

# COMUNICACIÓN DE DATOS - TP LAN

- Alumno: Lautaro De Lucía
- Padrón: 100203
- Fecha: 06/11/2023

1) Diseñar una topología que contemple las siguientes redes:

Red	Cantidad de Hosts conectados	Cantidad de Hosts que se podrían llegar a conectar
Red A	2	55
Red B	1	24
Red C	2	28
Red D	1	15
Red E	2	8
Red F	1	11
Red G	2	9

A partir del bloque de direcciones IP 130.55.48.0/23, definir los bloques de direcciones IP para las Redes de la Tabla, realizando las asignaciones de manera eficiente en función de la cantidad de hosts que se podrían llegar a conectar.

Indicar para cada Red de la Tabla la dirección de red y su máscara.

Dada la máscara /23, determinamos el espacio designado para **red** y **host** respectivamente:

$D : 130.55.48.0/23$

$B : 10000010.00110111.00110000.00000000$

$M : 1111111.11111111.11111110.00000000$

$B \& M : 10000010.00110111.00110000.00000000$

Tenemos un total de 9 dígitos para sub-netting y asignación de hosts.

Como ninguna red excede un umbral de  $2^6 = 64$  hosts a conectarse, podemos utilizar los dígitos 7, 8 y 9 para el subnetting, obteniendo de esa forma 7 redes con espacio para 64 hosts para cada uno. Sin embargo, esta **no** es una asignación **eficiente**, ya que tenemos información de la cantidad de límite de hosts para cada red, siendo muchas de estas muy inferiores a 64 hosts.

Si queremos hacer una asignación eficiente, conviene determinar el tamaño mínimo de máscara de sub-red para cada Red:

<i>Red</i>	<i>MaxHosts</i>	<i>MinBits</i>	<i>Máscara</i>	<i>Rango Total</i>
A	55	6	/26	64
B	24	5	/27	32
C	28	5	/27	32
D	15	5	/27	32
E	8	4	/28	16
F	11	4	/28	16
G	9	4	/28	16

- **Obs**

Notar que utilizamos 32 bits para la red D, ya que una máscara de sub-red de 16 bits solo permite 14 espacios para hosts, teniendo que reservar 2 espacios para red y broadcast respectivamente.

Luego, podemos hacer un sub-netting de la forma:

<i>Red</i>	<i>Dirección</i>	<i>Rango</i>
<i>A</i>	130.55.48.0/26	130.55.48.0 – 130.55.48.63
<i>B</i>	130.55.48.64/27	130.55.48.64 – 130.55.48.95
<i>C</i>	130.55.48.96/27	130.55.48.96 – 130.55.48.127
<i>D</i>	130.55.48.128/27	130.55.48.128 – 130.55.48.159
<i>E</i>	130.55.48.160/28	130.55.48.160 – 130.55.48.175
<i>F</i>	130.55.48.176/28	130.55.48.176 – 130.55.48.191
<i>G</i>	130.55.48.192/28	130.55.48.192 – 130.55.48.207

- **Obs**

Notar que estamos incluyendo la dirección de red y broadcast en el rango.

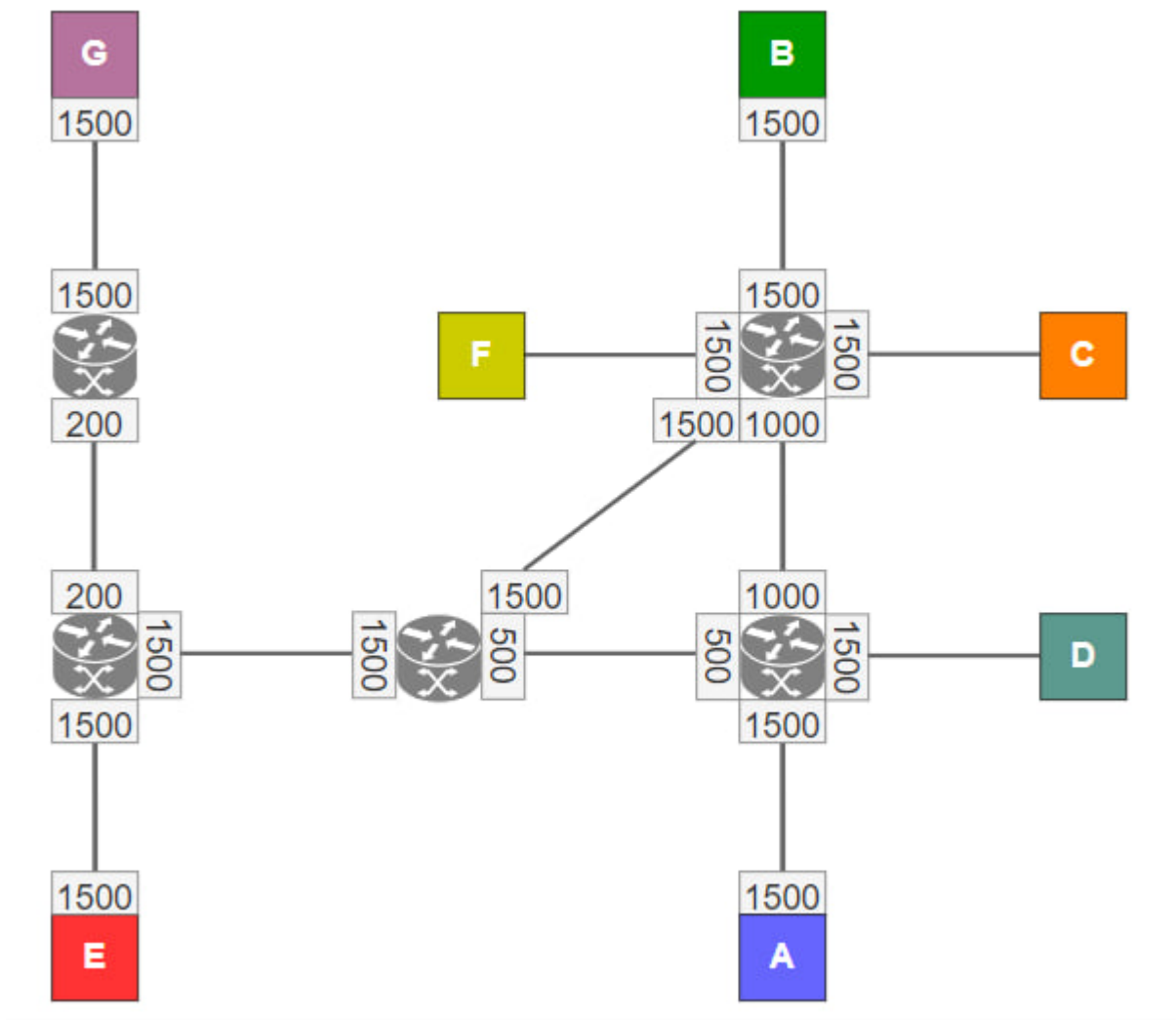
2) Realizar la topología en Imunes empleando la menor cantidad de elementos de red (switches, routers, etc.) de manera tal que se cumpla lo siguiente:

- Red A necesita un TTL mínimo = 4 para alcanzar a la Red E y el Path MTU mínimo es 500.
  - Red B necesita un TTL mínimo = 2 para alcanzar a la Red C y el Path MTU mínimo es 1500.
  - Red E necesita un TTL mínimo = 4 para alcanzar a la Red C y el Path MTU mínimo es 1500.
  - Red F necesita un TTL mínimo = 3 para alcanzar a la Red D y el Path MTU mínimo es 1000.
  - Red G necesita un TTL mínimo = 3 para alcanzar a la Red E y el Path MTU mínimo es 200.
  - El camino entre Red B y Red A tiene un Path MTU mínimo de 1000.
- 
- Las rutas tienen que ser óptimas (menor cantidad de saltos posibles).
  - Las Redes A, B, C, D, E, F y G tienen MTU = 1500.
  - La Red A solo tiene conexión a 1 router.

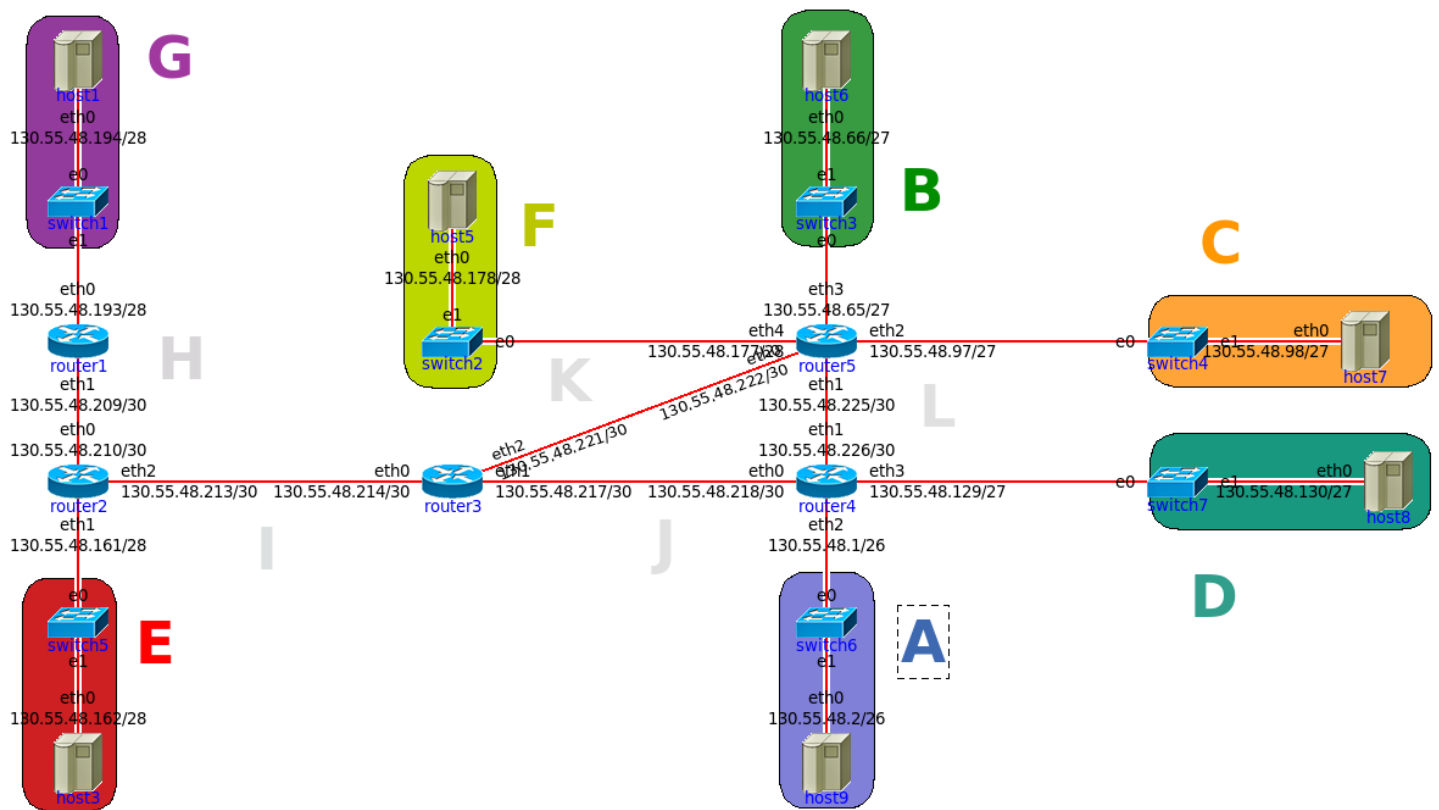
Como criterio de diseño, consideramos preferible que todas las interfaces en una ruta específica tengan el mismo MTU. Esto a modo de evitar la fragmentación innecesaria de paquetes, que puede causar una sobrecarga y potencialmente reducir el rendimiento.

Con esto, encontramos que la cantidad mínima de routers que nos permiten obtener un diseño que cumpla con todos los requisitos es cinco.

A continuación, se muestra un diagrama que muestra la topología de nuestra red. Es fácil ver que se cumplen las condiciones de el enunciado.



Pasando esta topología a IMUNES, esta nos queda de la forma:



Donde se definen sub-redes *punto a punto*  $H, I, J, K, L$  para los enlaces entre routers.

Red	Dirección	Rango
$H$	130.55.48.208/30	130.55.48.208 – 130.55.48.211
$I$	130.55.48.212/30	130.55.48.212 – 130.55.48.215
$J$	130.55.48.216/30	130.55.48.216 – 130.55.48.219
$K$	130.55.48.220/30	130.55.48.220 – 130.55.48.223
$L$	130.55.48.224/30	130.55.48.224 – 130.55.48.227

3) Indicar los comandos a ejecutar con todas las opciones necesarias para obtener como respuesta:

a. Ejecutar un ping desde un host de la Red A que genere la llegada de un mensaje ICMP Echo Reply a un host de la red B con las siguientes características:

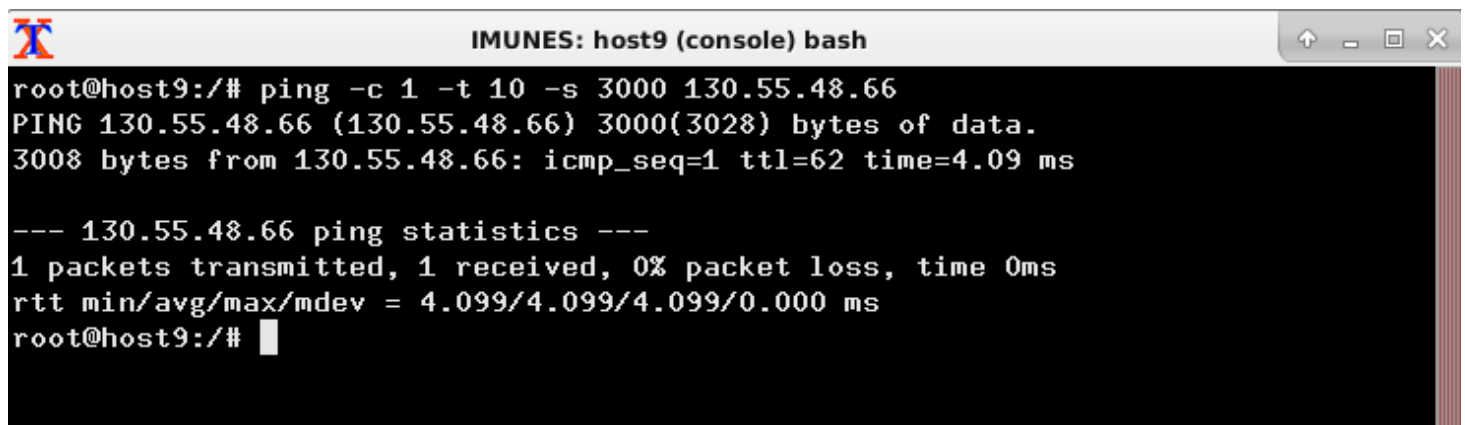
TTL = 10

Payload IP = Mayor a 970 bytes

MF = 1

Fragment Offset = 244 (expresión en múltiplo de 8 bytes)

A continuación, mostramos capturas de la terminal deel Host de la Red A y la captura de Wireshark de su interface eth0.



```
IMUNES: host9 (console) bash
root@host9:/# ping -c 1 -t 10 -s 3000 130.55.48.66
PING 130.55.48.66 (130.55.48.66) 3000(3028) bytes of data.
3008 bytes from 130.55.48.66: icmp_seq=1 ttl=62 time=4.09 ms

--- 130.55.48.66 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 4.099/4.099/4.099/0.000 ms
root@host9:/#
```

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	130.55.48.1	224.0.0.9	RIPv2	266	Response
2	6.652511	42:00:aa:00:00:0f	Broadcast	ARP	42	Who has 130.55.48.1? Tell 130.55.48.2
3	6.653050	42:00:aa:00:00:0e	42:00:aa:00:00:0f	ARP	42	130.55.48.1 is at 42:00:aa:00:00:0e
4	6.653090	130.55.48.2	130.55.48.66	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=2345) [Reassembled in #6]
5	6.653241	130.55.48.2	130.55.48.66	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=2345) [Reassembled in #6]
6	6.653288	130.55.48.2	130.55.48.66	ICMP	82	Echo (ping) request id=0x00d2, seq=1/256, ttl=10 (reply in 10)
7	6.656493	130.55.48.66	130.55.48.2	IPv4	1010	Fragmented IP protocol (proto=ICMP 1, off=0, ID=1ea0) [Reassembled in #10]
8	6.656535	130.55.48.66	130.55.48.2	IPv4	1010	Fragmented IP protocol (proto=ICMP 1, off=976, ID=1ea0) [Reassembled in #10]
9	6.656552	130.55.48.66	130.55.48.2	IPv4	1010	Fragmented IP protocol (proto=ICMP 1, off=1952, ID=1ea0) [Reassembled in #10]
10	6.656558	130.55.48.66	130.55.48.2	ICMP	114	Echo (ping) reply id=0x00d2, seq=1/256, ttl=62 (request in 6)
11	11.816328	42:00:aa:00:00:0e	42:00:aa:00:00:0f	ARP	42	Who has 130.55.48.2? Tell 130.55.48.1
12	11.816332	42:00:aa:00:00:0f	42:00:aa:00:00:0e	ARP	42	130.55.48.2 is at 42:00:aa:00:00:0f
13	15.117972	fe80::4000:aaff:fe0...	ff02::9	RIPng	106	Command Response, Version 1
14	24.007394	130.55.48.1	224.0.0.9	RIPv2	266	Response
15	31.130648	fe80::4000:aaff:fe0...	ff02::9	RIPng	106	Command Response, Version 1

- En la primera imagen vemos que el ping fue exitoso, enviando un paquete de 3000 bytes a la dirección 130.55.48.66 (Host en Red B). Recibimos una respuesta indicando que no hubo pérdida de paquetes, y el time to live (TTL) del paquete recibido es de 62, lo que es consistente con la topología de la red.
- En la captura de Wireshark, vemos como la interface primero observa el **ping request** saliendo de el Host A con destino en el Host B, fragmentado en 2 partes dado el *MTU*1500 de la interfaz eth0 de el host. Luego, recibe el **ping reply** fragmentado en tres paquetes (dado el *MTU*1000

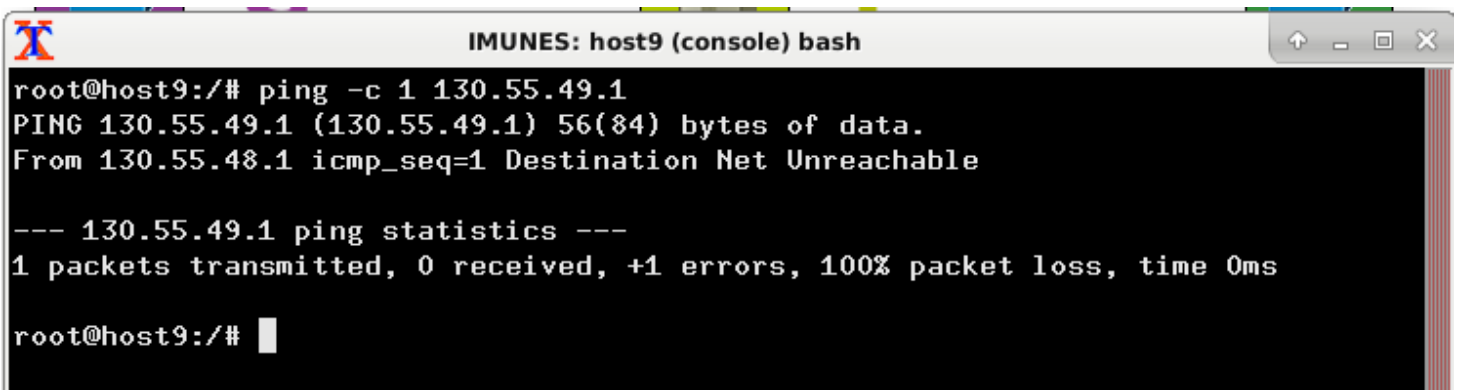
en el camino de la respuesta). El fragment offset de el tercer paquete es de 1952 bits ( $8 \times 244$  bytes).

- b. Ejecutar desde algún equipo un comando que permita generar tráfico con los siguientes mensajes:
  - ICMP Destination Unreachable:
    - a. Net Unreachable
    - b. Host Unreachable
    - c. Fragmentation Needed and Don't Fragment was set

Indicar desde qué equipo se ejecutaron los comandos y especificar los comandos ejecutados con todas las opciones empleadas para generar el tráfico solicitado.

## A. Net Unreachable

Realizamos un ping apuntado a una red inexistente (cambiamos 48 por 49 en el tercer byte)



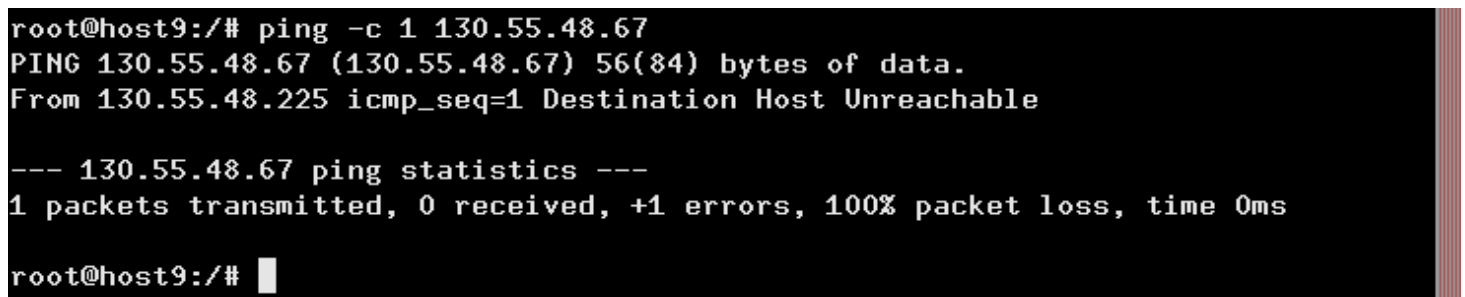
```
IMUNES: host9 (console) bash
root@host9:/# ping -c 1 130.55.49.1
PING 130.55.49.1 (130.55.49.1) 56(84) bytes of data.
From 130.55.48.1 icmp_seq=1 Destination Net Unreachable

--- 130.55.49.1 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

root@host9:/#
```

## B. Host Unreachable

Realizamos un ping apuntando a un host inexistente en la Red B.



```
root@host9:/# ping -c 1 130.55.48.67
PING 130.55.48.67 (130.55.48.67) 56(84) bytes of data.
From 130.55.48.225 icmp_seq=1 Destination Host Unreachable

--- 130.55.48.67 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

root@host9:/#
```

## C. Fragmentation Needed and Don't Fragment was set

Hacemos nuevamente el ping de el ejercicio 3a pero esta vez le agregamos la opción -M do para especificar que el paquete no debería fragmentarse en su trayecto, especificando un tamaño de 1400 mayor a el MTU=1000 de la ruta.

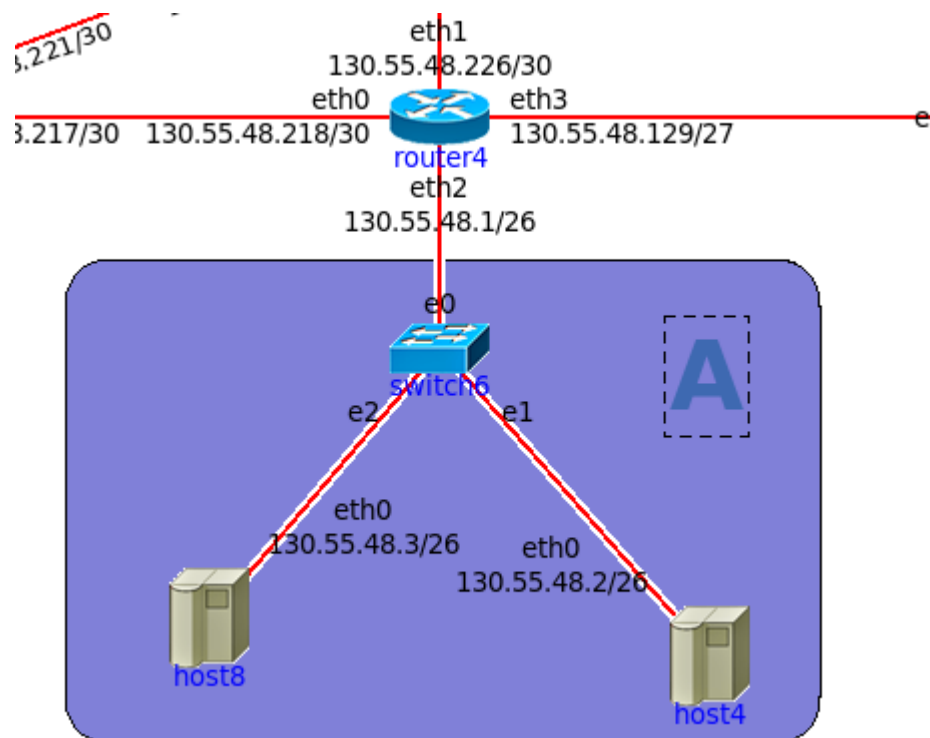
```
root@host9:/# ping -c 1 -M do -s 1400 130.55.48.66
PING 130.55.48.66 (130.55.48.66) 1400(1428) bytes of data.
From 130.55.48.1 icmp_seq=1 Frag needed and DF set (mtu = 1000)

--- 130.55.48.66 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
```

- c. Realizando la menor cantidad de modificaciones en la configuración de las Tablas de ruteo, generar un mensaje de tipo ICMP Redirect.

Indicar desde qué equipo se ejecutaron los comandos y especificar los comandos ejecutados con todas las opciones empleadas para generar el tráfico solicitado.

Como nuestra topología no incluye switches conectados a dos routers simultaneamente, la única forma de recibir un ICMP Redirect es seleccionando un Host dentro de una misma sub-red como el default gateway de otro host. Configuramos la topología de la Red A de la siguiente forma:



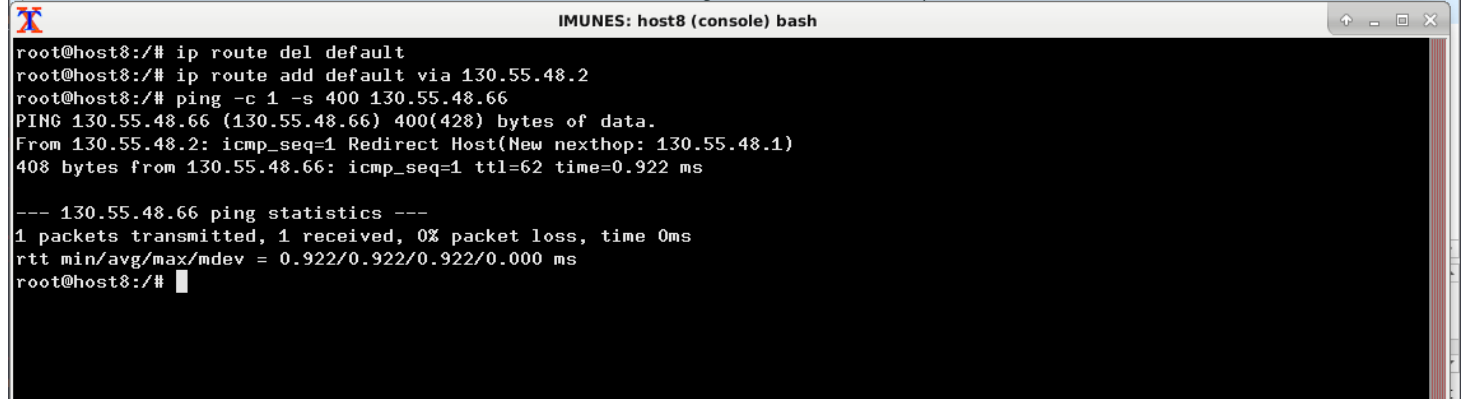


Es decir, agregamos un segundo host a la Red A en la topología de Imunes. Luego, ejecutamos los siguientes comandos en host8 a modo de cambiar su default gateway para que este sea eth0 de host4.

```
ip route del default
ip route add default via 130.55.48.2
```

Luego, si enviamos un ping a la Red B desde host8 y observamos la captura de wireshark

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	130.55.48.1	224.0.0.9	RIPv2	266	Response
2	7.237063	42:00:aa:00:00:18	Broadcast	ARP	42	Who has 130.55.48.2? Tell 130.55.48.3
3	7.237311	42:00:aa:00:00:0f	42:00:aa:00:00:18	ARP	42	130.55.48.2 is at 42:00:aa:00:00:0f
4	7.237318	130.55.48.3	130.55.48.66	ICMP	442	Echo (ping) request id=0x008f, seq=1/256, ttl=64 (reply in 9)
5	7.237468	130.55.48.2	130.55.48.3	ICMP	470	Redirect (Redirect for host)
6	7.237493	42:00:aa:00:00:0f	Broadcast	ARP	42	Who has 130.55.48.1? Tell 130.55.48.2
7	7.237874	42:00:aa:00:00:0e	Broadcast	ARP	42	Who has 130.55.48.3? Tell 130.55.48.1
8	7.237884	42:00:aa:00:00:18	42:00:aa:00:00:0e	ARP	42	130.55.48.3 is at 42:00:aa:00:00:0e
9	7.237941	130.55.48.66	130.55.48.3	ICMP	442	Echo (ping) reply id=0x008f, seq=1/256, ttl=62 (request in 4)
10	12.421714	42:00:aa:00:00:0f	42:00:aa:00:00:18	ARP	42	Who has 130.55.48.3? Tell 130.55.48.2
11	12.421718	42:00:aa:00:00:18	42:00:aa:00:00:0f	ARP	42	130.55.48.3 is at 42:00:aa:00:00:18
12	16.048837	fe80::4000:aaff:fe0...	ff02::9	RIPng	106	Command Response, Version 1

```
IMUNES: host8 (console) bash
root@host8:/# ip route del default
root@host8:/# ip route add default via 130.55.48.2
root@host8:/# ping -c 1 -s 400 130.55.48.66
PING 130.55.48.66 (130.55.48.66) 400(428) bytes of data.
From 130.55.48.2: icmp_seq=1 Redirect Host(New nexthop: 130.55.48.1)
408 bytes from 130.55.48.66: icmp_seq=1 ttl=62 time=0.922 ms

--- 130.55.48.66 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.922/0.922/0.922/0.000 ms
root@host8:/#
```

Vemos como estamos recibiendo un ICMP Redirect de parte de host 4, que es exactamente lo que esperabamos.