/sh

echo "Envio a 11.0.0.0/24"

mpls labelspace set dev eth1 labelspace 0

mpls ilm add label gen 10011 labelspace 0

key_2=`mpls nhlfe add key 0 instructions push gen 10012 \
       nexthop eth0 ipv4 13.0.0.2 | grep key | cut -d " " -f 4`

mpls xc add ilm_label gen 10011 ilm_labelspace 0 nhlfe_key $key_2
r3:~#
```

r2

```
r2:~# cat izquierda.sh
#!/bin/sh

echo "Envio a 11.0.0.0/24"

mpls labelspace set dev eth1 labelspace 0

mpls ilm add label gen 10012 labelspace 0

key_2=`mpls nhlfe add key 0 instructions push gen 10013 \
       nexthop eth0 ipv4 12.0.0.1 | grep key | cut -d " " -f 4`

mpls xc add ilm_label gen 10012 ilm_labelspace 0 nhlfe_key $key_2
r2:~#
```

```
r1
r1:~# cat izquierda.sh
#!/bin/sh

echo "Envio a 11.0.0.0/24"

mpls labelspace set dev eth1 labelspace 0

mpls ilm add label gen 10013 labelspace 0

r1:~#
```

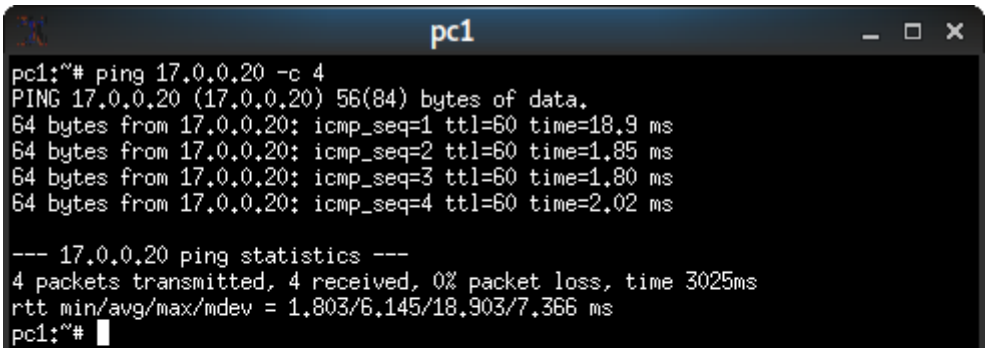


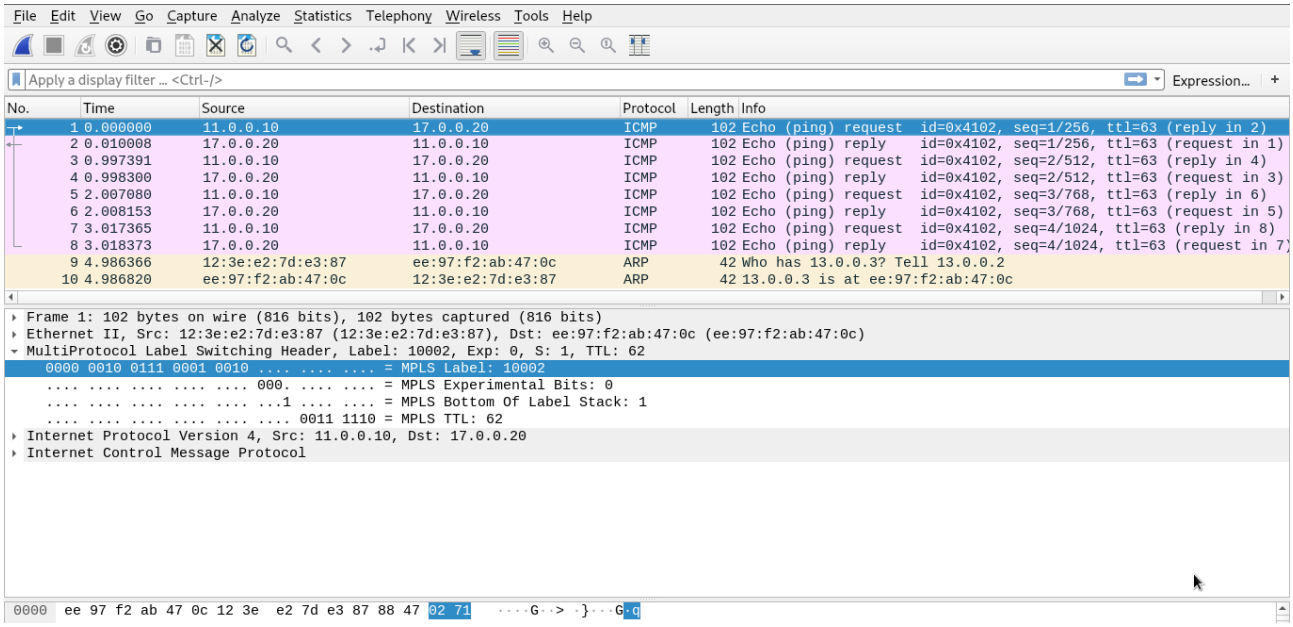

```
pc2
pc2:~# ping 11.0.0.10
PING 11.0.0.10 (11.0.0.10) 56(84) bytes of data.
64 bytes from 11.0.0.10: icmp_seq=1 ttl=60 time=22.1 ms
64 bytes from 11.0.0.10: icmp_seq=2 ttl=60 time=1.82 ms
64 bytes from 11.0.0.10: icmp_seq=3 ttl=60 time=1.64 ms
64 bytes from 11.0.0.10: icmp_seq=4 ttl=60 time=2.07 ms
64 bytes from 11.0.0.10: icmp_seq=5 ttl=60 time=1.83 ms
^C
--- 11.0.0.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4037ms
rtt min/avg/max/mdev = 1.640/5.907/22.165/8.130 ms
pc2:~#
```

1.2 Realiza una captura de tráfico en la 13.0.0.0/24 (mpls-01.cap) y en la 15.0.0.0/24 (mpls-02.cap) mientras ejecutas un ping desde pc1 -c 4 a pc2. Explica las cabeceras de los paquetes capturados en mpls-01.cap e indica en qué paquetes se convierten en la captura mpls-02.cap.

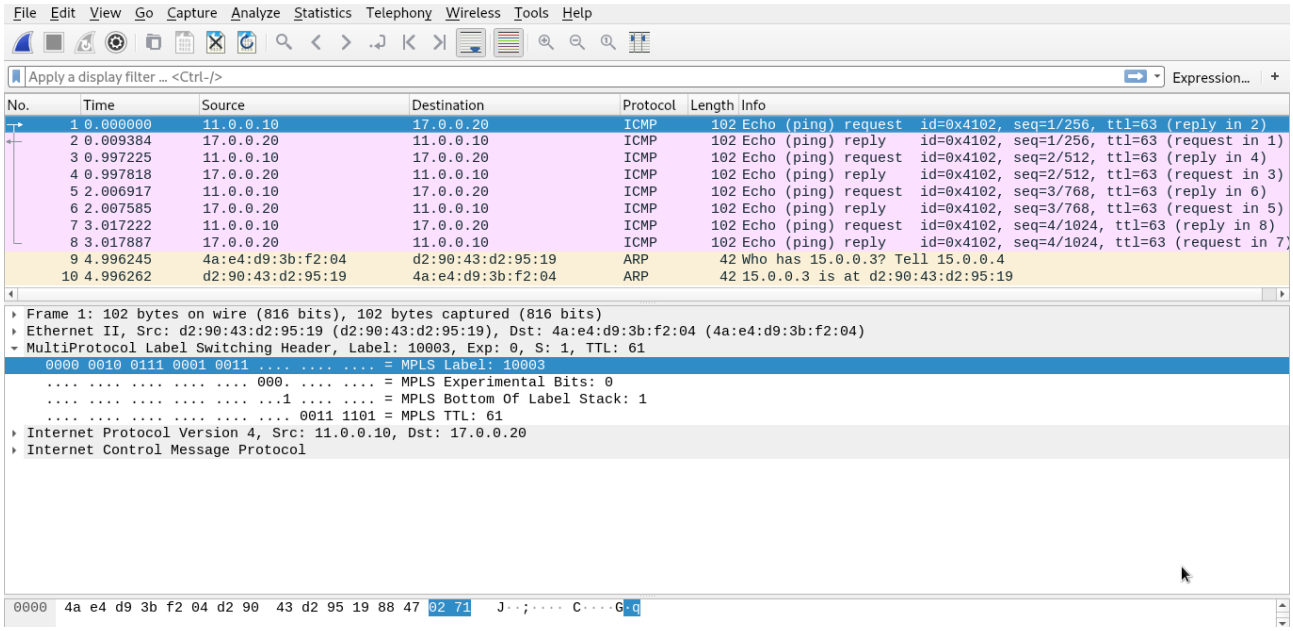
```
r2
r2:~# tcpdump -i eth1 -s 0 -w /hosthome/RAE/MPLS/mpls-01.cap
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 65535 byte
s
^C10 packets captured
10 packets received by filter
0 packets dropped by kernel
r2:~#
```

```
r3
r3:~# tcpdump -i eth1 -s 0 -w /hosthome/RAE/MPLS/mpls-02.cap
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 65535 byte
s
^C10 packets captured
10 packets received by filter
0 packets dropped by kernel
r3:~#
```





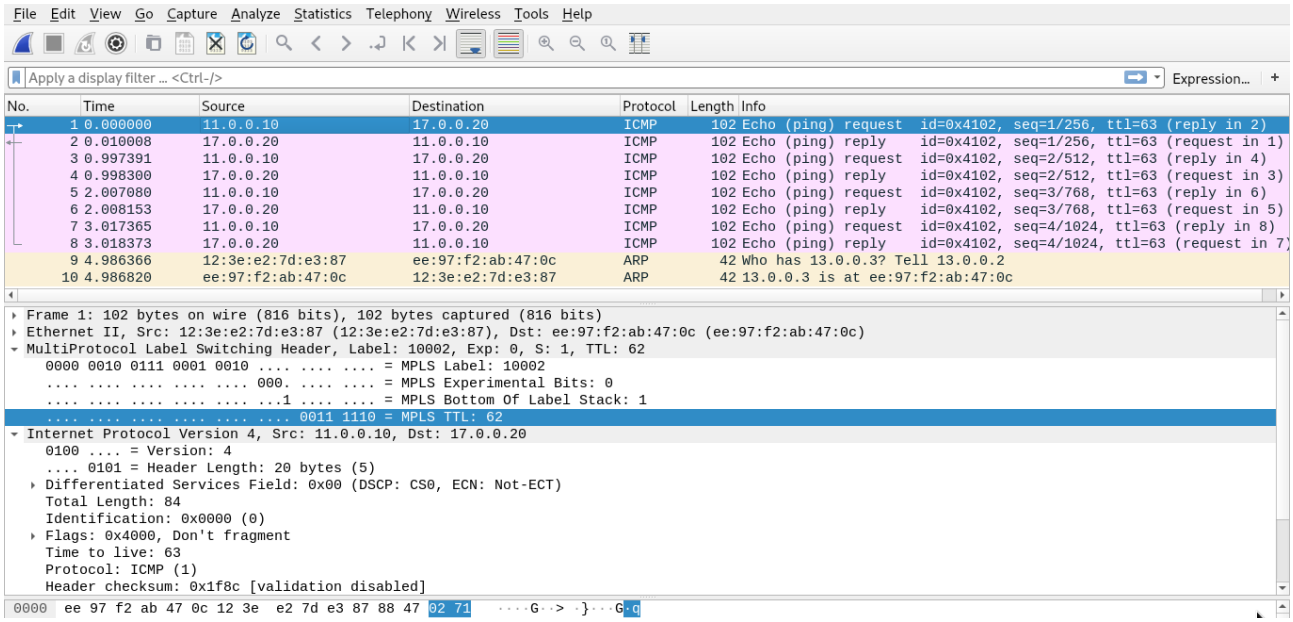
En el paquete 1, de la captura 1 realizada en R2 eth1 donde la etiqueta en el sentido de ida es 10002, podemos ver efectivamente que al ser enviado el paquete por R2 la etiqueta que tiene es la 10002.



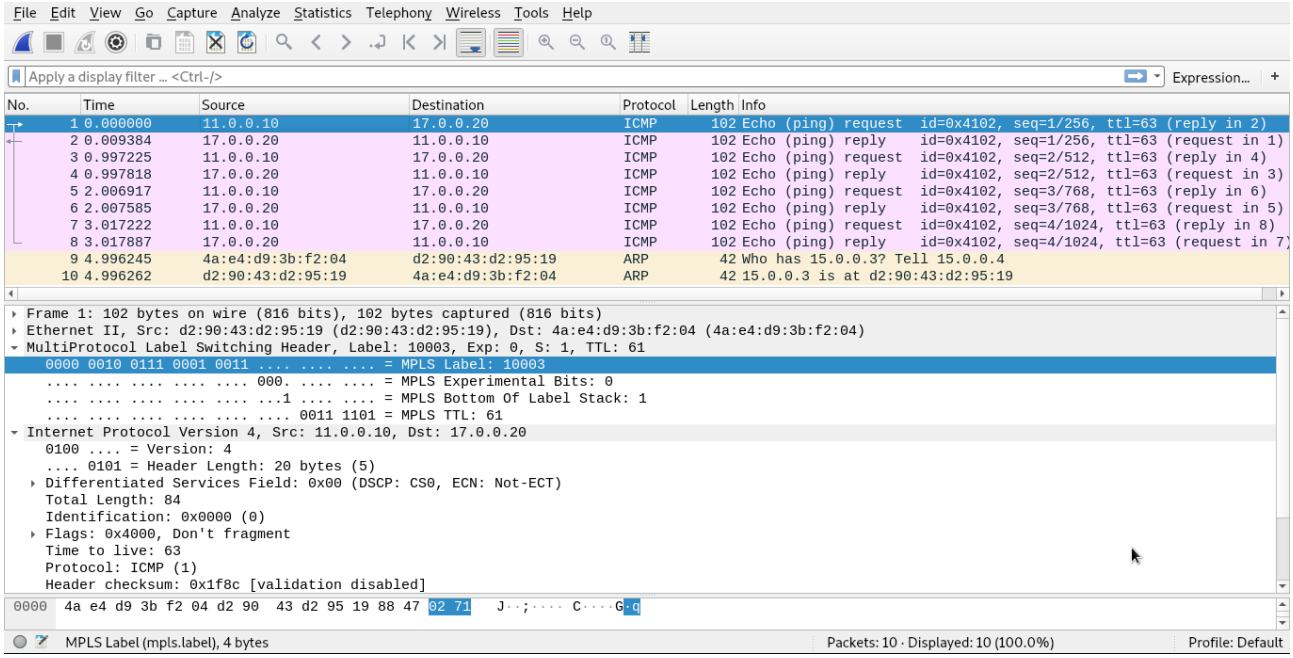
En la siguiente captura siguiendo la secuencia del mensaje enviado, se puede ver que en R3 se cambio la etiqueta a 10003 a como se puede observar en la captura. De esta misma manera ocurre en el otro sentido de la comunicación.

1.3 Observa los paquetes cuyo origen es pc1 en ambas capturas, fíjate en el campo TTL de las cabeceras IP y MPLS. Explica sus valores.

Captura 01



Captura 02



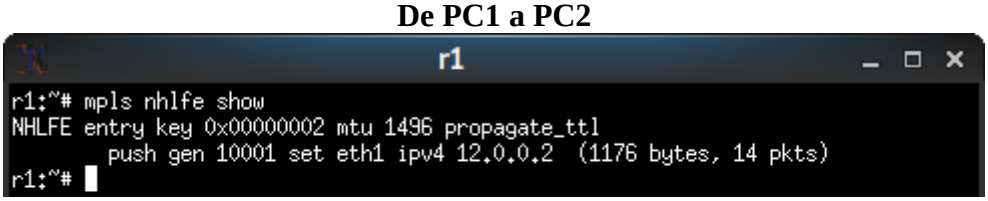
En la 01 captura en la cabecera IP el TTL del mensaje en todo el camino es de 63, y de la cabecera MPLS es de 62, en la captura 02 el TTL de la cabecera IP sigue siendo 63 y en la cabecera MPLS ahora es 61.

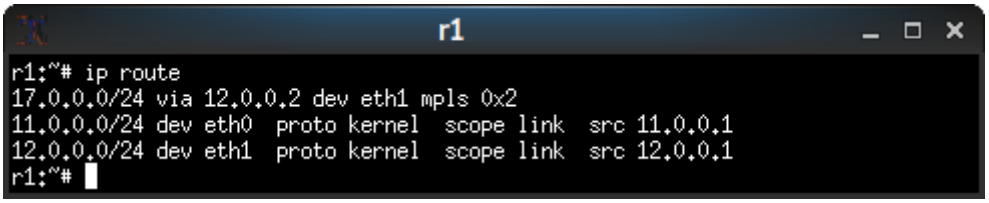
1.4 Explica con qué TTL pc2 recibirá los paquetes que envía pc1.

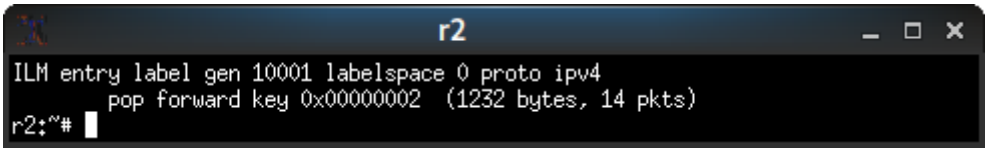
Con un TTL de 62.

1.5 Muestra las tablas NHLFE, XC, ILM e ip route en cada router e indica para cada sentido de la comunicación qué entrada de cada tabla se ha utilizado en cada salto para el envío de un paquete de pc1 a pc2 y de pc2 a pc1.

De PC1 a PC2

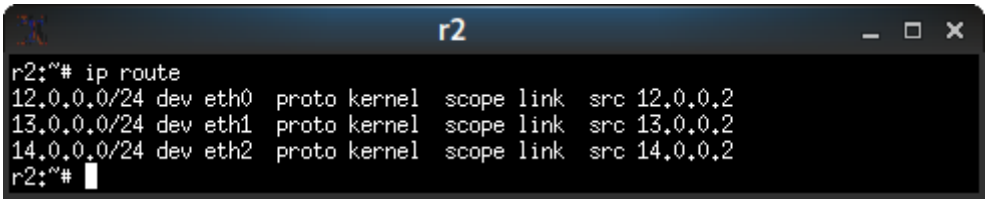



r1
r1:~# ip route
17.0.0.0/24 via 12.0.0.2 dev eth1 mpls 0x2
11.0.0.0/24 dev eth0 proto kernel scope link src 11.0.0.1
12.0.0.0/24 dev eth1 proto kernel scope link src 12.0.0.1
r1:~#

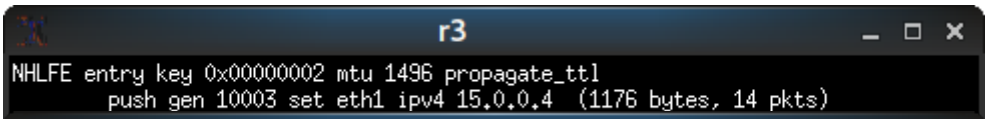

r2
ILM entry label gen 10001 labelspace 0 proto ipv4
pop forward key 0x00000002 (1232 bytes, 14 pkts)
r2:~#


r2
NHLFE entry key 0x00000002 mtu 1496 propagate_ttl
push gen 10002 set eth1 ipv4 13.0.0.3 (1176 bytes, 14 pkts)
r2:~#

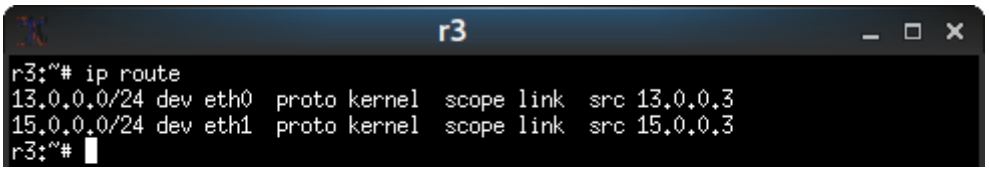

r2
XC entry ilm_label gen 10001 ilm_labelspace 0 nhlfe_key 0x00000002
r2:~#

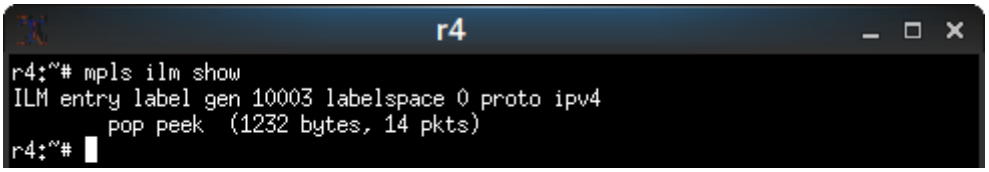

r2
r2:~# ip route
12.0.0.0/24 dev eth0 proto kernel scope link src 12.0.0.2
13.0.0.0/24 dev eth1 proto kernel scope link src 13.0.0.2
14.0.0.0/24 dev eth2 proto kernel scope link src 14.0.0.2
r2:~#

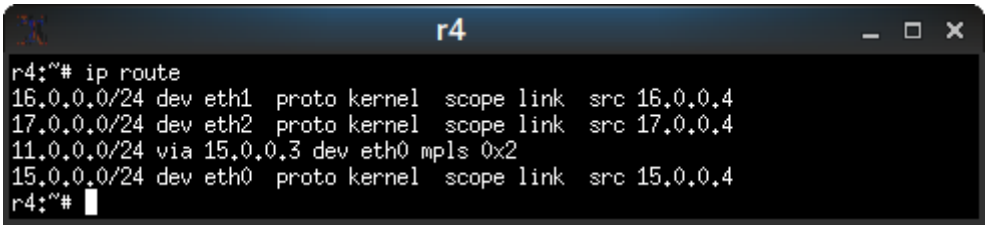

r3
ILM entry label gen 10002 labelspace 0 proto ipv4
pop forward key 0x00000002 (1232 bytes, 14 pkts)
r3:~#


r3
NHLFE entry key 0x00000002 mtu 1496 propagate_ttl
push gen 10003 set eth1 ipv4 15.0.0.4 (1176 bytes, 14 pkts)


r3
XC entry ilm_label gen 10002 ilm_labelspace 0 nhlfe_key 0x00000002
r3:~#


r3
r3:~# ip route
13.0.0.0/24 dev eth0 proto kernel scope link src 13.0.0.3
15.0.0.0/24 dev eth1 proto kernel scope link src 15.0.0.3
r3:~#


r4
r4:~# mpls ilm show
ILM entry label gen 10003 labelspace 0 proto ipv4
pop peek (1232 bytes, 14 pkts)
r4:~#


r4
r4:~# ip route
16.0.0.0/24 dev eth1 proto kernel scope link src 16.0.0.4
17.0.0.0/24 dev eth2 proto kernel scope link src 17.0.0.4
11.0.0.0/24 via 15.0.0.3 dev eth0 mpls 0x2
15.0.0.0/24 dev eth0 proto kernel scope link src 15.0.0.4
r4:~#

De PC2 a PC1

```
r4
r4:~# mpls nhlfe show
NHLFE entry key 0x00000002 mtu 1496 propagate_ttl
        push gen 10011 set eth0 ipv4 15.0.0.3  (1176 bytes, 14 pkts)
r4:~#
```

```
r3
r3:~# mpls ilm show
ILM entry label gen 10011 labelspace 0 proto ipv4
        pop forward key 0x00000003  (1232 bytes, 14 pkts)
```

```
r3
r3:~# mpls nhlfe show
NHLFE entry key 0x00000003 mtu 1496 propagate_ttl
        push gen 10012 set eth0 ipv4 13.0.0.2  (1176 bytes, 14 pkts)
```

```
r3
r3:~# mpls xc show
XC entry ilm_label gen 10011 ilm_labelspace 0 nhlfe_key 0x00000003
```

```
r2
r2:~# mpls ilm show
ILM entry label gen 10012 labelspace 0 proto ipv4
        pop forward key 0x00000003  (1232 bytes, 14 pkts)
```

```
r2
r2:~# mpls nhlfe show
NHLFE entry key 0x00000003 mtu 1496 propagate_ttl
        push gen 10013 set eth0 ipv4 12.0.0.1  (1176 bytes, 14 pkts)
```

```
r2
r2:~# mpls xc show
XC entry ilm_label gen 10012 ilm_labelspace 0 nhlfe_key 0x00000003
```

1.6 Ejecuta en pc1 la siguiente instrucción mientras capturas el tráfico en la 11.0.0.0/24 (mpls-03.cap) y en la 12.0.0.0/24 (mpls-04.cap):

pc1:~# ping -s 1472 -c 3 17.0.0.20

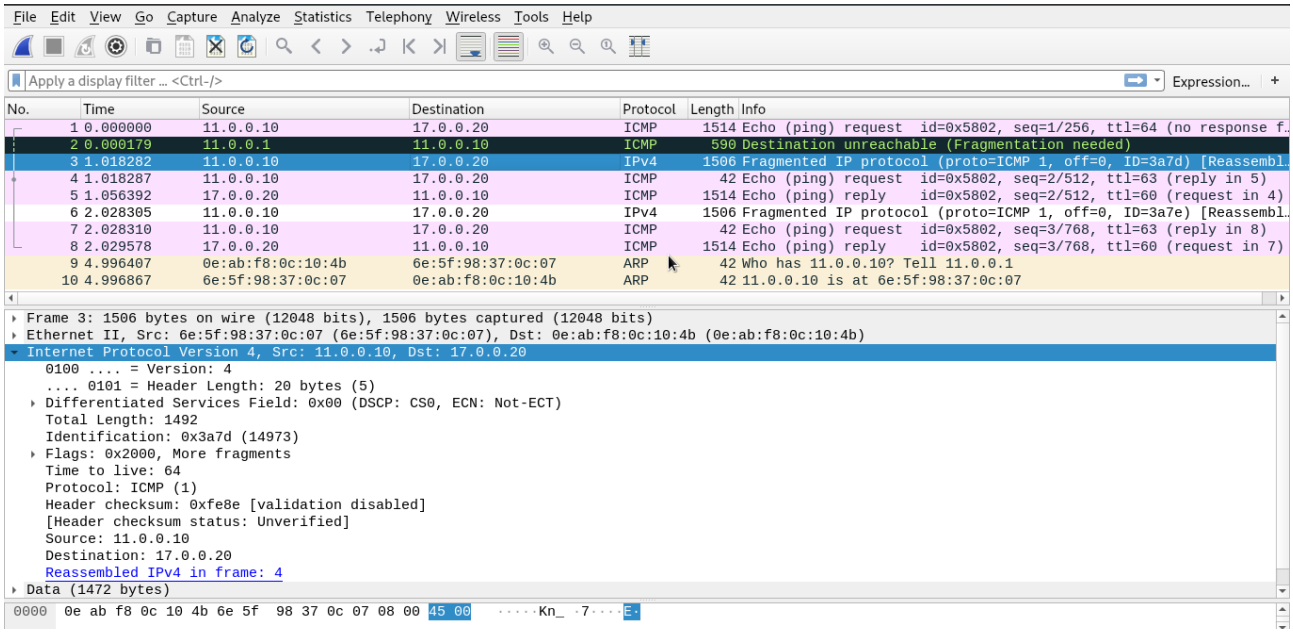
donde la opción -s indica el número de bytes de datos del mensaje ICMP Echo Request que envía ping. 1

```
pc1
pc1:~# ping -s 1472 -c 3 17.0.0.20
PING 17.0.0.20 (17.0.0.20) 1472(1500) bytes of data.
From 11.0.0.1 icmp_seq=1 Frag needed and DF set (mtu = 1496)
1480 bytes from 17.0.0.20: icmp_seq=2 ttl=60 time=38.4 ms
1480 bytes from 17.0.0.20: icmp_seq=3 ttl=60 time=1.68 ms

--- 17.0.0.20 ping statistics ---
3 packets transmitted, 2 received, +1 errors, 33% packet loss, time 2028ms
rtt min/avg/max/mdev = 1.682/20.083/38.485/18.402 ms
pc1:~#
```

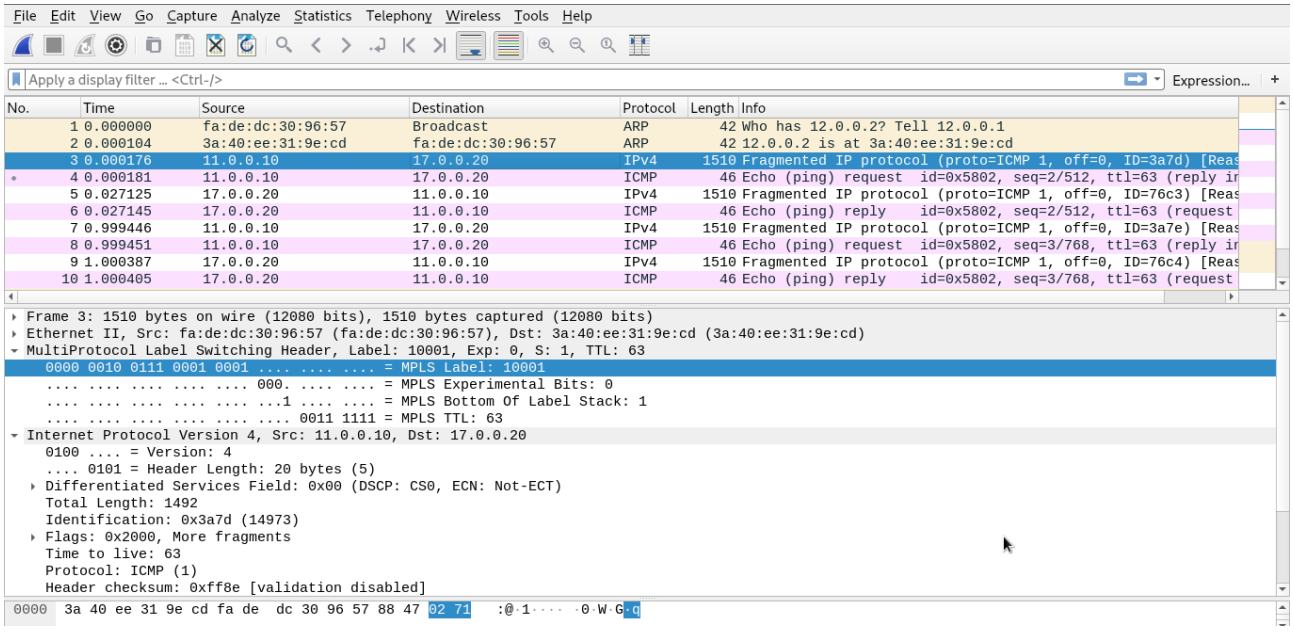
```
r1
r1:~# tcpdump -i eth0 -s 0 -w /hosthome/RAE/MPLS/mpls-031.cap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 byte
s
^C10 packets captured
10 packets received by filter
0 packets dropped by kernel
r1:~#
```

```
r2
r2:~# tcpdump -i eth0 -s 0 -w /hosthome/RAE/MPLS/mpls-041.cap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 byte
s
^C12 packets captured
12 packets received by filter
0 packets dropped by kernel
r2:~#
```

Explica el resultado de ambas capturas. ¿Qué crees que ocurriría si se transmitieran estos paquetes de ping en una red IP en la que no se utilizara MPLS?
Captura en R1



Captura en R2



Debido a que MPLS no puede manejar la fragmentación, si un paquete tiene que ser fragmentado, solo en el primer fragmento quedarian todas las etiquetas intactas. Por lo cual el resto se perdera.

2. Fusión de etiquetas: Comunicación entre pc3 y pc2 a través de MPLS

Cuando dos máquinas están enviando tráfico a un mismo destino, en algún punto del camino la ruta que siguen los paquetes de cada una de las dos máquinas origen puede solapar hasta llegar al destino. En la parte del camino en la que solapan, MPLS puede utilizar las mismas etiquetas para que los paquetes alcancen el destino. Por ejemplo en la figura 2 si hay tráfico dirigido a la subred 14.0.0.0/24 desde r1 y desde r4, en el tramo r2 → r3 todo el tráfico dirigido a esa subred puede compartir la etiqueta 10002, siendo que parte del tráfico traía etiqueta 10001 y otra parte 20001.

2.1 Realiza las mínimas modificaciones necesarias en los scripts MPLS del apartado 1 para que pc3 pueda intercambiar tráfico con pc2. Ten en cuenta la siguiente consideración:

En el camino común que comparten pc1 → pc2 y pc3 → pc2 debes reutilizar la configuración de etiquetas existente.

Los valores de las nuevas etiquetas que tengas que introducir en este apartado deberán ser diferentes de los ya utilizados.

En cada salto MPLS los routers deberán utilizar una etiqueta diferente a la recibida.

Todas las etiquetas de un router deberán pertenecer al mismo ámbito (labelspace 0).

Incluye los scripts que hayas modificado en la memoria, explicando las modificaciones.

De PC3 a PC2

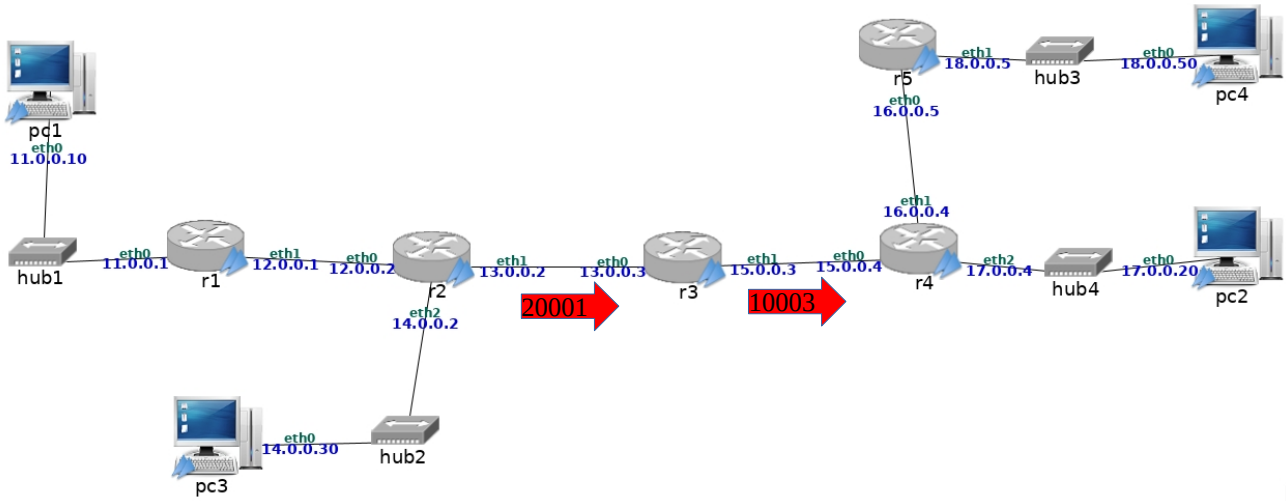
r2

```
 GNU nano 2.0.7                File: derecha.sh

#!/bin/sh

echo "Envio desde 11.0.0.0/24"

mpls labelspace set dev eth0 labelspace 0
mpls ilm add label gen 10001 labelspace 0

key_1=`mpls nhlfe add key 0 instructions push gen 10002 \
       nexthop eth1 ipv4 13.0.0.3 | grep key | cut -d " " -f 4`

mpls xc add ilm_label gen 10001 ilm_labelspace 0 nhlfe_key $key_1

echo "Envio desde 14.0.0.0/24"

key_3=`mpls nhlfe add key 0 instructions push gen 20001 \
       nexthop eth1 ipv4 13.0.0.3 | grep key | cut -d " " -f 4`

ip route add 17.0.0.0/24 via 13.0.0.3 mpls $key_3

                [ Read 18 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

r3

```
 GNU nano 2.0.7                File: derecha.sh

#!/bin/sh

echo "Envio a 11.0.0.0/24"

mpls labelspace set dev eth0 labelspace 0
mpls ilm add label gen 10002 labelspace 0

key_1=`mpls nhlfe add key 0 instructions push gen 10003 \
       nexthop eth1 ipv4 15.0.0.4 | grep key | cut -d " " -f 4`

mpls xc add ilm_label gen 10002 ilm_labelspace 0 nhlfe_key $key_1

echo "Envio desde 14.0.0.0/24"
mpls ilm add label gen 20001 labelspace 0
mpls xc add ilm_label gen 20001 ilm_labelspace 0 nhlfe_key $key_1

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

- En R4 de ida no se ha modificado nada



De PC2 a PC3

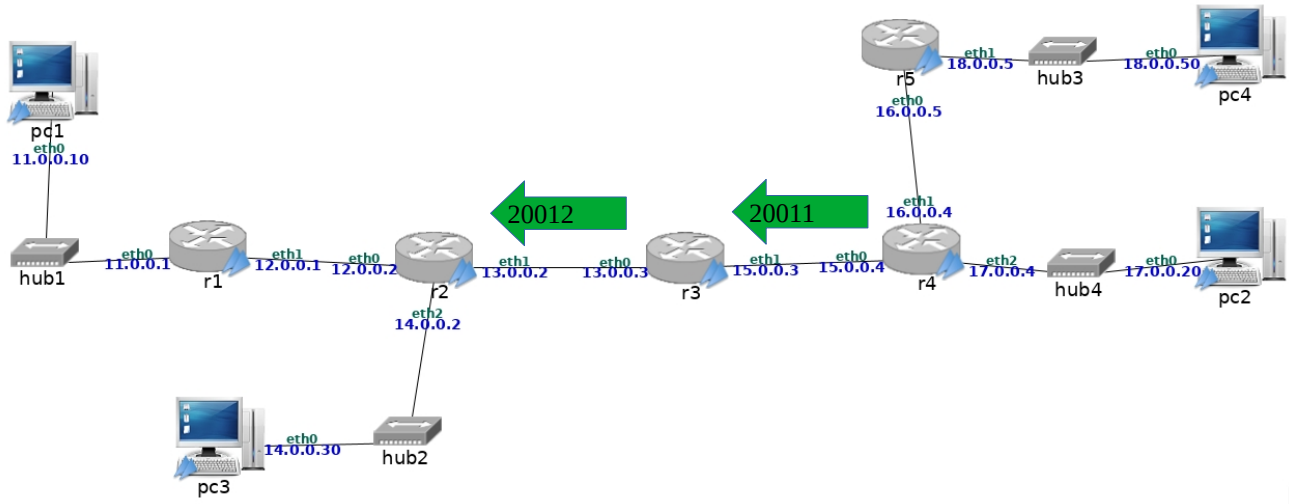
r4

```
 GNU nano 2.0.7                File: izquierda.sh

#!/bin/sh

echo "Envio a 11.0.0.0/24"

key_2=`mpls nhlfe add key 0 instructions push gen 10011 \
        nexthop eth0 ipv4 15.0.0.3 | grep key | cut -d " " -f 4`

ip route add 11.0.0.0/24 via 15.0.0.3 mpls $key_2

echo "Envio a 14.0.0.0/24"
key_4=`mpls nhlfe add key 0 instructions push gen 20011 \
        nexthop eth0 ipv4 15.0.0.3 | grep key | cut -d " " -f 4`

ip route add 14.0.0.0/24 via 15.0.0.3 mpls $key_4

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

r3

```
 GNU nano 2.0.7                File: izquierda.sh

#!/bin/sh

echo "Envio a 11.0.0.0/24"

mpls labelspace set dev eth1 labelspace 0

mpls ilm add label gen 10011 labelspace 0

key_2=`mpls nhlfe add key 0 instructions push gen 10012 \
        nexthop eth0 ipv4 13.0.0.2 | grep key | cut -d " " -f 4`

mpls xc add ilm_label gen 10011 ilm_labelspace 0 nhlfe_key $key_2

echo "Envio a 14.0.0.0/24"
key_4=`mpls nhlfe add key 0 instructions push gen 20012 \
        nexthop eth0 ipv4 13.0.0.2 | grep key | cut -d " " -f 4`

mpls ilm add label gen 20011 labelspace 0
mpls xc add ilm_label gen 20011 ilm_labelspace 0 nhlfe_key $key_4

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

```
r2
 GNU nano 2.0.7                File: izquierda.sh

#!/bin/sh

echo "Envio a 11.0.0.0/24"

mpls labelspace set dev eth1 labelspace 0

mpls ilm add label gen 10012 labelspace 0

key_2=`mpls nhlfe add key 0 instructions push gen 10013 \
       nexthop eth0 ipv4 12.0.0.1 | grep key | cut -d " " -f 4`

mpls xc add ilm_label gen 10012 ilm_labelspace 0 nhlfe_key $key_2

echo "Envio a 14.0.0.0/24"

mpls labelspace set dev eth0 labelspace 0
mpls ilm add label gen 20012 labelspace 0

                [ Read 17 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```



2.2 Realiza una captura de tráfico en la 13.0.0.0/24 mientras ejecutas un ping -c 5 desde pc1 a pc2 y desde pc3 a pc2 simultáneamente, guarda la captura en mpls-05.cap. Explica las etiquetas MPLS de los paquetes capturados.

```
r3
r3:~# tcpdump -i eth0 -s 0 -w /hosthome/RAE/MPLS/mpls-05.cap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 byte
s
```

```
pc1
pc1:~# ping -c 5 17.0.0.20
PING 17.0.0.20 (17.0.0.20) 56(84) bytes of data.
64 bytes from 17.0.0.20: icmp_seq=1 ttl=60 time=15.0 ms
64 bytes from 17.0.0.20: icmp_seq=2 ttl=60 time=1.55 ms
64 bytes from 17.0.0.20: icmp_seq=3 ttl=60 time=1.60 ms
64 bytes from 17.0.0.20: icmp_seq=4 ttl=60 time=1.99 ms
64 bytes from 17.0.0.20: icmp_seq=5 ttl=60 time=1.67 ms

--- 17.0.0.20 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4048ms
rtt min/avg/max/mdev = 1.559/4.378/15.063/5.344 ms
pc1:~#
```

```
pc3
pc3:~# ping -c 5 17.0.0.20
PING 17.0.0.20 (17.0.0.20) 56(84) bytes of data.
64 bytes from 17.0.0.20: icmp_seq=1 ttl=61 time=10.7 ms
64 bytes from 17.0.0.20: icmp_seq=2 ttl=61 time=1.27 ms
64 bytes from 17.0.0.20: icmp_seq=3 ttl=61 time=1.27 ms
64 bytes from 17.0.0.20: icmp_seq=4 ttl=61 time=1.43 ms
64 bytes from 17.0.0.20: icmp_seq=5 ttl=61 time=1.32 ms

--- 17.0.0.20 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4047ms
rtt min/avg/max/mdev = 1.273/3.215/10.767/3.776 ms
pc3:~#
```

```
r3
r3:~# tcpdump -i eth0 -s 0 -w /hosthome/RAE/MPLS/mpls-05.cap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 byte
s
^C22 packets captured
22 packets received by filter
0 packets dropped by kernel
r3:~#
```

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/> Expression... +

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	11.0.0.10	17.0.0.20	ICMP	102	Echo (ping) request id=0xcd02, seq=1/256, ttl=63 (reply in
2	0.010971	17.0.0.20	11.0.0.10	ICMP	102	Echo (ping) reply id=0xcd02, seq=1/256, ttl=63 (request
3	0.368633	14.0.0.30	17.0.0.20	ICMP	102	Echo (ping) request id=0xcf02, seq=1/256, ttl=63 (reply in
4	0.369269	17.0.0.20	14.0.0.30	ICMP	102	Echo (ping) reply id=0xcf02, seq=1/256, ttl=63 (request
5	1.015174	11.0.0.10	17.0.0.20	ICMP	102	Echo (ping) request id=0xcd02, seq=2/512, ttl=63 (reply in
6	1.015729	17.0.0.20	11.0.0.10	ICMP	102	Echo (ping) reply id=0xcd02, seq=2/512, ttl=63 (request
7	1.376244	14.0.0.30	17.0.0.20	ICMP	102	Echo (ping) request id=0xcf02, seq=2/512, ttl=63 (reply in
8	1.376747	17.0.0.20	14.0.0.30	ICMP	102	Echo (ping) reply id=0xcf02, seq=2/512, ttl=63 (request
9	2.024898	11.0.0.10	17.0.0.20	ICMP	102	Echo (ping) request id=0xcd02, seq=3/768, ttl=63 (reply in
10	2.025580	17.0.0.20	11.0.0.10	ICMP	102	Echo (ping) reply id=0xcd02, seq=3/768, ttl=63 (request

```
▸ Frame 1: 102 bytes on wire (816 bits), 102 bytes captured (816 bits)
▸ Ethernet II, Src: 12:3e:e2:7d:e3:87 (12:3e:e2:7d:e3:87), Dst: ee:97:f2:ab:47:0c (ee:97:f2:ab:47:0c)
▾ MultiProtocol Label Switching Header, Label: 10002, Exp: 0, S: 1, TTL: 62
    0000 0010 0111 0001 0010 .... .... .... = MPLS Label: 10002
    .... .... .... .... .... 000. .... .... = MPLS Experimental Bits: 0
    .... .... .... .... .... ...1 .... .... = MPLS Bottom Of Label Stack: 1
    .... .... .... .... .... .... 0011 1110 = MPLS TTL: 62
▾ Internet Protocol Version 4, Src: 11.0.0.10, Dst: 17.0.0.20
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0x0000 (0)
  ▸ Flags: 0x4000, Don't fragment
    Time to live: 63
    Protocol: ICMP (1)
    Header checksum: 0x1f8c [validation disabled]
```

```
0000  ee 97 f2 ab 47 0c 12 3e  e2 7d e3 87 88 47 02 71   ····G··> ·}···G·q
```

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	11.0.0.10	17.0.0.20	ICMP	102	Echo (ping) request id=0xcd02, seq=1/256, ttl=63 (reply in
2	0.010971	17.0.0.20	11.0.0.10	ICMP	102	Echo (ping) reply id=0xcd02, seq=1/256, ttl=63 (request
3	0.368633	14.0.0.30	17.0.0.20	ICMP	102	Echo (ping) request id=0xcf02, seq=1/256, ttl=63 (reply in
4	0.369269	17.0.0.20	14.0.0.30	ICMP	102	Echo (ping) reply id=0xcf02, seq=1/256, ttl=63 (request
5	1.015174	11.0.0.10	17.0.0.20	ICMP	102	Echo (ping) request id=0xcd02, seq=2/512, ttl=63 (reply in
6	1.015729	17.0.0.20	11.0.0.10	ICMP	102	Echo (ping) reply id=0xcd02, seq=2/512, ttl=63 (request
7	1.376244	14.0.0.30	17.0.0.20	ICMP	102	Echo (ping) request id=0xcf02, seq=2/512, ttl=63 (reply in
8	1.376747	17.0.0.20	14.0.0.30	ICMP	102	Echo (ping) reply id=0xcf02, seq=2/512, ttl=63 (request
9	2.024898	11.0.0.10	17.0.0.20	ICMP	102	Echo (ping) request id=0xcd02, seq=3/768, ttl=63 (reply in
10	2.025580	17.0.0.20	11.0.0.10	ICMP	102	Echo (ping) reply id=0xcd02, seq=3/768, ttl=63 (request

```
▸ Frame 3: 102 bytes on wire (816 bits), 102 bytes captured (816 bits)
▸ Ethernet II, Src: 12:3e:e2:7d:e3:87 (12:3e:e2:7d:e3:87), Dst: ee:97:f2:ab:47:0c (ee:97:f2:ab:47:0c)
▾ MultiProtocol Label Switching Header, Label: 20001, Exp: 0, S: 1, TTL: 63
    0000 0100 1110 0010 0001 .... .... .... = MPLS Label: 20001
    .... .... .... .... .... 000. .... .... = MPLS Experimental Bits: 0
    .... .... .... .... .... ...1 .... .... = MPLS Bottom Of Label Stack: 1
    .... .... .... .... .... .... 0011 1111 = MPLS TTL: 63
▾ Internet Protocol Version 4, Src: 14.0.0.30, Dst: 17.0.0.20
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0x0000 (0)
  ▸ Flags: 0x4000, Don't fragment
    Time to live: 63
    Protocol: ICMP (1)
    Header checksum: 0x1c78 [validation disabled]
```

```
0000  ee 97 f2 ab 47 0c 12 3e  e2 7d e3 87 88 47 04 e2   ····G··> ·}···G··
```

MPLS Label (mpls.label), 4 bytes | Packets: 22 · Displayed: 22 (100.0%) | Profile: Default

3. Etiquetas con ámbito de interfaz: Comunicación entre pc3 y pc4 a través de MPLS

3.1 Piensa en qué modificaciones tendrías que realizar para que pc3 y pc4 pudieran intercambiar tráfico, sin usar túnel. Explica en qué lugares podrías aplicar la técnica de fusión de etiquetas realizada en 2.

3.2 Modifica los scripts del apartado 2 para permitir la comunicación entre pc3 y pc4. Si es necesario crear nuevos scripts en otras máquinas, hazlo. Ten en cuenta las siguientes consideraciones:

Debes utilizar cuando sea posible la técnica de la fusión de etiquetas.

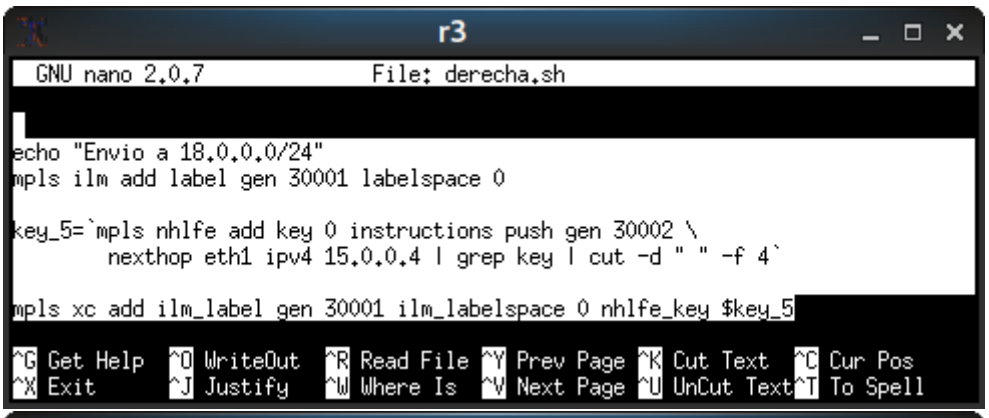
Los valores de las nuevas etiquetas que introduzcas en este apartado deberán ser diferentes de los ya utilizados.

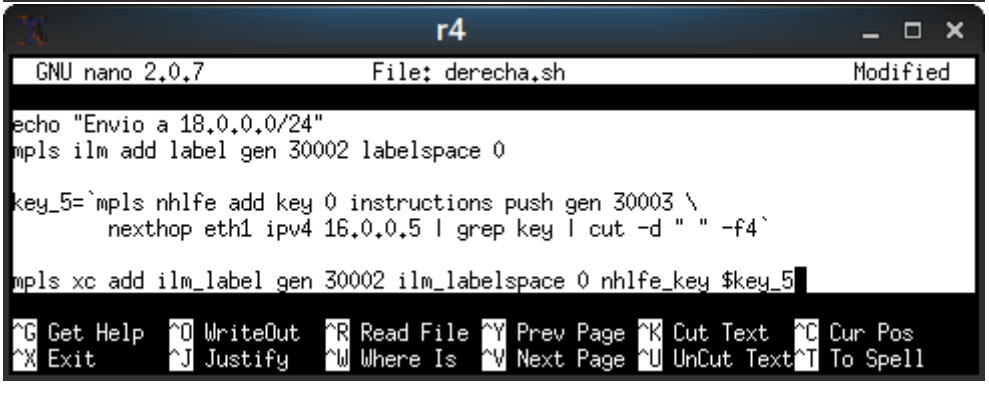
Todos los routers deberán utilizar el labelspace 0 para definir etiquetas con ámbito de router, excepto en r4 que deberá utilizar etiquetas con ámbito de interfaz.

Para que resulte necesario utilizar etiquetas con ámbito de interfaz en r4, obliga a que r5 envíe a r4 para alcanzar la 14.0.0.0/24 la misma etiqueta que ya has utilizado en r3 cuando envía a r4 para alcanzar la
17.0.0.0/24. De esta forma te resultará necesario que en r4 los mismos valores de etiquetas recibidos a través de interfaces diferentes sean tratados como etiquetas diferentes.

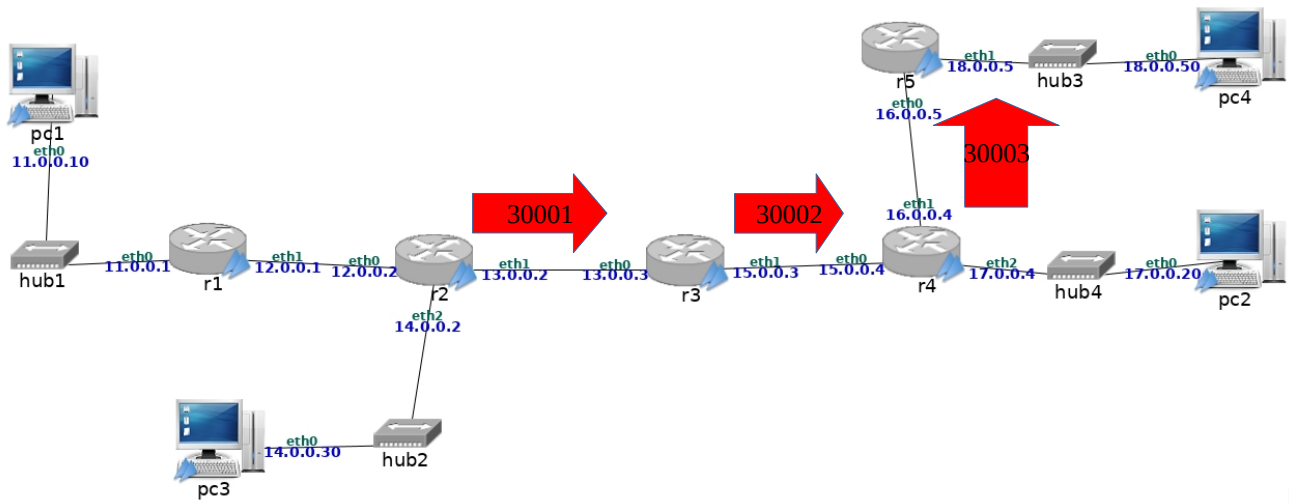
Incluye los scripts que hayas modificado/creado en la memoria, explicando las modificaciones.

De PC3 a PC4





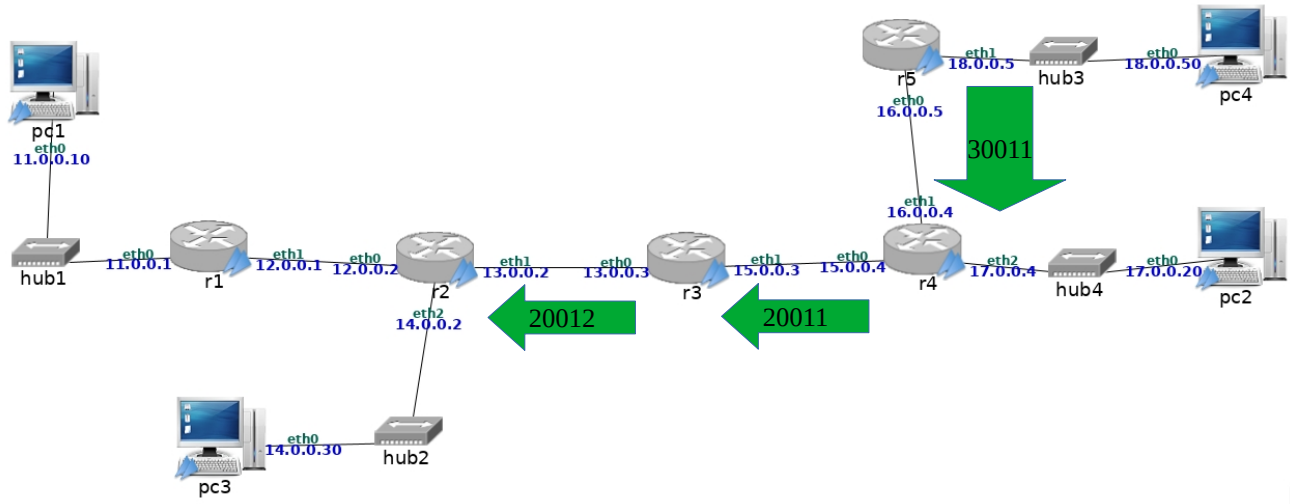
```
r5
 GNU nano 2.0.7                File: derecha.sh

#!/bin/bash
echo "Envio a 18.0.0.0/24"

mpls labelspace set dev eth0 labelspace 0
mpls ilm add label gen 30003 labelspace 0


^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```



De PC4 a PC3

```
r5
 GNU nano 2.0.7                File: izquierda.sh

#!/bin/bash

key_6=`mpls nhlfe add key 0 instructions push gen 30011 \
        nexthop eth0 ipv4 16.0.0.4 | grep key | cut -d " " -f 4`

ip route add 14.0.0.0/24 via 16.0.0.4 mpls $key_6


^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

```
r4
 GNU nano 2.0.7                File: izquierda.sh


echo "Envio desde 18.0.0.0/24"
mpls labelspace set dev eth1 labelspace 0
mpls ilm add label gen 30011 labelspace 0

mpls xc add ilm_label gen 30011 ilm_labelspace 0 nhlfe_key $key_4


^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```



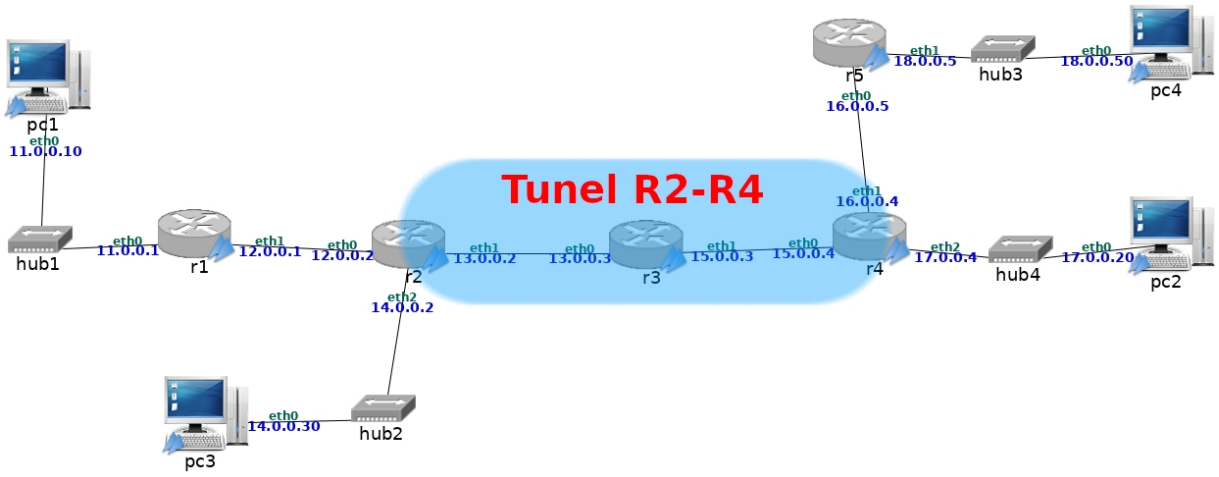
3.3 Muestra las tablas ILM y NHLFE de r4 explicando qué ocurre en el caso de recibir la etiqueta que es igual a través de las 2 interfaces diferentes.

```
r4
r4:~# mpls ilm show
ILM entry label gen 30011 labelspace 0 proto ipv4
        pop forward key 0x00000004  (704 bytes, 8 pkts)
ILM entry label gen 30002 labelspace 0 proto ipv4
        pop forward key 0x00000002  (704 bytes, 8 pkts)
ILM entry label gen 10003 labelspace 0 proto ipv4
        pop peek  (352 bytes, 4 pkts)
r4:~#
```

```
r4
r4:~# mpls nhlfe show
NHLFE entry key 0x00000004 mtu 1496 propagate_ttl
        push gen 20011 set eth0 ipv4 15.0.0.3  (840 bytes, 10 pkts)
NHLFE entry key 0x00000003 mtu 1496 propagate_ttl
        push gen 10011 set eth0 ipv4 15.0.0.3  (168 bytes, 2 pkts)
NHLFE entry key 0x00000002 mtu 1496 propagate_ttl
        push gen 30003 set eth1 ipv4 16.0.0.5  (672 bytes, 8 pkts)
r4:~#
```

4 Túnel: apilar etiquetas MPLS

4.1 Vamos a configurar un túnel MPLS entre r2 y r4 en ambos sentidos:



En el sentido r2 → r3 → r4: Todo el tráfico que tiene que reenviar r2 de pc1 y pc3 irá encapsulado con una cabecera adicional MPLS. En r3 se mantendrá la cabecera adicional realizando el swap de etiquetas. En r4 se eliminará la cabecera adicional, realizando el reenvío basado en la cabecera MPLS más interna.

En el sentido r4 → r3 → r2: Todo el tráfico que tiene que reenviar r4 de pc2 y pc4 irá encapsulado con una cabecera adicional MPLS. En r3 se mantendrá la cabecera adicional realizando el swap de etiquetas. En r2 se eliminará la cabecera adicional, realizando el reenvío basado en la cabecera MPLS más interna.

Modifica los scripts de las máquinas que consideres necesario para que implementen el túnel descrito. Incluye los nuevos scripts en la memoria.

De ida

r2

GNU nano 2.0.7 File: tunnelida.sh Modified

```
#!/bin/sh

echo "Envio desde 11.0.0.0/24"
mpls labelspace set dev eth0 labelspace 0
mpls ilm add label gen 10001 labelspace 0

key_1=`mpls nhlfe add key 0 instructions push gen 10002 \\
        nexthop eth1 ipv4 13.0.0.3 | grep key | cut -d " " -f 4`

key_11=`mpls nhlfe add key 0 instructions push gen 11111 \\
        forward $key_1 | grep key | cut -d " " -f 4`

mpls xc add ilm_label gen 10001 ilm_labelspace 0 nhlfe_key $key_1

echo "Envio desde 14.0.0.0/24"
key_3=`mpls nhlfe add key 0 instructions push gen 20001 \\
        nexthop eth1 ipv4 13.0.0.3 | grep key | cut -d " " -f 4`

key_33=`mpls nhlfe add key 0 instructions push gen 22221 \\
        forward $key_3 | grep key | cut -d " " -f 4`

ip route add 17.0.0.0/24 via 13.0.0.3 mpls $key_33

echo "Envio a 18.0.0.0/24"
key_5=`mpls nhlfe add key 0 instructions push gen 30001 \\
        nexthop eth1 ipv4 13.0.0.3 | grep key | cut -d " " -f 4`

key_55=`mpls nhlfe add key 0 instructions push gen 33331 \\
        forward $key_5 | grep key | cut -d " " -f 4`

ip route add 18.0.0.0/24 via 13.0.0.3 mpls $key_55
```

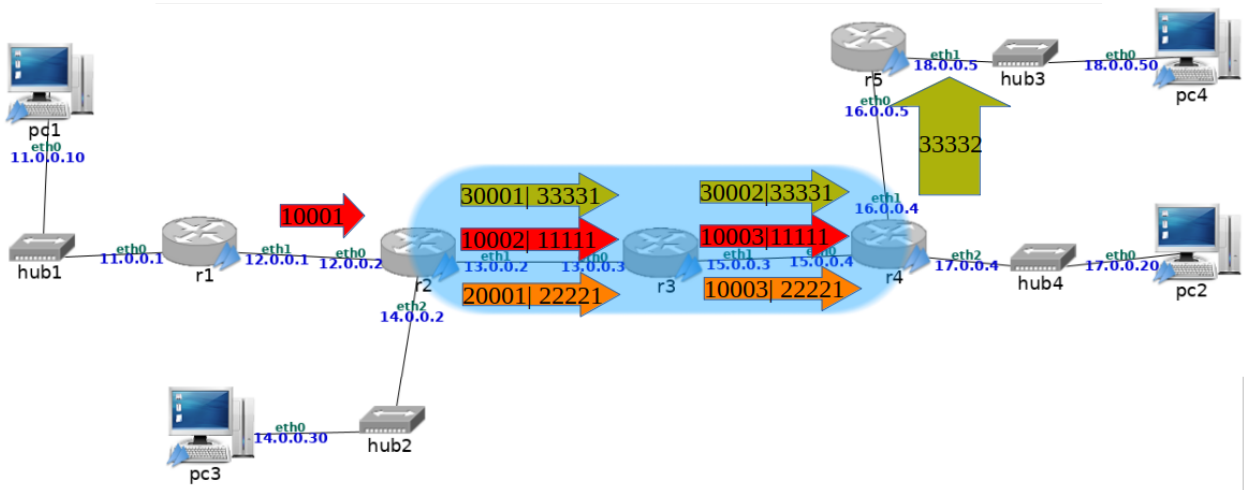
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

r4

```
 GNU nano 2.0.7                File: tunelida.sh

#!/bin/sh

echo "Envio a 17.0.0.0/24"
mpls labelspace set dev eth0 labelspace 0
mpls ilm add label gen 11111 labelspace 0
mpls ilm add label gen 10003 labelspace 0
mpls ilm add label gen 22221 labelspace 0
mpls ilm add label gen 30002 labelspace 0
mpls ilm add label gen 33331 labelspace 0

echo "Envio a 18.0.0.0/24"

key_5=`mpls nhlfe add key 0 instructions push gen 33332 \
       nexthop eth1 ipv4 16.0.0.5 | grep key | cut -d " " -f4`

mpls xc add ilm_label gen 33331 ilm_labelspace 0 nhlfe_key $key_5


^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

r5

```
 GNU nano 2.0.7                File: tunnelida.sh

#!/bin/bash
echo "Envio a 18.0.0.0/24"

mpls labelspace set dev eth0 labelspace 0
mpls ilm add label gen 33332 labelspace 0

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```



De PC1 a PC2

De PC3 a PC2

De PC2 a PC4

De vuelta

r4

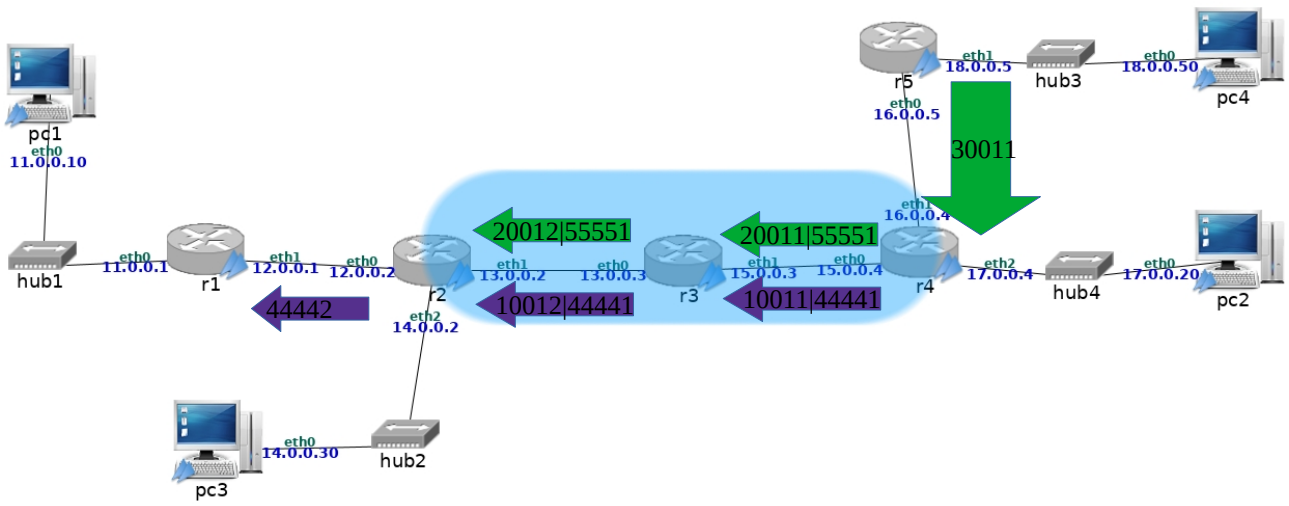
```
 GNU nano 2.0.7            File: tunelregreso.sh

#!/bin/sh

echo "Envio a 11.0.0.0/24"

key_2=`mpls nhlfe add key 0 instructions push gen 10011 \
        nexthop eth0 ipv4 15.0.0.3 | grep key | cut -d " " -f 4`

key_20=`mpls nhlfe add key 0 instructions push gen 44441 \
        forward $key_2 | grep key | cut -d " " -f 4`

ip route add 11.0.0.0/24 via 15.0.0.3 mpls $key_20

echo "Envio a 14.0.0.0/24"

key_4=`mpls nhlfe add key 0 instructions push gen 20011 \
        nexthop eth0 ipv4 15.0.0.3 | grep key | cut -d " " -f 4`

key_40=`mpls nhlfe add key 0 instructions push gen 55551 \
        forward $key_4 | grep key | cut -d " " -f 4`

ip route add 14.0.0.0/24 via 15.0.0.3 mpls $key_40

echo "Envio desde 18.0.0.0/24"
mpls labelspace set dev eth1 labelspace 0
mpls ilm add label gen 30011 labelspace 0

mpls xc add ilm_label gen 30011 ilm_labelspace 0 nhlfe_key $key_4

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

r2

```
 GNU nano 2.0.7            File: tunnelregreso.sh

#!/bin/sh

echo "Envio a 11.0.0.0/24"

mpls labelspace set dev eth1 labelspace 0
mpls ilm add label gen 10012 labelspace 0
mpls ilm add label gen 44441 labelspace 0

key_2=`mpls nhlfe add key 0 instructions push gen 44442 \
        nexthop eth0 ipv4 12.0.0.1 | grep key | cut -d " " -f 4`

mpls xc add ilm_label gen 44441 ilm_labelspace 0 nhlfe_key $key_2

echo "Envio a 14.0.0.0/24"

mpls ilm add label gen 20012 labelspace 0
mpls ilm add label gen 55551 labelspace 0

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

r1

```
 GNU nano 2.0.7            File: tunelregreso.sh

#!/bin/sh

echo "Envio a 11.0.0.0/24"

mpls labelspace set dev eth1 labelspace 0

mpls ilm add label gen 44442 labelspace 0

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```



De PC4 y PC2 a PC3

De PC2 a PC1

4.2 Realiza una captura de tráfico en la 13.0.0.0/24 (mpls-06.cap), 15.0.0.0/24 (mpls-07.cap) y 17.0.0.0/24 (mpls-08.cap) mientras ejecutas un ping -c 5 desde pc1 a pc2 y desde pc3 a pc4 simultáneamente. Explica las etiquetas MPLS de los paquetes capturados.

```
r2
r2:~# tcpdump -i eth1 -s 0 -w /hosthome/RAE/MPLS/mpls-07.cap
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 65535 byte
s
```

```
r3
r3:~# tcpdump -i eth1 -s 0 -w /hosthome/RAE/MPLS/mpls-08.cap
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 65535 byte
s
```

```
r4
r4:~# tcpdump -i eth2 -s 0 -w /hosthome/RAE/MPLS/mpls-09.cap
tcpdump: listening on eth2, link-type EN10MB (Ethernet), capture size 65535 byte
s
```

```
pc1
pc1:~# ping -c 5 17.0.0.20
PING 17.0.0.20 (17.0.0.20) 56(84) bytes of data.
64 bytes from 17.0.0.20: icmp_seq=1 ttl=60 time=11.5 ms
64 bytes from 17.0.0.20: icmp_seq=2 ttl=60 time=1.63 ms
64 bytes from 17.0.0.20: icmp_seq=3 ttl=60 time=1.64 ms
64 bytes from 17.0.0.20: icmp_seq=4 ttl=60 time=0.465 ms
64 bytes from 17.0.0.20: icmp_seq=5 ttl=60 time=1.39 ms

--- 17.0.0.20 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4024ms
rtt min/avg/max/mdev = 0.465/3.335/11.536/4.123 ms
pc1:~#
```

```
pc3
pc3:~# ping -c 5 18.0.0.50
PING 18.0.0.50 (18.0.0.50) 56(84) bytes of data.
64 bytes from 18.0.0.50: icmp_seq=1 ttl=60 time=15.5 ms
64 bytes from 18.0.0.50: icmp_seq=2 ttl=60 time=1.98 ms
64 bytes from 18.0.0.50: icmp_seq=3 ttl=60 time=1.92 ms
64 bytes from 18.0.0.50: icmp_seq=4 ttl=60 time=1.86 ms
64 bytes from 18.0.0.50: icmp_seq=5 ttl=60 time=0.448 ms

--- 18.0.0.50 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4048ms
rtt min/avg/max/mdev = 0.448/4.363/15.589/5.642 ms
pc3:~#
```

```
r2
r2:~# tcpdump -i eth1 -s 0 -w /hosthome/RAE/MPLS/mpls-07.cap
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 65535 byte
s
^C22 packets captured
22 packets received by filter
0 packets dropped by kernel
r2:~#
```

```
r3
r3:~# tcpdump -i eth1 -s 0 -w /hosthome/RAE/MPLS/mpls-08.cap
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 65535 byte
s
^C22 packets captured
22 packets received by filter
0 packets dropped by kernel
r3:~#
```

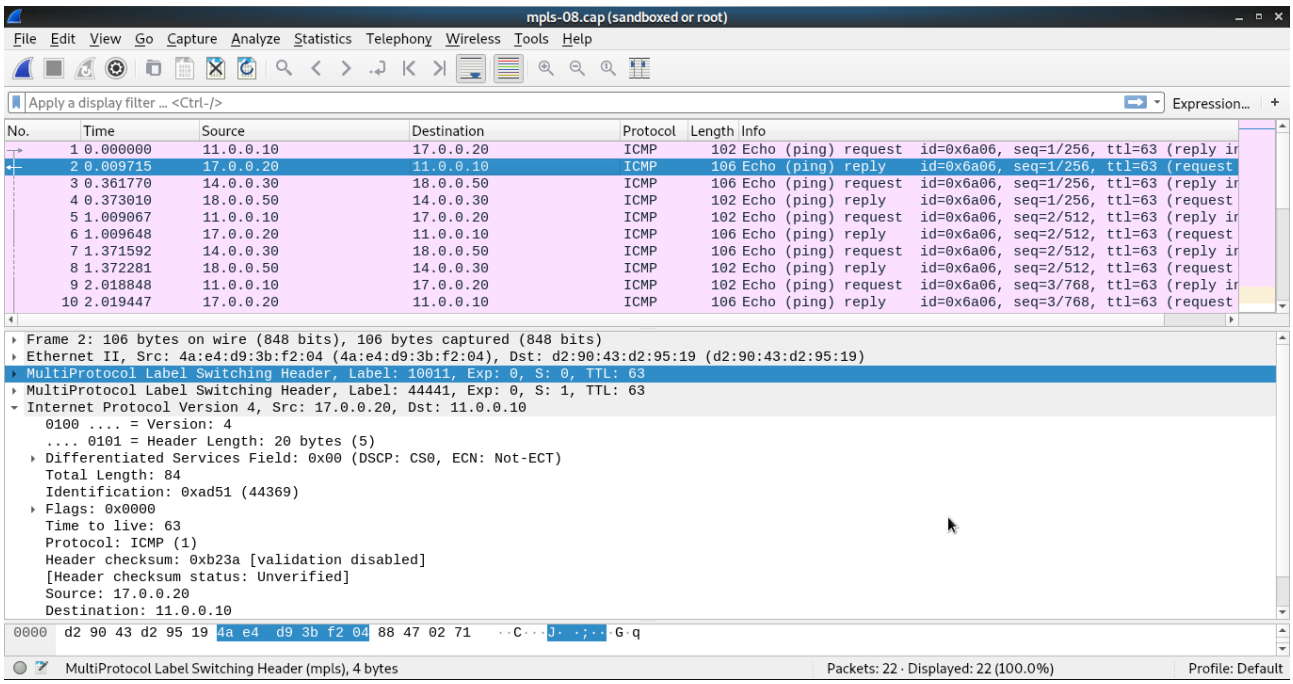
```
r4
r4:~# tcpdump -i eth2 -s 0 -w /hosthome/RAE/MPLS/mpls-09.cap
tcpdump: listening on eth2, link-type EN10MB (Ethernet), capture size 65535 byte
s
^C14 packets captured
14 packets received by filter
0 packets dropped by kernel
r4:~#
```

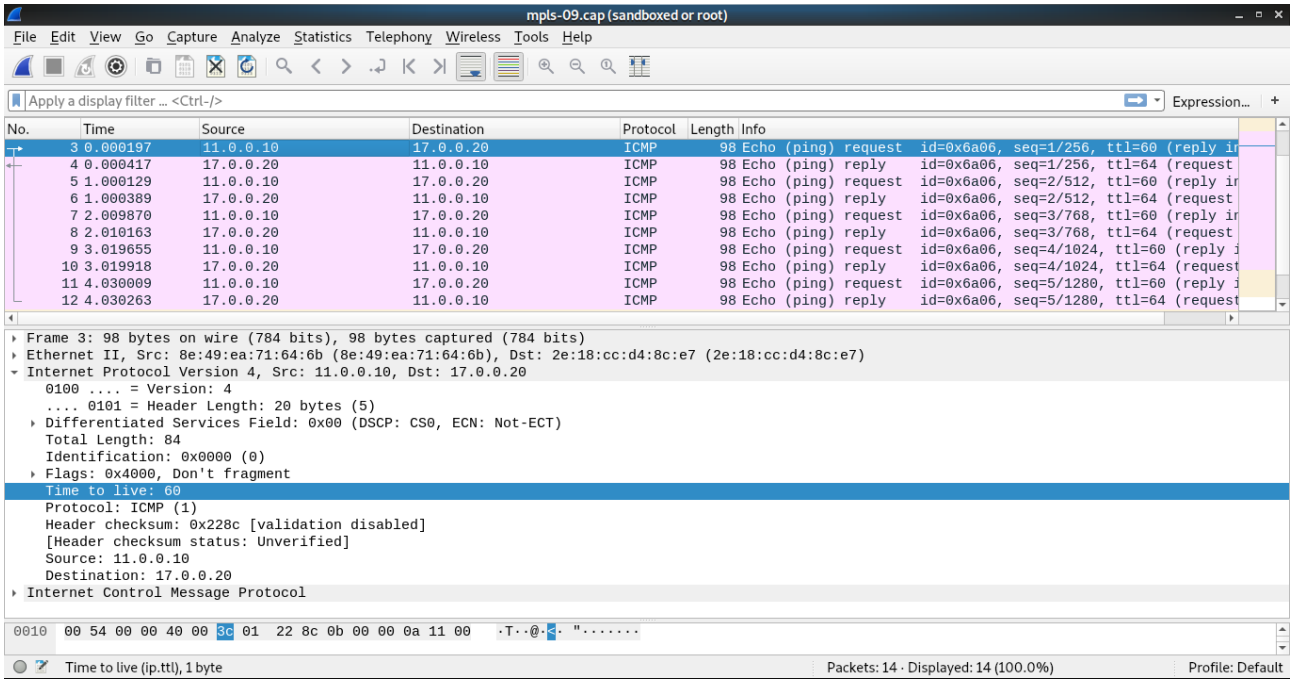
4.3 Observando las capturas explica el valor de TTL de las cabeceras MPLS e IP en cada una de ellas.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	11.0.0.10	17.0.0.20	ICMP	102	Echo (ping) request id=0x6a06, seq=1/256, ttl=63 (reply ir
2	0.009944	17.0.0.20	11.0.0.10	ICMP	106	Echo (ping) reply id=0x6a06, seq=1/256, ttl=63 (request
3	0.361694	14.0.0.30	18.0.0.50	ICMP	106	Echo (ping) request id=0x6a06, seq=1/256, ttl=63 (reply ir
4	0.373301	18.0.0.50	14.0.0.30	ICMP	102	Echo (ping) reply id=0x6a06, seq=1/256, ttl=63 (request
5	1.009006	11.0.0.10	17.0.0.20	ICMP	102	Echo (ping) request id=0x6a06, seq=2/512, ttl=63 (reply ir
6	1.009882	17.0.0.20	11.0.0.10	ICMP	106	Echo (ping) reply id=0x6a06, seq=2/512, ttl=63 (request
7	1.371563	14.0.0.30	18.0.0.50	ICMP	106	Echo (ping) request id=0x6a06, seq=2/512, ttl=63 (reply ir
8	1.372497	18.0.0.50	14.0.0.30	ICMP	102	Echo (ping) reply id=0x6a06, seq=2/512, ttl=63 (request
9	2.018784	11.0.0.10	17.0.0.20	ICMP	102	Echo (ping) request id=0x6a06, seq=3/768, ttl=63 (reply ir
10	2.019674	17.0.0.20	11.0.0.10	ICMP	106	Echo (ping) reply id=0x6a06, seq=3/768, ttl=63 (request

```
▸ Frame 2: 106 bytes on wire (848 bits), 106 bytes captured (848 bits)
▸ Ethernet II, Src: ee:97:f2:ab:47:0c (ee:97:f2:ab:47:0c), Dst: 12:3e:e2:7d:e3:87 (12:3e:e2:7d:e3:87)
▸ MultiProtocol Label Switching Header, Label: 10012, Exp: 0, S: 0, TTL: 62
▸ MultiProtocol Label Switching Header, Label: 44441, Exp: 0, S: 1, TTL: 63
▾ Internet Protocol Version 4, Src: 17.0.0.20, Dst: 11.0.0.10
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0xad51 (44369)
  ▸ Flags: 0x0000
    Time to live: 63
    Protocol: ICMP (1)
    Header checksum: 0xb23a [validation disabled]
    [Header checksum status: Unverified]
    Source: 17.0.0.20
    Destination: 11.0.0.10
```

```
0000  12 3e e2 7d e3 87 ee 97  f2 ab 47 0c 88 47 02 71   ·>·}····  ··G··G·q
```





4.4 ¿Cuál sería la cantidad máxima de datos IP que podrían atravesar el túnel sin fragmentación?