

# Trabajo Práctico Integrador

Link al repositorio de GitHub: [julianasoto5/Redes2TPI](https://github.com/julianasoto5/Redes2TPI)

BLOQUES IP	
IPv4 (32 bits)	IPv6 (128 bits)
14.200.8.0/21 00001110.11001000.00001xxx.xxxxxxxx	200A:b8d:1014::/48
$2^{11}-2 = 2046$ hosts	$2^{80}$ hosts

## Práctica 3 - Ejercicio 12

Resolver el direccionamiento IPv4 con el bloque IP asignado.

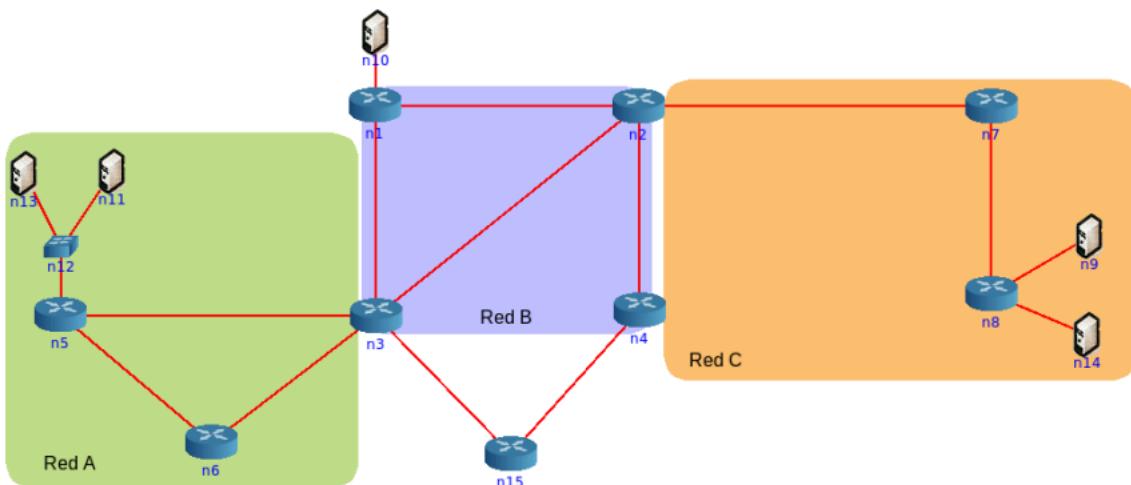


Figura 6: Diagrama de topología a entregar

Cada interfaz de un router tiene su propia IP

### RED A:

$$A.1 \rightarrow (n5-n11-n12-n13)$$

red 328 hosts

interfaz n5

Redes Backbone /30

$$A.2 \rightarrow (n5-n6): 2 \text{ interfaces}$$

$$A.3 \rightarrow (n5-n3): 2 \text{ interfaces}$$

$$A.4 \rightarrow (n3-n6): 2 \text{ interfaces}$$

**Total direcciones: 335 direcciones**

**RED B:**

- B.1 (n1-n2): 2 interfaces
- B.2 (n1-n3): 2 interfaces
- B.3 (n1-n10): 2 interfaces
- B.4 (n2 - n3): 2 interfaces
- B.5 (n2-n4): 2 interfaces
- B.6 (n3-n15): 2 interfaces
- B.7 (n4-n15): 2 interfaces

**Total direcciones: 14 direcciones**

**RED C:**

- C.1 (n8-n9): 501 direcciones
  - red 500 hosts
  - interfaz n8
- C.2 (n8-n14): 41 direcciones
  - red 40 hosts
  - interfaz n8
- C.3 (n2-n7): 2 interfaces
- C.4 (n7-n8): 2 interfaces

**Total direcciones: 546 direcciones**

Se le asignó a cada red un bloque de direcciones IP:

RED A: 14.200.8.0/23 y 14.200.10.0/24

RED B: 14.200.11.0/24

RED C: 14.200.12.0/22

Red A:

- A.1 14.200.8.0/23
- A.2 14.200.10.0/30
- A.3 14.200.10.4/30
- A.4 14.200.10.8/30

<b>RED A 14.200.8.0/23 + 14.200.10.0/24</b>		
<b>Red A.1 14.200.8.0/23</b>		
<b>Dispositivo</b>	<b>Interfaz</b>	<b>IPv4</b>
n11	eth0	14.200.8.2/23
n13	eth0	14.200.8.3/23
n5	eth2	14.200.8.1/23
<b>Red A.2 14.200.10.0/30</b>		

<b>Dispositivo</b>	<b>Interfaz</b>	<b>IPv4</b>
n5	eth0	14.200.10.1/30
n6	eth0	14.200.10.2/30
<b>Red A.3 14.200.10.4/30</b>		
<b>Dispositivo</b>	<b>Interfaz</b>	<b>IPv4</b>
n5	eth1	14.200.10.5/30
n3	eth1	14.200.10.6/30
<b>Red A.4 14.200.10.8/30</b>		
<b>Dispositivo</b>	<b>Interfaz</b>	<b>IPv4</b>
n3	eth0	14.200.10.9/30
n6	eth1	14.200.10.10/30

Red B:

- B.1 14.200.11.0/30
- B.2 14.200.11.4/30
- B.3 14.200.11.8/30
- B.4 14.200.11.12/30
- B.5 14.200.11.16/30
- B.6 14.200.11.20/30
- B.7 14.200.11.24/30

<b>RED B 14.200.11.0/24</b>		
<b>Red B.1 14.200.11.0/30</b>		
<b>Dispositivo</b>	<b>Interfaz</b>	<b>IPv4</b>
n1	eth2	14.200.11.1/30
n2	eth2	14.200.11.2/30
<b>Red B.2 14.200.11.4/30</b>		
<b>Dispositivo</b>	<b>Interfaz</b>	<b>IPv4</b>
n1	eth1	14.200.11.5/30
n3	eth3	14.200.11.6/30
<b>Red B.3 14.200.11.8/30</b>		

<b>Dispositivo</b>	<b>Interfaz</b>	<b>IPv4</b>
n1	eth0	14.200.11.9/30
n10	eth0	14.200.11.10/30
<b>Red B.4 14.200.11.12/30</b>		
<b>Dispositivo</b>	<b>Interfaz</b>	<b>IPv4</b>
n2	eth3	14.200.11.13/30
n3	eth2	14.200.11.14/30
<b>Red B.5 14.200.11.16/30</b>		
<b>Dispositivo</b>	<b>Interfaz</b>	<b>IPv4</b>
n2	eth1	14.200.11.17/30
n4	eth1	14.200.11.18/30
<b>Red B.6 14.200.11.20/30</b>		
<b>Dispositivo</b>	<b>Interfaz</b>	<b>IPv4</b>
n3	eth4	14.200.11.21/30
n15	eth1	14.200.11.22/30
<b>Red B.7 14.200.11.24/30</b>		
<b>Dispositivo</b>	<b>Interfaz</b>	<b>IPv4</b>
n4	eth0	14.200.11.25/30
n15	eth0	14.200.11.26/30

Red C:

- C.1 14.200.12.0/23
- C.2 14.200.14.0/26
- C.3 14.200.14.64/30
- C.4 14.200.14.68/30

<b>RED C 14.200.12.0/22</b>		
<b>Red C.1 14.200.12.0/23</b>		
<b>Dispositivo</b>	<b>Interfaz</b>	<b>IPv4</b>
n8	eth0	14.200.12.1/23
n9	eth0	14.200.13.2/23

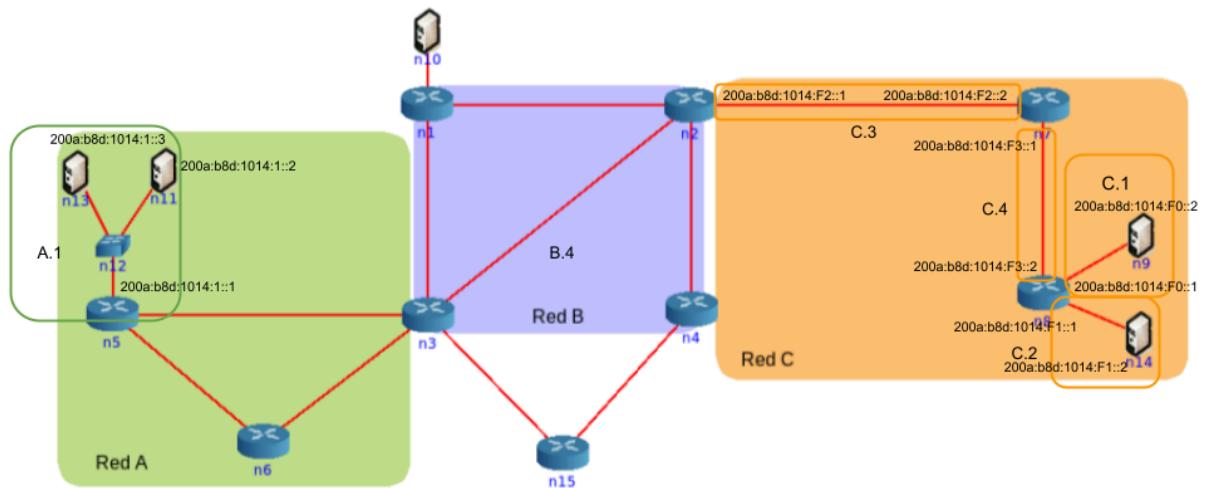
<b>Red C.2</b> 14.200.14.0/26		
<b>Dispositivo</b>	<b>Interfaz</b>	<b>IPv4</b>
n8	eth2	14.200.14.1/26
n14	eth0	14.200.14.2/26
<b>Red C.3</b> 14.200.14.64/30		
<b>Dispositivo</b>	<b>Interfaz</b>	<b>IPv4</b>
n2	eth0	14.200.14.65/30
n7	eth0	14.200.14.66/30
<b>Red C.4</b> 14.200.14.68/30		
<b>Dispositivo</b>	<b>Interfaz</b>	<b>IPv4</b>
n7	eth1	14.200.14.69/30
n8	eth1	14.200.14.70/30

**a) Configurar la red detrás de n5 (n5, n13, n11) y la Red C con el bloque IPv6 asignado.**

200A:b8d:1014::/48  
 200A:0b8d:1024:0000:0000:0000:0000/48  
 200A:0b8d:1024:xxxx:0000:0000:0000:0000/64

Cada subred utiliza /64, que es el estándar en IPv6 para garantizar suficiente espacio de direccionamiento.

Asignación de bloques de dirección IPv6



Red A.1: 200A:b8d:1014:1::/64

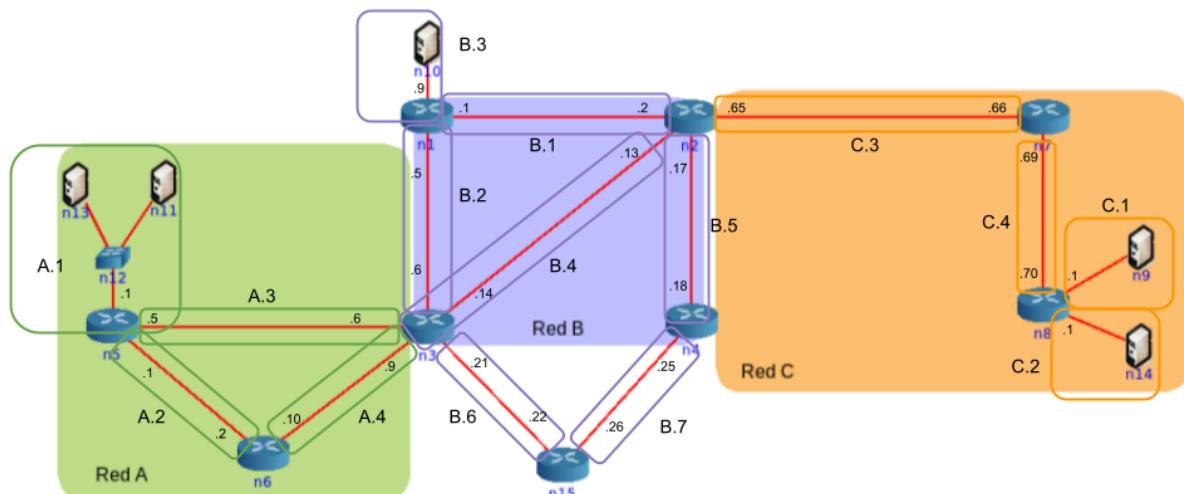
Red C: 200A:b8d:1014:F0::/60

RED A		
Red A.1 200A:b8d:1014:1::/64		
Dispositivo	Interfaz	IPv6
n11	eth0	200A:b8d:1014:1::2/64
n13	eth0	200A:b8d:1014:1::3/64
n5	eth2	200A:b8d:1014:1::1/64

RED C 200A:b8d:1014:F0::/60		
Red C.1 200A:b8d:1014:F0::/64		
Dispositivo	Interfaz	IPv6
n8	eth0	200A:b8d:1014:F0::1/64
n9	eth0	200A:b8d:1014:F0::2/64
Red C.2 200A:b8d:1014:F1::/64		
Dispositivo	Interfaz	IPv6
n8	eth2	200A:b8d:1014:F1::1/64

n14	eth0	200A:b8d:1014:F1::2/64
<b>Red C.3 200A:b8d:1014:F2::/64</b>		
<b>Dispositivo</b>	<b>Interfaz</b>	<b>IPv6</b>
n2	eth0	200A:b8d:1014:F2::1/64
n7	eth0	200A:b8d:1014:F2::2/64
<b>Red C.4 200A:b8d:1014:F3::/64</b>		
<b>Dispositivo</b>	<b>Interfaz</b>	<b>IPv6</b>
n7	eth1	200A:b8d:1014:F3::1/64
n8	eth1	200A:b8d:1014:F3::2/64

### b) Resolver con ruteo estático la topología.



Se muestra en primer lugar una tabla de ruteo detallada, donde aparecen todas las redes de la topología y luego, se simplificaron a través de la salida gateway o a partir de la summarización de redes. Las redes que se simplificaron a través del gateway se escribieron en rojo, y las que fueron summarizadas en naranja.

Router n5 - detallada				
Destino	Dir Red Destino	Máscara	Next-Hop	Interfaz
Red A.1	14.200.8.0	/23	-	eth2
Red A.2	14.200.10.0	/30	-	eth0

Red A.3	14.200.10.4	/30	-	eth1
Red A.4	14.200.10.8	/30	14.200.10.6	eth1
Red B.1	14.200.11.0	/30	14.200.10.6	eth1
Red B.2	14.200.11.4	/30	14.200.10.6	eth1
Red B.3	14.200.11.8	/30	14.200.10.6	eth1
Red B.4	14.200.11.12	/30	14.200.10.6	eth1
Red B.5	14.200.11.16	/30	14.200.10.6	eth1
Red B.6	14.200.11.20	/30	14.200.10.6	eth1
Red B.7	14.200.11.24	/30	14.200.10.6	eth1
Red C.1	14.200.12.0	/23	14.200.10.6	eth1
Red C.2	14.200.14.0	/26	14.200.10.6	eth1
Red C.3	14.200.14.64	/30	14.200.10.6	eth1
Red C.4	14.200.14.68	/30	14.200.10.6	eth1

Router n5				
Destino	Dir Red Destino	Máscara	Next-Hop	Interfaz
Red A.1	14.200.8.0	/23	-	eth2
Red A.2	14.200.10.0	/30	-	eth0
Red A.3	14.200.10.4	/30	-	eth1
Gateway	0.0.0.0	/0	14.200.10.6/30	eth1

Router n6 - detallada				
Destino	Dir Red Destino	Máscara	Next-Hop	Interfaz
Red A.1	14.200.8.0	/23	14.200.10.1	eth0
Red A.2	14.200.10.0	/30	-	eth0
Red A.3	14.200.10.4	/30	14.200.10.9	eth1
Red A.4	14.200.10.8	/30	-	eth1
Red B.1	14.200.11.0	/30	14.200.10.9	eth1
Red B.2	14.200.11.4	/30	14.200.10.9	eth1

Red B.3	14.200.11.8	/30	14.200.10.9	eth1
Red B.4	14.200.11.12	/30	14.200.10.9	eth1
Red B.5	14.200.11.16	/30	14.200.10.9	eth1
Red B.6	14.200.11.20	/30	14.200.10.9	eth1
Red B.7	14.200.11.24	/30	14.200.10.9	eth1
Red C.1	14.200.12.0	/23	14.200.10.9	eth1
Red C.2	14.200.14.0	/26	14.200.10.9	eth1
Red C.3	14.200.14.64	/30	14.200.10.9	eth1
Red C.4	14.200.14.68	/30	14.200.10.9	eth1

Router n6				
Destino	Dir Red Destino	Máscara	Next-Hop	Interfaz
Red A.1	14.200.8.0	/23	14.200.10.1	eth0
Red A.2	14.200.10.0	/30	-	eth0
Red A.4	14.200.10.8	/30	-	eth1
Gateway	0.0.0.0	0	14.200.10.9	eth1

Router n3 - detallada				
Destino	Dir Red Destino	Máscara	Next-Hop	Interfaz
Red A.1	14.200.8.0	/23	14.200.10.5	eth1
Red A.2	14.200.10.0	/30	14.200.10.5	eth1
Red A.3	14.200.10.4	/30	-	eth1
Red A.4	14.200.10.8	/30	-	eth0
Red B.1	14.200.11.0	/30	14.200.11.13	eth2
Red B.2	14.200.11.4	/30	-	eth3
Red B.3	14.200.11.8	/30	14.200.11.5	eth1
Red B.4	14.200.11.12	/30	-	eth2
Red B.5	14.200.11.16	/30	14.200.11.13	eth2
Red B.6	14.200.11.20	/30	-	eth4

Red B.7	14.200.11.24	/30	14.200.11.22	eth4
Red C.1	14.200.12.0	/23	14.200.11.13	eth2
Red C.2	14.200.14.0	/26	14.200.11.13	eth2
Red C.3	14.200.14.64	/30	14.200.11.13	eth2
Red C.4	14.200.14.68	/30	14.200.11.13	eth2

Router n3				
Destino	Dir Red Destino	Máscara	Next-Hop	Interfaz
Red A.1	14.200.8.0	/23	14.200.10.5/30	eth1
Red A.2	14.200.10.0	/30	14.200.10.5/30	eth1
Red A.3	14.200.10.4	/30	-	eth1
Red A.4	14.200.10.8	/30	-	eth0
Red B.2	14.200.11.4	/30	-	eth2
Red B.3	14.200.11.8	/30	14.200.11.5	eth2
Red B.4	14.200.11.12	/30	-	eth3
Red B.6	14.200.11.20	/30	-	eth4
Red B.7	14.200.11.24	/30	14.200.11.22	eth4
Gateway	0.0.0.0	/0	14.200.11.13	eth3

Router n1 - detallada				
Destino	Dir Red Destino	Máscara	Next-Hop	Interfaz
Red A.1	14.200.8.0	/23	14.200.11.6	eth1
Red A.2	14.200.10.0	/30	14.200.11.6	eth1
Red A.3	14.200.10.4	/30	14.200.11.6	eth1
Red A.4	14.200.10.8	/30	14.200.11.6	eth1
Red B.1	14.200.11.0	/30	-	eth2
Red B.2	14.200.11.4	/30	-	eth1
Red B.3	14.200.11.8	/30	-	eth0
Red B.4	14.200.11.12	/30	14.200.11.6	eth1

Red B.5	14.200.11.16	/30	14.200.11.2	eth2
Red B.6	14.200.11.20	/30	14.200.11.6	eth1
Red B.7	14.200.11.24	/30	14.200.11.6	eth1
Red C.1	14.200.12.0	/23	14.200.11.2	eth2
Red C.2	14.200.14.0	/26	14.200.11.2	eth2
Red C.3	14.200.14.64	/30	14.200.11.2	eth2
Red C.4	14.200.14.68	/30	14.200.11.2	eth2

Router n1				
Destino	Dir Red Destino	Máscara	Next-Hop	Interfaz
Red B.1	14.200.11.0	/30	-	eth2
Red B.2	14.200.11.4	/30	-	eth0
Red B.3	14.200.11.8	/30	-	eth1
Red B.5	14.200.11.16	/30	14.200.11.2	eth2
Red C.1	14.200.12.0	/23	14.200.11.2	eth2
Red C.2	14.200.14.0	/26	14.200.11.2	eth2
Red C.3 - Red C.4	14.200.14.64	/29	14.200.11.2	eth2
Gateway	0.0.0.0	/0	14.200.11.6	eth0

Router n2 - detallada				
Destino	Dir Red Destino	Máscara	Next-Hop	Interfaz
Red A.1	14.200.8.0	/23	14.200.11.14	eth2
Red A.2	14.200.10.0	/30	14.200.11.14	eth2
Red A.3	14.200.10.4	/30	14.200.11.14	eth2
Red A.4	14.200.10.8	/30	14.200.11.14	eth2
Red B.1	14.200.11.0	/30	-	eth1
Red B.2	14.200.11.4	/30	14.200.11.14	eth2
Red B.3	14.200.11.8	/30	14.200.11.1	eth1

Red B.4	14.200.11.12	/30	-	eth2
Red B.5	14.200.11.16	/30	-	eth0
Red B.6	14.200.11.20	/30	14.200.11.14	eth3
Red B.7	14.200.11.24	/30	14.200.11.18	eth0
Red C.1	14.200.12.0	/23	14.200.14.66	eth3
Red C.2	14.200.14.0	/26	14.200.14.66	eth3
Red C.3	14.200.14.64	/30	-	
Red C.4	14.200.14.68	/30	14.200.14.66	eth3

Router n2				
Destino	Dir Red Destino	Máscara	Next-Hop	Interfaz
Red B.1	14.200.11.0	/30	-	eth1
Red B.2	14.200.11.12	/30	-	eth2
Red B.5	14.200.11.16	/30	-	eth0
Red C.1	14.200.12.0	/23	14.200.14.66	eth3
Red C.2	14.200.14.0	/26	14.200.14.66	eth3
Red C.3	14.200.14.64	/30	-	eth3
Red C.4	14.200.14.68	/30	14.200.14.66	eth3
Gateway	0.0.0.0	/0	14.200.11.14	eth2

Router n4 - detallada				
Destino	Dir Red Destino	Máscara	Next-Hop	Interfaz
Red A.1	14.200.8.0	/23	14.200.11.17	eth1
Red A.2	14.200.10.0	/30	14.200.11.17	eth1
Red A.3	14.200.10.4	/30	14.200.11.17	eth1
Red A.4	14.200.10.8	/30	14.200.11.17	eth1
Red B.1	14.200.11.0	/30	14.200.11.17	eth1
Red B.2	14.200.11.4	/30	14.200.11.17	eth1
Red B.3	14.200.11.8	/30	14.200.11.17	eth1

Red B.4	14.200.11.12	/30	14.200.11.17	eth1
Red B.5	14.200.11.16	/30	-	eth1
Red B.6	14.200.11.20	/30	14.200.11.26	eth0
Red B.7	14.200.11.24	/30	-	eth0
Red C.1	14.200.12.0	/23	14.200.11.17	eth1
Red C.2	14.200.14.0	/26	14.200.11.17	eth1
Red C.3	14.200.14.64	/30	14.200.11.17	eth1
Red C.4	14.200.14.68	/30	14.200.11.17	eth1

Router n4				
Destino	Dir Red Destino	Máscara	Next-Hop	Interfaz
Red B.5	14.200.11.16	/30	-	eth1
Red B.6	14.200.11.20	/30	14.200.11.26	eth0
Red B.7	14.200.11.24	/30	-	eth0
Gateway	0.0.0.0	/0	14.200.11.17	eth1

Router n15 - detallada				
Destino	Dir Red Destino	Máscara	Next-Hop	Interfaz
Red A.1	14.200.8.0	/30	14.200.11.21	eth1
Red A.2	14.200.10.0	/30	14.200.11.21	eth1
Red A.3	14.200.10.4	/30	14.200.11.21	eth1
Red A.4	14.200.10.8	/30	14.200.11.21	eth1
Red B.1	14.200.11.0	/30	14.200.11.21	eth1
Red B.2	14.200.11.4	/30	14.200.11.21	eth1
Red B.3	14.200.11.8	/30	14.200.11.21	eth1
Red B.4	14.200.11.12	/30	14.200.11.21	eth1
Red B.5	14.200.11.16	/30	14.200.11.25	eth0
Red B.6	14.200.11.20	/30	-	eth1
Red B.7	14.200.11.24	/30	-	eth0

Red C.1	14.200.12.0	/23	14.200.11.21	eth1
Red C.2	14.200.14.0	/26	14.200.11.21	eth1
Red C.3	14.200.14.64	/30	14.200.11.21	eth1
Red C.4	14.200.14.68	/30	14.200.11.21	eth1

Router n15				
Destino	Dir Red Destino	Máscara	Next-Hop	Interfaz
Red B.5	14.200.11.16	/30	14.200.11.25	eth0
Red B.6	14.200.11.20	/30	-	eth1
Red B.7	14.200.11.24	/30	-	eth0
Gateway	0.0.0.0	/0	14.200.11.21	eth1

Router n7 - detallada				
Destino	Dir Red Destino	Máscara	Next-Hop	Interfaz
Red A.1	14.200.8.0	/23	14.200.14.65	eth0
Red A.2	14.200.10.0	/30	14.200.14.65	eth0
Red A.3	14.200.10.4	/30	14.200.14.65	eth0
Red A.4	14.200.10.8	/30	14.200.14.65	eth0
Red B.1	14.200.11.0	/30	14.200.14.65	eth0
Red B.2	14.200.11.4	/30	14.200.14.65	eth0
Red B.3	14.200.11.8	/30	14.200.14.65	eth0
Red B.4	14.200.11.12	/30	14.200.14.65	eth0
Red B.5	14.200.11.16	/30	14.200.14.65	eth0
Red B.6	14.200.11.20	/30	14.200.14.65	eth0
Red B.7	14.200.11.24	/30	14.200.14.65	eth0
Red C.1	14.200.12.0	/23	14.200.14.70	eth1
Red C.2	14.200.14.0	/26	14.200.14.70	eth1
Red C.3	14.200.14.64	/30	-	eth0
Red C.4	14.200.14.68	/30	-	eth1

Gateway	0.0.0.0	/0		
---------	---------	----	--	--

Router n7				
Destino	Dir Red Destino	Máscara	Next-Hop	Interfaz
Red C.1	14.200.12.0	/23	14.200.14.70	eth1
Red C.2	14.200.14.0	/26	14.200.14.70	eth1
Red C.3	14.200.14.64	/30	-	eth0
Red C.4	14.200.14.68	/30	-	eth1
Gateway	0.0.0.0	/0	14.200.14.65	eth0

Router n8 - detallada				
Destino	Dir Red Destino	Máscara	Next-Hop	Interfaz
Red A.1	14.200.8.0	/23	14.200.14.69	eth1
Red A.2	14.200.10.0	/30	14.200.14.69	eth1
Red A.3	14.200.10.4	/30	14.200.14.69	eth1
Red A.4	14.200.10.8	/30	14.200.14.69	eth1
Red B.1	14.200.11.0	/30	14.200.14.69	eth1
Red B.2	14.200.11.4	/30	14.200.14.69	eth1
Red B.3	14.200.11.8	/30	14.200.14.69	eth1
Red B.4	14.200.11.12	/30	14.200.14.69	eth1
Red B.5	14.200.11.16	/30	14.200.14.69	eth1
Red B.6	14.200.11.20	/30	14.200.14.69	eth1
Red B.7	14.200.11.24	/30	14.200.14.69	eth1
Red C.1	14.200.12.0	/23	-	eth0
Red C.2	14.200.14.0	/26	-	eth2
Red C.3	14.200.14.64	/30	14.200.14.69	eth1
Red C.4	14.200.14.68	/30	-	eth1

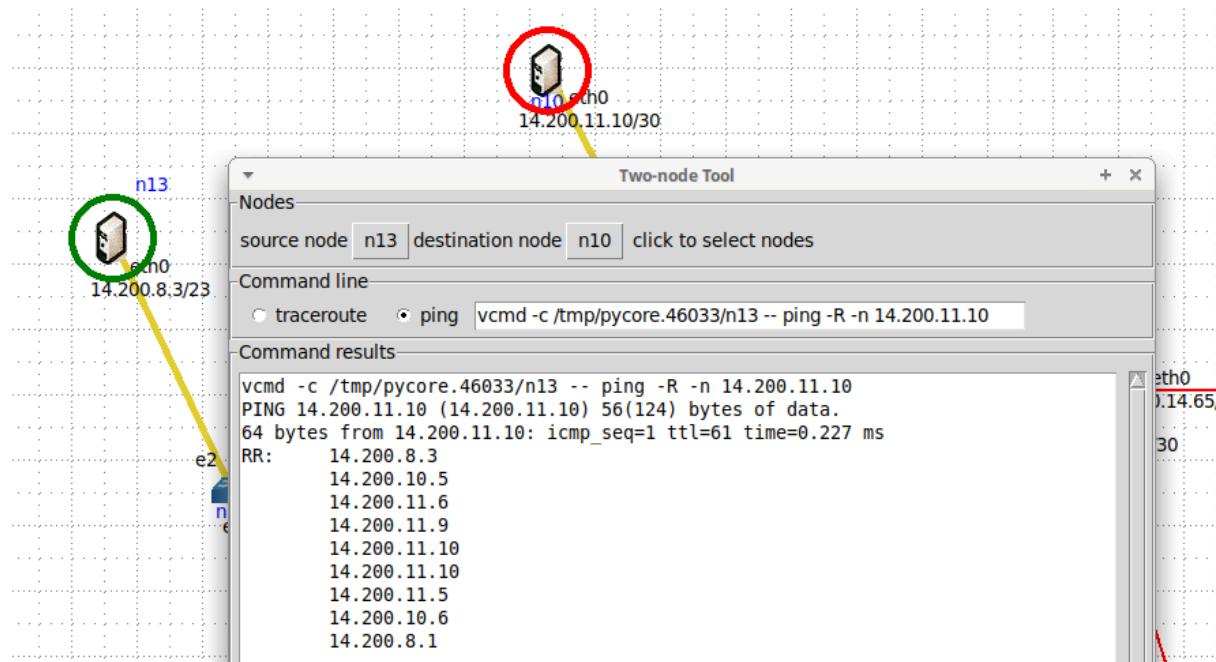
Router n8				
-----------	--	--	--	--

Destino	Dir Red Destino	Máscara	Next-Hop	Interfaz
Red C.1	14.200.12.0	/23	-	eth0
Red C.2	14.200.14.0	/26	-	eth2
Red C.4	14.200.14.68	/30	-	eth1
Gateway	0.0.0.0	/0	14.200.14.69	eth1

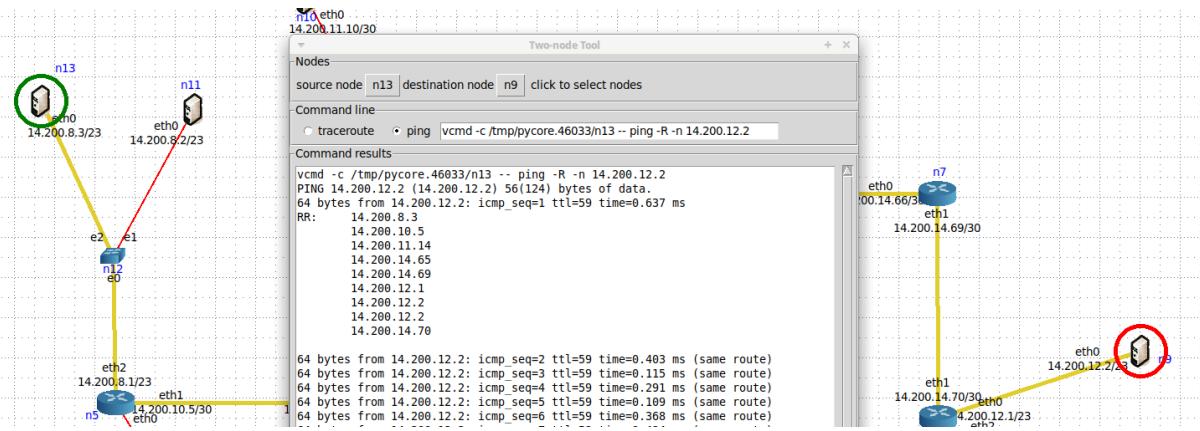
**d) Realizar test con ping (ICMP) y traceroute para probar que funciona la topología.**

Para comprobar la topología se realizaron test con ping y traceroute entre los hosts n13 y n10, y n13 y n9.

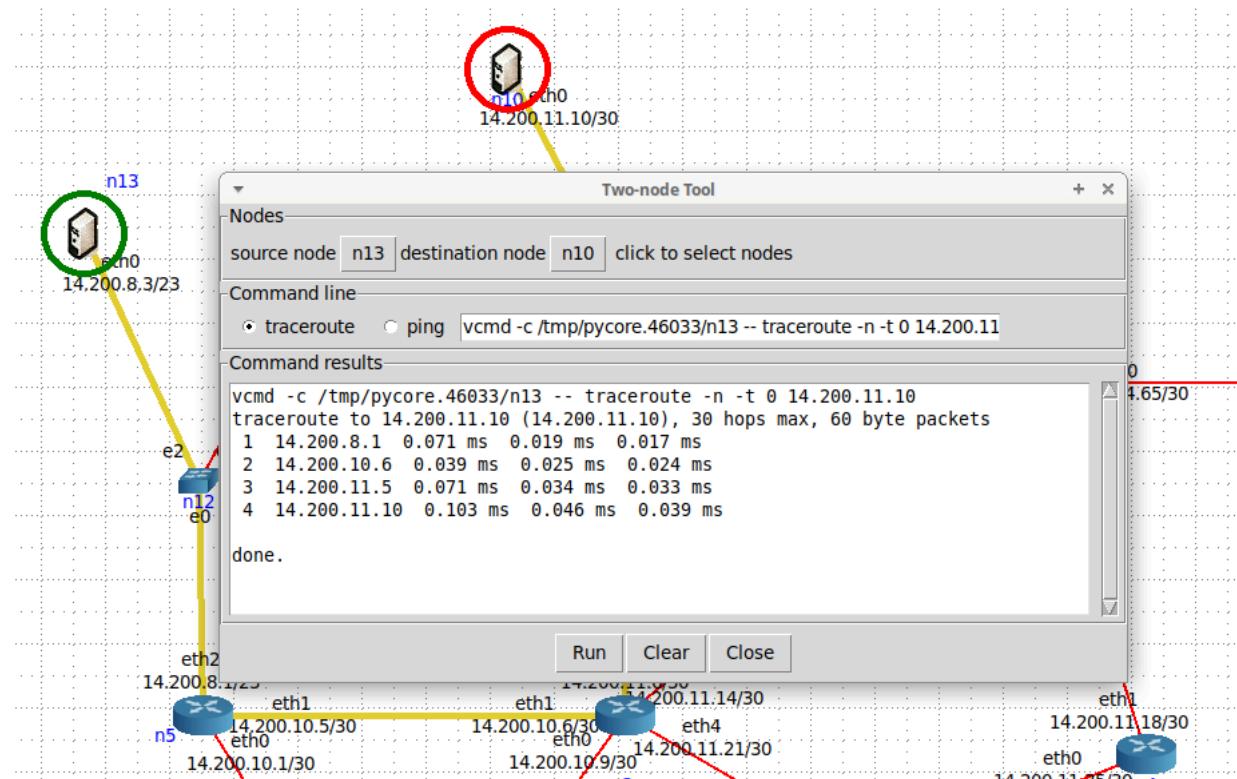
Primero se utilizó el comando “ping -R -n 14.200.11.100” desde n13. Se observa una lista de las direcciones IPv4 que siguen los paquetes primero de n13 a n10 (echo request) y luego de n10 a n13 (echo reply).



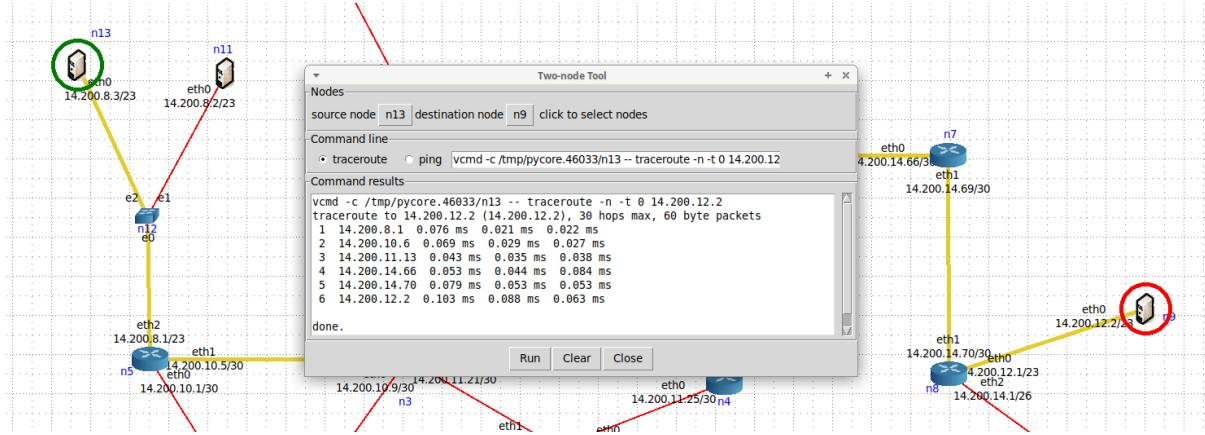
Luego se usó el comando “ping -R -n 14.200.11.100” desde n13. También se observa la lista de las direcciones IPv4 que siguen los paquetes primero de n13 a n9 (echo request). En este caso solo es posible ver las primeras direcciones que sigue el “echo reply” que envía n9 porque el comando solo permite ver hasta 9 direcciones.



A continuación se utiliza desde n13 el comando “traceroute -n -t 0 14.200.11.10”. Se pueden ver las direcciones IPv4 de las interfaces de los routers que se van atravesando para llegar de n13 a n10, junto con los tiempos de ida y de vuelta de los mensajes entre cada router.



Finalmente, se usa el comando “traceroute -n -t 0 14.200.12.2” desde n13 y se observan los routers que forman el camino hacia n9.



### e) Capturar puntualmente el tráfico de n13 hacia n7 y analizar: ARP e ICMP.

Tras ejecutar el comando “ping 14.200.14.66” y examinar el tráfico de paquetes en esta misma interfaz, se observó lo siguiente:

No.	Time	Source	Destination	Protocol	Length Info
1	0.000000000	00:00:00_aa:00:13	Broadcast	ARP	42 Who has 14.200.14.66? Tell 14.200.14.66
2	0.0000008370	00:00:00_aa:00:14	00:00:00_aa:00:13	ARP	42 14.200.14.66 is at 00:00:00:aa:00:14
3	0.0000018302	14.200.8.3	14.200.14.66	ICMP	138 Echo (ping) request id=0x0084, seq=1/256, ttl=61 (req)
4	0.0000024494	14.200.14.66	14.200.8.3	ICMP	138 Echo (ping) reply id=0x0084, seq=1/256, ttl=64 (rep)
5	1.022032451	14.200.8.3	14.200.14.66	ICMP	138 Echo (ping) request id=0x0084, seq=2/512, ttl=61 (req)
6	1.022042531	14.200.14.66	14.200.8.3	ICMP	138 Echo (ping) reply id=0x0084, seq=2/512, ttl=64 (rep)
7	2.046291910	14.200.8.3	14.200.14.66	ICMP	138 Echo (ping) request id=0x0084, seq=3/768, ttl=61 (req)
8	2.046302833	14.200.14.66	14.200.8.3	ICMP	138 Echo (ping) reply id=0x0084, seq=3/768, ttl=64 (rep)
9	3.070001330	14.200.8.3	14.200.14.66	ICMP	138 Echo (ping) request id=0x0084, seq=4/1024, ttl=61 (req)
10	3.0700012704	14.200.14.66	14.200.8.3	ICMP	138 Echo (ping) reply id=0x0084, seq=4/1024, ttl=64 (rep)
11	4.094436415	14.200.8.3	14.200.14.66	ICMP	138 Echo (ping) request id=0x0084, seq=5/1280, ttl=61 (req)
12	4.094456034	14.200.14.66	14.200.8.3	ICMP	138 Echo (ping) reply id=0x0084, seq=5/1280, ttl=64 (rep)
13	5.118480289	14.200.8.3	14.200.14.66	ICMP	138 Echo (ping) request id=0x0084, seq=6/1536, ttl=61 (req)
14	5.118493074	14.200.14.66	14.200.8.3	ICMP	138 Echo (ping) reply id=0x0084, seq=6/1536, ttl=64 (rep)
15	5.182060988	00:00:00_aa:00:14	00:00:00_aa:00:13	ARP	42 Who has 14.200.14.65? Tell 14.200.14.66
16	5.182159094	00:00:00_aa:00:13	00:00:00_aa:00:14	ARP	42 14.200.14.65 is at 00:00:00:aa:00:13
17	6.142213173	14.200.8.3	14.200.14.66	ICMP	138 Echo (ping) request id=0x0084, seq=7/1792, ttl=61 (req)
18	6.142226620	14.200.14.66	14.200.8.3	ICMP	138 Echo (ping) reply id=0x0084, seq=7/1792, ttl=64 (rep)

Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface veth7.0.b6, id 0  
 ▾ EtherType II, Src: 00:00:00\_aa:00:13 (00:00:00:aa:00:13), Dst: Broadcast (ff:ff:ff:ff:ff:ff)  
 ▾ Address Resolution Protocol (request)  
   Hardware type: Ethernet (1)  
   Protocol type: IPv4 (0x0800)  
   Hardware size: 6  
   Protocol size: 4  
   Opcode: request (1)  
   Sender MAC address: 00:00:00\_aa:00:13 (00:00:00:aa:00:13)  
   Sender IP address: 14.200.14.65  
   Target MAC address: 00:00:00\_00:00:00 (00:00:00:00:00:00)  
   Target IP address: 14.200.14.66

Los primeros dos paquetes capturados corresponden al protocolo ARP. El primero fue enviado por el router n2 (14.200.14.65) al router n7 (14.200.14.66), preguntando por su dirección MAC. Puede observarse que el destino del primer paquete es “Broadcast”, ya que el origen no sabe a qué MAC enviarlo. En los headers de los paquetes ARP, se pueden apreciar los siguientes campos:

- “Hardware size” (longitud de dirección MAC): 6 para direcciones MAC de dispositivos que corresponden a una red del protocolo IEEE 802.
- “Protocol size” (longitud de IP): 4 para direcciones IPv4.
- “Opcode”: 1 es ARP request y 2 es ARP reply.
- “Hardware Sender Address” y “Protocol Sender Address”: dirección MAC y dirección IPv4 del dispositivo que envió el mensaje

- “Hardware Target Address” y “Protocol Target Address”: dirección MAC y dirección IPv4 del dispositivo destino. La MAC inicialmente se coloca en un valor nulo, porque es para que el destino del paquete ARP lo complete con su MAC.

El segundo paquete ARP es la respuesta de n7 a n2. Los campos “hardware type/size” y “protocol type/size” son los mismos en un ARP request que en un reply. Los que varían son los campos “opcode” (ahora con valor 2 para indicar ARP reply) y las direcciones MAC (hardware) e IP (protocolo) de las estaciones origen y destino, las cuales se intercambian. La MAC que era 00:00:00:00:00:00 en el ARP request ahora lleva una dirección verdadera, completada por n7.

No.	Time	Source	Destination	Protocol	Length Info
1	0.0000000000	00:00:00_aa:00:13	Broadcast	ARP	42 Who has 14.200.14.66? Tell 14.200.14.65
2	0.0000008370	00:00:00_aa:00:14	00:00:00_aa:00:13	ARP	42 14.200.14.66 is at 00:00:00_aa:00:14
3	0.0000018302	14.200.8.3	14.200.14.66	ICMP	138 Echo (ping) request id=0x0084, seq=1/256, ttl=61 (req)
4	0.0000024494	14.200.14.66	14.200.8.3	ICMP	138 Echo (ping) reply id=0x0084, seq=1/256, ttl=64 (rep)
5	1.022032451	14.200.8.3	14.200.14.66	ICMP	138 Echo (ping) request id=0x0084, seq=2/512, ttl=61 (req)
6	1.022042531	14.200.14.66	14.200.8.3	ICMP	138 Echo (ping) reply id=0x0084, seq=2/512, ttl=64 (rep)
7	2.046291910	14.200.8.3	14.200.14.66	ICMP	138 Echo (ping) request id=0x0084, seq=3/768, ttl=61 (req)
8	2.046302833	14.200.14.66	14.200.8.3	ICMP	138 Echo (ping) reply id=0x0084, seq=3/768, ttl=64 (rep)
9	3.070001330	14.200.8.3	14.200.14.66	ICMP	138 Echo (ping) request id=0x0084, seq=4/1024, ttl=61 (req)
10	3.070012704	14.200.14.66	14.200.8.3	ICMP	138 Echo (ping) reply id=0x0084, seq=4/1024, ttl=64 (rep)
11	4.0944436415	14.200.8.3	14.200.14.66	ICMP	138 Echo (ping) request id=0x0084, seq=5/1280, ttl=61 (req)
12	4.0944456034	14.200.14.66	14.200.8.3	ICMP	138 Echo (ping) reply id=0x0084, seq=5/1280, ttl=64 (rep)
13	5.118480289	14.200.8.3	14.200.14.66	ICMP	138 Echo (ping) request id=0x0084, seq=6/1536, ttl=61 (req)
14	5.118493074	14.200.14.66	14.200.8.3	ICMP	138 Echo (ping) reply id=0x0084, seq=6/1536, ttl=64 (rep)
15	5.182060980	00:00:00_aa:00:14	00:00:00_aa:00:13	ARP	42 Who has 14.200.14.65? Tell 14.200.14.66
16	5.182159094	00:00:00_aa:00:13	00:00:00_aa:00:14	ARP	42 14.200.14.65 is at 00:00:00_aa:00:13
17	6.142213173	14.200.8.3	14.200.14.66	ICMP	138 Echo (ping) request id=0x0084, seq=7/1792, ttl=61 (req)
18	6.142226620	14.200.14.66	14.200.8.3	ICMP	138 Echo (ping) reply id=0x0084, seq=7/1792, ttl=64 (rep)

Frame 2: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface veth7.0.b6, id 0  
 ▷ Ethernet II, Src: 00:00:00\_aa:00:14 (00:00:00\_aa:00:14), Dst: 00:00:00\_aa:00:13 (00:00:00\_aa:00:13)  
 ▷ Address Resolution Protocol (reply)  
 Hardware type: Ethernet (1)  
 Protocol type: IPv4 (0x0800)  
 Hardware size: 6  
 Protocol size: 4  
 Opcode: reply (2)  
 Sender MAC address: 00:00:00\_aa:00:14 (00:00:00\_aa:00:14)  
 Sender IP address: 14.200.14.66  
 Target MAC address: 00:00:00\_aa:00:13 (00:00:00\_aa:00:13)  
 Target IP address: 14.200.14.65

Tras haberse obtenido la dirección MAC del destino utilizando los paquetes ARP, se envía el primer paquete ICMP. El mismo tiene como origen 14.200.8.3 (n13) y como destino 14.200.14.66 (n7). Se puede reconocer que es un “echo request” por el campo “type”, que contiene un 8, y se lo identifica con un Identifier (ID) y Sequence Number (Seq) que permitirán asociar su “echo reply”.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	00:00:00_aa:00:13	Broadcast	ARP	42	Who has 14.200.14.66? Tell 14.200.14.65
2	0.0000008370	00:00:00_aa:00:14	00:00:00_aa:00:13	ARP	42	14.200.14.66 is at 00:00:00_aa:00:14
3	0.0000018302	14.200.8.3	14.200.14.66	ICMP	138	Echo (ping) request id=0x0084, seq=1/256, ttl=61 (req)
4	0.0000024494	14.200.14.66	14.200.8.3	ICMP	138	Echo (ping) reply id=0x0084, seq=1/256, ttl=64 (rep)
5	1.022032451	14.200.8.3	14.200.14.66	ICMP	138	Echo (ping) request id=0x0084, seq=2/512, ttl=61 (req)
6	1.022042531	14.200.14.66	14.200.8.3	ICMP	138	Echo (ping) reply id=0x0084, seq=2/512, ttl=64 (rep)
7	2.046291910	14.200.8.3	14.200.14.66	ICMP	138	Echo (ping) request id=0x0084, seq=3/768, ttl=61 (req)
8	2.046302833	14.200.14.66	14.200.8.3	ICMP	138	Echo (ping) reply id=0x0084, seq=3/768, ttl=64 (rep)
9	3.070001330	14.200.8.3	14.200.14.66	ICMP	138	Echo (ping) request id=0x0084, seq=4/1024, ttl=61 (req)
10	3.070012704	14.200.14.66	14.200.8.3	ICMP	138	Echo (ping) reply id=0x0084, seq=4/1024, ttl=64 (rep)
11	4.0944436415	14.200.8.3	14.200.14.66	ICMP	138	Echo (ping) request id=0x0084, seq=5/1280, ttl=61 (req)
12	4.094456034	14.200.14.66	14.200.8.3	ICMP	138	Echo (ping) reply id=0x0084, seq=5/1280, ttl=64 (rep)
13	5.118480289	14.200.8.3	14.200.14.66	ICMP	138	Echo (ping) request id=0x0084, seq=6/1536, ttl=61 (req)
14	5.118493074	14.200.14.66	14.200.8.3	ICMP	138	Echo (ping) reply id=0x0084, seq=6/1536, ttl=64 (rep)
15	5.182066980	00:00:00_aa:00:14	00:00:00_aa:00:13	ARP	42	Who has 14.200.14.65? Tell 14.200.14.66
16	5.182159094	00:00:00_aa:00:13	00:00:00_aa:00:14	ARP	42	14.200.14.65 is at 00:00:00_aa:00:13
17	6.142213173	14.200.8.3	14.200.14.66	ICMP	138	Echo (ping) request id=0x0084, seq=7/1792, ttl=61 (req)
18	6.142226620	14.200.8.3	14.200.8.3	ICMP	138	Echo (ping) reply id=0x0084, seq=7/1792, ttl=64 (rep)

Frame 3: 138 bytes on wire (1104 bits), 138 bytes captured (1104 bits) on interface veth7.0.b6, id 0

Ethernet II, Src: 00:00:00\_aa:00:13 (00:00:00:aa:00:13), Dst: 00:00:00\_aa:00:14 (00:00:00:aa:00:14)

Internet Protocol Version 4, Src: 14.200.8.3, Dst: 14.200.14.66

Internet Control Message Protocol

- Type: 8 (Echo (ping) request)
- Code: 0
- Checksum: 0xd04e [correct]
- [Checksum Status: Good]
- Identifier (BE): 132 (0x0084)
- Identifier (LE): 33792 (0x8400)
- Sequence number (BE): 1 (0x0001)
- Sequence number (LE): 256 (0x0100)

[Response frame: 4]

Timestamp from icmp data: Dec 11, 2024 18:49:36.000000000 -03

[Timestamp from icmp data (relative): 0.387534781 seconds]

Data (48 bytes)

A continuación, se recibe un segundo paquete ICMP type 0, con dirección destino 14.200.8.3 (n13) y dirección origen 14.200.14.66 (n7). Este es el “echo reply”, que tiene su ID y Seq iguales a las del paquete anterior.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	00:00:00_aa:00:13	Broadcast	ARP	42	Who has 14.200.14.66? Tell 14.200.14.65
2	0.0000008370	00:00:00_aa:00:14	00:00:00_aa:00:13	ARP	42	14.200.14.66 is at 00:00:00_aa:00:14
3	0.0000018302	14.200.8.3	14.200.14.66	ICMP	138	Echo (ping) request id=0x0084, seq=1/256, ttl=61 (req)
4	0.0000024494	14.200.14.66	14.200.8.3	ICMP	138	Echo (ping) reply id=0x0084, seq=1/256, ttl=64 (rep)
5	1.022032451	14.200.8.3	14.200.14.66	ICMP	138	Echo (ping) request id=0x0084, seq=2/512, ttl=61 (req)
6	1.022042531	14.200.14.66	14.200.8.3	ICMP	138	Echo (ping) reply id=0x0084, seq=2/512, ttl=64 (rep)
7	2.046291910	14.200.8.3	14.200.14.66	ICMP	138	Echo (ping) request id=0x0084, seq=3/768, ttl=61 (req)
8	2.046302833	14.200.14.66	14.200.8.3	ICMP	138	Echo (ping) reply id=0x0084, seq=3/768, ttl=64 (rep)
9	3.070001330	14.200.8.3	14.200.14.66	ICMP	138	Echo (ping) request id=0x0084, seq=4/1024, ttl=61 (req)
10	3.070012704	14.200.14.66	14.200.8.3	ICMP	138	Echo (ping) reply id=0x0084, seq=4/1024, ttl=64 (rep)
11	4.0944436415	14.200.8.3	14.200.14.66	ICMP	138	Echo (ping) request id=0x0084, seq=5/1280, ttl=61 (req)
12	4.094456034	14.200.14.66	14.200.8.3	ICMP	138	Echo (ping) reply id=0x0084, seq=5/1280, ttl=64 (rep)
13	5.118480289	14.200.8.3	14.200.14.66	ICMP	138	Echo (ping) request id=0x0084, seq=6/1536, ttl=61 (req)
14	5.118493074	14.200.14.66	14.200.8.3	ICMP	138	Echo (ping) reply id=0x0084, seq=6/1536, ttl=64 (rep)
15	5.182066980	00:00:00_aa:00:14	00:00:00_aa:00:13	ARP	42	Who has 14.200.14.65? Tell 14.200.14.66
16	5.182159094	00:00:00_aa:00:13	00:00:00_aa:00:14	ARP	42	14.200.14.65 is at 00:00:00_aa:00:13
17	6.142213173	14.200.8.3	14.200.14.66	ICMP	138	Echo (ping) request id=0x0084, seq=7/1792, ttl=61 (req)
18	6.142226620	14.200.8.3	14.200.8.3	ICMP	138	Echo (ping) reply id=0x0084, seq=7/1792, ttl=64 (rep)

Frame 4: 138 bytes on wire (1104 bits), 138 bytes captured (1104 bits) on interface veth7.0.b6, id 0

Ethernet II, Src: 00:00:00\_aa:00:14 (00:00:00:aa:00:14), Dst: 00:00:00\_aa:00:13 (00:00:00:aa:00:13)

Internet Protocol Version 4, Src: 14.200.14.66, Dst: 14.200.8.3

Internet Control Message Protocol

- Type: 0 (Echo (ping) reply)
- Code: 0
- Checksum: 0xd84e [correct]
- [Checksum Status: Good]
- Identifier (BE): 132 (0x0084)
- Identifier (LE): 33792 (0x8400)
- Sequence number (BE): 1 (0x0001)
- Sequence number (LE): 256 (0x0100)

[Request frame: 3]

[Response time: 0,006 ms]

Timestamp from icmp data: Dec 11, 2024 18:49:36.000000000 -03

[Timestamp from icmp data (relative): 0.387540973 seconds]

Data (48 bytes)

Luego de intercambiar paquetes ICMP con n13, n7 envía un nuevo paquete ARP request hacia n2, preguntando por su MAC. Si bien n2 ya “conoce” a n7 y lo tiene en su tabla ARP, n7 no lo tiene a n2. Entonces, n7 intercambia mensajes de ARP con n2 para poder agregarlo a la tabla.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	00:00:00_aa:00:13	Broadcast	ARP	42	Who has 14.200.14.66? Tell 14.200.14.65
2	0.000008370	00:00:00_aa:00:14	00:00:00_aa:00:13	ARP	42	14.200.14.66 is at 00:00:00_aa:00:14
3	0.000018302	14.200.8.3	14.200.14.66	ICMP	138	Echo (ping) request id=0x0084, seq=1/256, ttl=61 (req)
4	0.000024494	14.200.14.66	14.200.8.3	ICMP	138	Echo (ping) reply id=0x0084, seq=1/256, ttl=64 (rep)
5	1.022032451	14.200.8.3	14.200.14.66	ICMP	138	Echo (ping) request id=0x0084, seq=2/512, ttl=61 (req)
6	1.022042531	14.200.14.66	14.200.8.3	ICMP	138	Echo (ping) reply id=0x0084, seq=2/512, ttl=64 (rep)
7	2.046291910	14.200.8.3	14.200.14.66	ICMP	138	Echo (ping) request id=0x0084, seq=3/768, ttl=61 (req)
8	2.046302833	14.200.14.66	14.200.8.3	ICMP	138	Echo (ping) reply id=0x0084, seq=3/768, ttl=64 (rep)
9	3.070001330	14.200.8.3	14.200.14.66	ICMP	138	Echo (ping) request id=0x0084, seq=4/1024, ttl=61 (req)
10	3.070012704	14.200.14.66	14.200.8.3	ICMP	138	Echo (ping) reply id=0x0084, seq=4/1024, ttl=64 (rep)
11	4.094436415	14.200.8.3	14.200.14.66	ICMP	138	Echo (ping) request id=0x0084, seq=5/1280, ttl=61 (req)
12	4.094456034	14.200.14.66	14.200.8.3	ICMP	138	Echo (ping) reply id=0x0084, seq=5/1280, ttl=64 (rep)
13	5.118480289	14.200.8.3	14.200.14.66	ICMP	138	Echo (ping) request id=0x0084, seq=6/1536, ttl=61 (req)
14	5.118493974	14.200.14.66	14.200.8.3	ICMP	138	Echo (ping) reply id=0x0084, seq=6/1536, ttl=64 (rep)
15	5.182060980	00:00:00_aa:00:14	00:00:00_aa:00:13	ARP	42	Who has 14.200.14.65? Tell 14.200.14.66
16	5.182159094	00:00:00_aa:00:13	00:00:00_aa:00:14	ARP	42	14.200.14.65 is at 00:00:00_aa:00:13
17	6.142213173	14.200.8.3	14.200.14.66	ICMP	138	Echo (ping) request id=0x0084, seq=7/1792, ttl=61 (req)
18	6.142226620	14.200.14.66	14.200.8.3	ICMP	138	Echo (ping) reply id=0x0084, seq=7/1792, ttl=64 (rep)

Frame 15: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface veth7.0.b6, id 0

Ethernet II, Src: 00:00:00\_aa:00:14 (00:00:00:aa:00:14), Dst: 00:00:00\_aa:00:13 (00:00:00:aa:00:13)

Address Resolution Protocol (request)

- Hardware type: Ethernet (1)
- Protocol type: IPv4 (0x0800)
- Hardware size: 6
- Protocol size: 4
- Opcode: request (1)
- Sender MAC address: 00:00:00\_aa:00:14 (00:00:00:aa:00:14)
- Sender IP address: 14.200.14.66
- Target MAC address: 00:00:00\_00:00:00 (00:00:00:00:00:00)
- Target IP address: 14.200.14.65

## f) Realizar un traceroute entre los mismos equipos, capturar los mensajes.

A continuación se realiza un análisis de las capturas obtenidas al realizar un traceroute desde n13 a n7 y observar el tráfico en la interfaz eth0 de n7.

La herramienta traceroute envía mensajes UDP por toda la red y recibe respuestas de los routers. Para determinar que se llegó a destino, espera una respuesta de “puerto inalcanzable”. Los paquetes UDP y sus respuestas ICMP se ven repetidos porque traceroute envía varios mensajes para determinar la ruta de los datos. Este comienza enviando mensajes con TTL=1 en donde espera recibir el mensaje TIME\_EXCEEDED por parte del primer router el cual descarta el mensaje y envía un ICMP al origen, luego el origen continúa enviando TTL incrementales. El primer mensaje es un segmento UDP proveniente de n13.

El formato del mensaje UDP es:

- Puerto origen: 55176
- Puerto destino: 33443
- Longitud: 40
- Datos: 32 bytes

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	14.200.8.3	14.200.14.66	UDP	74	55176 - 33443 Len=32
2	0.0000084614	14.200.14.66	14.200.8.3	ICMP	102	Destination unreachable (Port unreachable)
3	0.0000202732	14.200.8.3	14.200.14.66	UDP	74	56026 - 33444 Len=32
4	0.0000211927	14.200.14.66	14.200.8.3	ICMP	102	Destination unreachable (Port unreachable)
5	0.0000275811	14.200.8.3	14.200.14.66	UDP	74	39479 - 33445 Len=32
6	0.0000281470	14.200.14.66	14.200.8.3	ICMP	102	Destination unreachable (Port unreachable)
7	0.0000336721	14.200.8.3	14.200.14.66	UDP	74	42926 - 33446 Len=32
8	0.0000342125	14.200.14.66	14.200.8.3	ICMP	102	Destination unreachable (Port unreachable)
9	0.0000493685	14.200.8.3	14.200.14.66	UDP	74	48767 - 33447 Len=32
10	0.0000503921	14.200.14.66	14.200.8.3	ICMP	102	Destination unreachable (Port unreachable)
11	0.0000644163	14.200.8.3	14.200.14.66	UDP	74	48592 - 33448 Len=32
12	0.0000656125	14.200.14.66	14.200.8.3	ICMP	102	Destination unreachable (Port unreachable)
13	0.0000732576	14.200.8.3	14.200.14.66	UDP	74	57527 - 33449 Len=32
14	5.251300686	00:00:00_aa:00:14	00:00:00_aa:00:13	ARP	42	Who has 14.200.14.65? Tell 14.200.14.66
15	5.251368412	00:00:00_aa:00:13	00:00:00_aa:00:14	ARP	42	Who has 14.200.14.66? Tell 14.200.14.65
16	5.251394227	00:00:00_aa:00:13	00:00:00_aa:00:14	ARP	42	14.200.14.65 is at 00:00:00_aa:00:13
17	5.251386260	00:00:00_aa:00:14	00:00:00_aa:00:13	ARP	42	14.200.14.66 is at 00:00:00_aa:00:14
▶ Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface veth7.0.58, id 0						
▶ Ethernet II, Src: 00:00:00_aa:00:13 (00:00:00:aa:00:13), Dst: 00:00:00_aa:00:14 (00:00:00:aa:00:14)						
▶ Internet Protocol Version 4, Src: 14.200.8.3, Dst: 14.200.14.66						
▶ User Datagram Protocol, Src Port: 55176, Dst Port: 33443						
Source Port: 55176						
▶ Destination Port: 33443						
Length: 40						
Checksum: 0x7c98 [unverified]						
[Checksum Status: Unverified]						
[Stream index: 0]						
▶ [Timestamps]						
▶ Data (32 bytes)						

El formato del mensaje ICMP es:

- Tipo: 3 (destino inalcanzable)
- Código: 3 (puerto inalcanzable)
- Cabecera IPv4 original (20 bytes) + paquete UDP (40 bytes)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	14.200.8.3	14.200.14.66	UDP	74	55176 - 33443 Len=32
2	0.0000084614	14.200.14.66	14.200.8.3	ICMP	102	Destination unreachable (Port unreachable)
3	0.0000202732	14.200.8.3	14.200.14.66	UDP	74	56026 - 33444 Len=32
4	0.0000211927	14.200.14.66	14.200.8.3	ICMP	102	Destination unreachable (Port unreachable)
5	0.0000275811	14.200.8.3	14.200.14.66	UDP	74	39479 - 33445 Len=32
6	0.0000281470	14.200.14.66	14.200.8.3	ICMP	102	Destination unreachable (Port unreachable)
7	0.0000336721	14.200.8.3	14.200.14.66	UDP	74	42926 - 33446 Len=32
8	0.0000342125	14.200.14.66	14.200.8.3	ICMP	102	Destination unreachable (Port unreachable)
9	0.0000493685	14.200.8.3	14.200.14.66	UDP	74	48767 - 33447 Len=32
10	0.0000503921	14.200.14.66	14.200.8.3	ICMP	102	Destination unreachable (Port unreachable)
11	0.0000644163	14.200.8.3	14.200.14.66	UDP	74	48592 - 33448 Len=32
12	0.0000656125	14.200.14.66	14.200.8.3	ICMP	102	Destination unreachable (Port unreachable)
13	0.0000732576	14.200.8.3	14.200.14.66	UDP	74	57527 - 33449 Len=32
14	5.251300686	00:00:00_aa:00:14	00:00:00_aa:00:13	ARP	42	Who has 14.200.14.65? Tell 14.200.14.66
15	5.251368412	00:00:00_aa:00:13	00:00:00_aa:00:14	ARP	42	Who has 14.200.14.66? Tell 14.200.14.65
16	5.251394227	00:00:00_aa:00:13	00:00:00_aa:00:14	ARP	42	14.200.14.65 is at 00:00:00_aa:00:13
17	5.251386260	00:00:00_aa:00:14	00:00:00_aa:00:13	ARP	42	14.200.14.66 is at 00:00:00_aa:00:14
▶ Frame 2: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface veth7.0.58, id 0						
▶ Ethernet II, Src: 00:00:00_aa:00:14 (00:00:00:aa:00:14), Dst: 00:00:00_aa:00:13 (00:00:00:aa:00:13)						
▶ Internet Protocol Version 4, Src: 14.200.14.66, Dst: 14.200.8.3						
▶ Internet Control Message Protocol						
Type: 3 (Destination unreachable)						
Code: 3 (Port unreachable)						
Checksum: 0x310b [correct]						
[Checksum Status: Good]						
Unused: 00000000						
▶ Internet Protocol Version 4, Src: 14.200.8.3, Dst: 14.200.14.66						
▶ User Datagram Protocol, Src Port: 55176, Dst Port: 33443						
Source Port: 55176						
▶ Destination Port: 33443						
Length: 40						
Checksum: 0x7c98 [unverified]						
[Checksum Status: Unverified]						
[Stream index: 0]						
▶ [Timestamps]						
▶ Data (32 bytes)						

En esta última captura se puede observar como el router descarta el mensaje y envía un ICMP “Destination unreachable, Port unreachable” al origen.

También se capturó el tráfico en la interfaz eth0 de n13, donde pueden verse los mensajes UDP que han sido rechazados por la expiración del TTL.

En la imagen se muestra una de las respuestas del router n5 (14.200.8.1) a n13. Este mensaje es de tipo “tiempo excedido” (se rechazó el mensaje UDP) pues el TTL del mensaje UDP era 1 y expiró en el router n5. Se hacen dos intentos más con TTL = 1, ambos rechazados por n5, devolviendo el mismo mensaje ICMP, cuyo formato es:

- Tipo: 11 (tiempo de vida excedido)
- Código: 0 (tiempo de vida excedido en tránsito)
- Cabecera IPv4 original (20 bytes) + paquete UDP (40 bytes)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	14.200.8.3	14.200.14.66	UDP	74	50465 → 33434 Len=32
2	0.000120516	14.200.8.1	14.200.8.3	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
3	0.000182147	14.200.8.3	14.200.14.66	UDP	74	47725 → 33435 Len=32
4	0.000224540	14.200.8.1	14.200.8.3	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
5	0.000351895	14.200.8.3	14.200.14.66	UDP	74	42363 → 33436 Len=32
6	0.000406369	14.200.8.1	14.200.8.3	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)

```

Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface vethd.0.42, id 0
Ethernet II, Src: 00:00:00:aa:00:04 (00:00:00:aa:00:04), Dst: 00:00:00_aa:00:02 (00:00:00:aa:00:02)
Internet Protocol Version 4, Src: 14.200.8.3, Dst: 14.200.14.66
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
        Total Length: 60
        Identification: 0x7f97 (32663)
        Flags: 0x0000
        Fragment offset: 0
    Time to live: 1
    Protocol: UDP (17)
    Header checksum: 0x0646 [validation disabled]
        [Header checksum status: Unverified]
    Source: 14.200.8.3
    Destination: 14.200.14.66
User Datagram Protocol, Src Port: 50465, Dst Port: 33434
Data (32 bytes)

```

Los siguientes tres mensajes UDP con TTL=2 también son rechazados, pero por el router n3 (14.200.10.6). El host n13 ahora “sabe” que la ruta es n5→n3.

No.	Time	Source	Destination	Protocol	Length	Info
7	0.000463750	14.200.8.3	14.200.14.66	UDP	74	38483 → 33437 Len=32
8	0.000537463	14.200.10.6	14.200.8.3	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
9	0.000587998	14.200.8.3	14.200.14.66	UDP	74	35266 → 33438 Len=32
10	0.0006645587	14.200.10.6	14.200.8.3	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
11	0.000694181	14.200.8.3	14.200.14.66	UDP	74	56415 → 33439 Len=32
12	0.000870637	14.200.10.6	14.200.8.3	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)

```

Frame 7: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface vethd.0.42, id 0
Ethernet II, Src: 00:00:00_aa:00:04 (00:00:00_aa:00:04), Dst: 00:00:00_aa:00:02 (00:00:00_aa:00:02)
Internet Protocol Version 4, Src: 14.200.8.3, Dst: 14.200.14.66
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
        Total Length: 60
        Identification: 0xfba3 (64419)
        Flags: 0x0000
        Fragment offset: 0
    Time to live: 2
    Protocol: UDP (17)
    Header checksum: 0x8939 [validation disabled]
        [Header checksum status: Unverified]
    Source: 14.200.8.3
    Destination: 14.200.14.66
User Datagram Protocol, Src Port: 38483, Dst Port: 33437
Data (32 bytes)

```

Se envían tres paquetes UDP más, con TTL = 3. Estos son rechazados por el router n2 (14.200.11.13) a través de un ICMP de “tiempo excedido”.

No.	Time	Source	Destination	Protocol	Length Info
13	0.001304716	14.200.8.3	14.200.14.66	UDP	74 59188 - 33440 Len=32
14	0.001425003	14.200.11.13	14.200.8.3	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
15	0.001492621	14.200.8.3	14.200.14.66	UDP	74 53329 - 33441 Len=32
16	0.001683659	14.200.11.13	14.200.8.3	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
17	0.001845901	14.200.8.3	14.200.14.66	UDP	74 44231 - 33442 Len=32
18	0.001945878	14.200.11.13	14.200.8.3	ICMP	102 Time-to-live exceeded (Time to live exceeded in transit)
Frame 13: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface vethd.0.42, id 0					
Ethernet II, Src: 00:00:00_aa:00:04 (00:00:00:aa:00:04), Dst: 00:00:00_aa:00:02 (00:00:00:aa:00:02)					
Internet Protocol Version 4, Src: 14.200.8.3, Dst: 14.200.14.66					
0100 .... = Version: 4					
.... 0101 = Header Length: 20 bytes (5)					
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)					
Total Length: 60					
Identification: 0xc5ba (50618)					
Flags: 0x0000					
Fragment offset: 0					
Time to live: 3					
Protocol: UDP (17)					
Header checksum: 0xbe22 [validation disabled]					
[Header checksum status: Unverified]					
Source: 14.200.8.3					
Destination: 14.200.14.66					
User Datagram Protocol, Src Port: 59108, Dst Port: 33440					
Data (32 bytes)					

Finalmente el host n13 envía tres paquetes UDP con TTL = 4 y recibe respuesta de n7 (14.200.14.66) en un mensaje ICMP “destino inalcanzable”, cuyo formato es:

- Tipo: 3 (destino inalcanzable)
- Código: 3 (puerto inalcanzable)
- Cabecera IPv4 original (20 bytes) + paquete UDP (40 bytes)

No.	Time	Source	Destination	Protocol	Length Info
19	0.001995858	14.200.8.3	14.200.14.66	UDP	74 48367 - 33443 Len=32
20	0.002166968	14.200.14.66	14.200.8.3	ICMP	102 Destination unreachable (Port unreachable)
21	0.002157498	14.200.8.3	14.200.14.66	UDP	74 57440 - 33444 Len=32
22	0.002423604	14.200.14.66	14.200.8.3	ICMP	102 Destination unreachable (Port unreachable)
23	0.003297198	14.200.8.3	14.200.14.66	UDP	74 35255 - 33445 Len=32
24	0.003433975	14.200.14.66	14.200.8.3	ICMP	102 Destination unreachable (Port unreachable)
Frame 19: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface vethd.0.42, id 0					
Ethernet II, Src: 00:00:00_aa:00:04 (00:00:00:aa:00:04), Dst: 00:00:00_aa:00:02 (00:00:00:aa:00:02)					
Internet Protocol Version 4, Src: 14.200.8.3, Dst: 14.200.14.66					
0100 .... = Version: 4					
.... 0101 = Header Length: 20 bytes (5)					
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)					
Total Length: 60					
Identification: 0xb059 (45145)					
Flags: 0x0000					
Fragment offset: 0					
Time to live: 4					
Protocol: UDP (17)					
Header checksum: 0xd283 [validation disabled]					
[Header checksum status: Unverified]					
Source: 14.200.8.3					
Destination: 14.200.14.66					
User Datagram Protocol, Src Port: 48367, Dst Port: 33443					
Data (32 bytes)					

### g) Alternativo: Modificar los MTU para ver la fragmentación.

Para ver la fragmentación se ejecutó el comando ping -c 1 -s 1500 14.200.12.2 en la terminal del nodo n13. La opción -c indica la cantidad de paquetes que se van a enviar (1) y -s determina el tamaño del payload (1500). Como el MTU es 1500, entonces el mayor payload que se puede transmitir es de 1480 (1500 menos los 20 bytes de la cabecera de IPv4). Dado que se pidió a través del comando un payload de 1500, el mensaje tuvo que ser fragmentado en 2 fragmentos.

#### 1er fragmento del Echo Request:

No.	Time	Source	Destination	Protocol	Length Info
1	0.000000000	14.200.8.3	14.200.12.2	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=85d2) [Reassembled in #2]
2	0.00011557	14.200.8.3	14.200.12.2	ICMP	62 Echo (ping) request id=0x0032, seq=1/256, ttl=64 (reply in 4)
3	0.000181439	14.200.12.2	14.200.8.3	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=ce2b) [Reassembled in #4]
4	0.000182682	14.200.12.2	14.200.8.3	ICMP	62 Echo (ping) reply id=0x0032, seq=1/256, ttl=59 (request in 2)

A partir de la captura del tráfico de la interfaz eth0 de n13, podemos ver que efectivamente hubo fragmentación.

No.	Time	Source	Destination	Protocol	Length Info
1	0.000000000	14.200.8.3	14.200.12.2	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=85d2) [Reassembled in #2]
2	0.000011557	14.200.8.3	14.200.12.2	ICMP	62 Echo (ping) request id=0x0032, seq=1/256, ttl=64 (reply in 4)
3	0.000181439	14.200.12.2	14.200.8.3	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=ce2b) [Reassembled in #4]
4	0.000182682	14.200.12.2	14.200.8.3	ICMP	62 Echo (ping) reply id=0x0032, seq=1/256, ttl=59 (request in 2)
Frame 1: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface vethd.0.b4, id 0					
Ethernet II, Src: 00:00:00_aa:00:04 (00:00:00:aa:00:04), Dst: 00:00:00_aa:00:02 (00:00:00:aa:00:02)					
Destination: 00:00:00_aa:00:02 (00:00:00:aa:00:02)					
Source: 00:00:00_aa:00:04 (00:00:00:aa:00:04)					
Type: IPv4 (0x0800)					
Internet Protocol Version 4, Src: 14.200.8.3, Dst: 14.200.12.2					
0100 .... = Version: 4					
.... 0101 = Header Length: 20 bytes (5)					
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)					
Total Length: 1500					
Identification: 0x85d2 (34258)					
Flags: 0x2000, More fragments					
Fragment offset: 0					
Time to live: 64					
Protocol: ICMP (1)					
Header checksum: 0x9dba [validation disabled]					
[Header checksum status: Unverified]					
Source: 14.200.8.3					
Destination: 14.200.12.2					
<b>Reassembled IPv4 in frame: 2</b>					
Data (1480 bytes)					

Se puede ver que el primer segmento (fragment offset = 0) tiene 20 bytes que conforman la cabecera IPv4, y completa los 1500 bytes con los 1480 del payload. Para indicar que hay más fragmentos, se activó el bit More Fragment.

## 2do fragmento del Echo Request:

No.	Time	Source	Destination	Protocol	Length Info
1	0.000000000	14.200.8.3	14.200.12.2	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=85d2) [Reassembled in #2]
2	0.000011557	14.200.8.3	14.200.12.2	ICMP	62 Echo (ping) request id=0x0032, seq=1/256, ttl=64 (reply in 4)
3	0.000181439	14.200.12.2	14.200.8.3	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=ce2b) [Reassembled in #4]
4	0.000182682	14.200.12.2	14.200.8.3	ICMP	62 Echo (ping) reply id=0x0032, seq=1/256, ttl=59 (request in 2)
Frame 2: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface vethd.0.b4, id 0					
Ethernet II, Src: 00:00:00_aa:00:04 (00:00:00:aa:00:04), Dst: 00:00:00_aa:00:02 (00:00:00:aa:00:02)					
Destination: 00:00:00_aa:00:02 (00:00:00:aa:00:02)					
Source: 00:00:00_aa:00:04 (00:00:00:aa:00:04)					
Type: IPv4 (0x0800)					
Internet Protocol Version 4, Src: 14.200.8.3, Dst: 14.200.12.2					
0100 .... = Version: 4					
.... 0101 = Header Length: 20 bytes (5)					
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)					
Total Length: 48					
Identification: 0x00b9 (34258)					
Flags: 0x00b9					
0... .... .... = Reserved bit: Not set					
. .... .... .... = Don't fragment: Not set					
..0 .... .... .... = More fragments: Not set					
Fragment offset: 1480					
Time to live: 64					
Protocol: ICMP (1)					
Header checksum: 0xc2ad [validation disabled]					
[Header checksum status: Unverified]					
Source: 14.200.8.3					
Destination: 14.200.12.2					
<b>[2 IPv4 Fragments (1508 bytes): #1(1480), #2(28)]</b>					
<b>[Frame: 1, payload: 0-1479 (1480 bytes)]</b>					
<b>[Frame: 2, payload: 1480-1507 (28 bytes)]</b>					
<b>[Fragment count: 2]</b>					
<b>[Reassembled IPv4 length: 1508]</b>					
<b>[Reassembled IPv4 data: 080023d80032000132905b670000000ac8b0b0000000000...]</b>					
<b>Internet Control Message Protocol</b>					
Type: 8 (Echo (ping) request)					
Code: 0					
Checksum: 0x23d8 [correct]					
[Checksum Status: Good]					
Identifier (BE): 50 (0x0032)					
Identifier (LE): 12800 (0x3200)					
Sequence number (BE): 1 (0x0001)					
Sequence number (LE): 256 (0x0100)					
[Response frame: 4]					
Timestamp from icmp data: Dec 12, 2024 22:38:58.000000000 -03					
[Timestamp from icmp data (relative): 0.756679272 seconds]					
Data (1492 bytes)					

En este caso, vemos que el segment offset es 1480, por lo que es el siguiente segmento (y el último, porque el flag More Fragment está desactivado). Se puede observar que la longitud total del paquete IPv4 es de 48 bytes, conformados por 20 bytes de cabecera IP, 20 bytes de payload (los que faltaban de los 1500), y 8 bytes de la cabecera de ICMP.

## **h) Probar conectividad en las redes con IPv6 (por separado), capturar tráfico y analizar ICMPv6.**

A continuación se comprueba la conectividad de las dos redes con IPv6 mediante el uso de los comandos ping y traceroute.

### RED A.1

Esta red está conformada por los nodos n5, n11 y n13. Primero se comenzó enviando realizando tres pings: de n13 a n5, de n13 a n11 y de n11 a n5. A continuación se detalla la información detectada desde Wireshark.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000...	200a:b8d:1014:... ff02::1:ff00:2		ICMP...	86	Neighbor Solicitation for 200a:b8d:1014:1::2 ...
2	0.000070...	200a:b8d:1014:... 200a:b8d:1014:...	200a:b8d:1014:...	ICMP...	86	Neighbor Advertisement 200a:b8d:1014:1::2 (so...
3	0.000073...	200a:b8d:1014:... 200a:b8d:1014:...	200a:b8d:1014:...	ICMP...	118	Echo (ping) request id=0x0089, seq=1, hop lim...
4	0.000104...	200a:b8d:1014:... 200a:b8d:1014:...	200a:b8d:1014:...	ICMP...	118	Echo (ping) reply id=0x0089, seq=1, hop limit...
5	1.030420...	200a:b8d:1014:... 200a:b8d:1014:...	200a:b8d:1014:...	ICMP...	118	Echo (ping) request id=0x0089, seq=2, hop lim...
6	1.030446...	200a:b8d:1014:... 200a:b8d:1014:...	200a:b8d:1014:...	ICMP...	118	Echo (ping) reply id=0x0089, seq=2, hop limit...
7	2.054465...	200a:b8d:1014:... 200a:b8d:1014:...	200a:b8d:1014:...	ICMP...	118	Echo (ping) request id=0x0089, seq=3, hop lim...
8	2.054492...	200a:b8d:1014:... 200a:b8d:1014:...	200a:b8d:1014:...	ICMP...	118	Echo (ping) reply id=0x0089, seq=3, hop limit...
9	3.078161...	200a:b8d:1014:... 200a:b8d:1014:...	200a:b8d:1014:...	ICMP...	118	Echo (ping) request id=0x0089, seq=4, hop lim...
10	3.078187...	200a:b8d:1014:... 200a:b8d:1014:...	200a:b8d:1014:...	ICMP...	118	Echo (ping) reply id=0x0089, seq=4, hop limit...
11	4.102164...	200a:b8d:1014:... 200a:b8d:1014:...	200a:b8d:1014:...	ICMP...	118	Echo (ping) request id=0x0089, seq=5, hop lim...
12	4.102191...	200a:b8d:1014:... 200a:b8d:1014:...	200a:b8d:1014:...	ICMP...	118	Echo (ping) reply id=0x0089, seq=5, hop limit...

Next Header: ICMPv6 (58)  
Hop Limit: 255  
Source: 200a:b8d:1014:1::3  
Destination: ff02::1:ff00:2

Internet Control Message Protocol v6

Type: Neighbor Solicitation (135)  
Code: 0  
Checksum: 0x0192 [correct]  
[Checksum Status: Good]  
Reserved: 00000000  
Target Address: 200a:b8d:1014:1::2

ICMPv6 Option (Source link-layer address : 00:00:00:aa:00:04)

0000	33 33 ff 00 00 02 00 00	00 aa 00 04 86 dd 60 00	33 . . . . . . . .
0010	00 00 00 20 3a ff 20 0a	0b 8d 10 14 00 01 00 00	. . . : . . . . . . .
0020	00 00 00 00 00 03 ff 02	00 00 00 00 00 00 00 00	. . . . . . . . . . .
0030	00 01 ff 00 00 02 87 00	01 92 00 00 00 00 20 0a	. . . . . . . . . . .
0040	0b 8d 10 14 00 01 00 00	00 00 00 00 00 00 02 01 01	. . . . . . . . . . .
0050	00 00 00 aa 00 04		. . . . . . . . . . .

En esta captura se puede observar que el primer mensaje es de “Neighbor Solicitation” y el segundo de “Neighbor Advertisement”, a partir de estos comenzaron los mensajes de tipo echo (ping)

No.	Time	Source	Destination	Protocol	Length/Traffic
1	0.000000...	200a:b8d:1014...	ff02::1:ff00:2	ICMP...	86 Neighbor Solicitation for 200a:b8d:1014:1::2 ...
2	0.000070...	200a:b8d:1014...	200a:b8d:1014:...	ICMP...	86 Neighbor Advertisement 200a:b8d:1014:1::2 (so...
3	0.000073...	200a:b8d:1014:...	200a:b8d:1014:...	ICMP...	118 Echo (ping) request id=0x0089, seq=1, hop lim...
4	0.000104...	200a:b8d:1014:...	200a:b8d:1014:...	ICMP...	118 Echo (ping) reply id=0x0089, seq=1, hop limit...
5	1.030420...	200a:b8d:1014:...	200a:b8d:1014:...	ICMP...	118 Echo (ping) request id=0x0089, seq=2, hop lim...
6	1.030446...	200a:b8d:1014:...	200a:b8d:1014:...	ICMP...	118 Echo (ping) reply id=0x0089, seq=2, hop limit...
7	2.054465...	200a:b8d:1014:...	200a:b8d:1014:...	ICMP...	118 Echo (ping) request id=0x0089, seq=3, hop lim...
8	2.054492...	200a:b8d:1014:...	200a:b8d:1014:...	ICMP...	118 Echo (ping) reply id=0x0089, seq=3, hop limit...
9	3.078161...	200a:b8d:1014:...	200a:b8d:1014:...	ICMP...	118 Echo (ping) request id=0x0089, seq=4, hop lim...
10	3.078187...	200a:b8d:1014:...	200a:b8d:1014:...	ICMP...	118 Echo (ping) reply id=0x0089, seq=4, hop limit...
11	4.102164...	200a:b8d:1014:...	200a:b8d:1014:...	ICMP...	118 Echo (ping) request id=0x0089, seq=5, hop lim...
12	4.102191...	200a:b8d:1014:...	200a:b8d:1014:...	ICMP...	118 Echo (ping) reply id=0x0089, seq=5, hop limit...
Destination: 200a:b8d:1014:1::2					
Internet Control Message Protocol v6					
Type: Echo (ping) request (128)					
Code: 0					
Checksum: 0x1466 [correct]					
[Checksum Status: Good]					
Identifier: 0x0089					
Sequence: 1					
<u>[Response In: 4]</u>					
Data (56 bytes)					
Data: 63a05767000000000735d0600000000001011121314151617...					
[Length: 56]					
0000	00 00 00 aa 00 03 00 00 00 aa 00 04 86 dd 60 0c				..... . . . . .
0010	f0 a8 00 40 3a 40 20 0a 0b 8d 10 14 00 01 00 00				.. @:@ .. . . . .
0020	00 00 00 00 00 03 20 0a 0b 8d 10 14 00 01 00 00				..... . . . . .
0030	00 00 00 00 00 02 80 00 14 66 00 89 00 01 63 a0				..... f . . c .
0040	57 67 00 00 00 00 73 5d 06 00 00 00 00 00 10 11				Wg . . s ] .. . . . !
0050	12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21				..... . . . . .
0060	22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31				"#\$%&'() *+, -./01
0070	32 33 34 35 36 37				234567

Aquí se puede observar que el paquete resaltado en la captura corresponde al comando ping (“echo request” en ICMPv6) y lleva los siguientes campos:

- Type (1 byte): echo request (128)
  - Código (1 byte): 0
  - Checksum (2 bytes)
  - ID (2 bytes): 0x0089
  - Data (56 bytes)

```

1 0.000000... 200a:b8d:1014:... ff02::1:ff00:2 ICMP... 86 Neighbor Solicitation for 200a:b8d:1014:1::2 ...
2 0.000070... 200a:b8d:1014:... 200a:b8d:1014:... ICMP... 86 Neighbor Advertisement 200a:b8d:1014:1::2 (so...
3 0.000073... 200a:b8d:1014:... 200a:b8d:1014:... ICMP... 118 Echo (ping) request id=0x0089, seq=1, hop lim...
4 0.000104... 200a:b8d:1014:... 200a:b8d:1014:... ICMP... 118 Echo (ping) reply id=0x0089, seq=1, hop limit...
5 1.030420... 200a:b8d:1014:... 200a:b8d:1014:... ICMP... 118 Echo (ping) request id=0x0089, seq=2, hop lim...
6 1.030446... 200a:b8d:1014:... 200a:b8d:1014:... ICMP... 118 Echo (ping) reply id=0x0089, seq=2, hop limit...
7 2.054465... 200a:b8d:1014:... 200a:b8d:1014:... ICMP... 118 Echo (ping) request id=0x0089, seq=3, hop lim...
8 2.054492... 200a:b8d:1014:... 200a:b8d:1014:... ICMP... 118 Echo (ping) reply id=0x0089, seq=3, hop limit...
9 3.078161... 200a:b8d:1014:... 200a:b8d:1014:... ICMP... 118 Echo (ping) request id=0x0089, seq=4, hop lim...
10 3.078187... 200a:b8d:1014:... 200a:b8d:1014:... ICMP... 118 Echo (ping) reply id=0x0089, seq=4, hop limit...
11 4.102164... 200a:b8d:1014:... 200a:b8d:1014:... ICMP... 118 Echo (ping) request id=0x0089, seq=5, hop lim...
12 4.102191... 200a:b8d:1014:... 200a:b8d:1014:... ICMP... 118 Echo (ping) reply id=0x0089, seq=5, hop limit...
.....0 ..... . .... . .... = IG bit: Individual address (unicast)
Type: IPv6 (0x86dd)
Internet Protocol Version 6, Src: 200a:b8d:1014:1::3, Dst: 200a:b8d:1014:1::2
0110 .... = Version: 6
> .... 0000 0000 .... .... .... .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
.... .... 1100 1111 0000 1010 1000 = Flow Label: 0xcf0a8
Payload Length: 64
Next Header: ICMPv6 (58)
Hop Limit: 64
Source: 200a:b8d:1014:1::3
Destination: 200a:b8d:1014:1::2
Internet Control Message Protocol v6
Type: Echo (ping) request (128)
0000 00 00 aa 00 03 00 00 00 aa 00 04 86 dd 60 0c
0010 f0 a8 00 40 3a 40 20 0a 0b 8d 10 14 00 01 00 00
0020 00 00 00 00 00 03 20 0a 0b 8d 10 14 00 01 00 00
0030 00 00 00 00 00 02 80 00 14 66 00 89 00 01 63 a0
0040 57 67 00 00 00 00 73 5d 06 00 00 00 00 00 10 11
0050 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21
0060 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31
0070 32 33 34 35 36 37
.....@:@ .....f....c.
.....Wg....s] .....!
"#$%&'() *+,--./01
234567

```

El paquete es encapsulado en un datagrama IPv6 enviado desde n13 a n11. Sus campos tienen los siguientes valores:

- Version (4 bits): 6
- Traffic Class (1 byte): 0
- Flow Label (20 bytes): 0xcf0a8
- Payload Length (2 bytes): 64
- Next Header (1 byte): ICMPV6 (58)
- Hop Limit (1 byte): 64
- Source (128 bits): 200a:b8d:1014:1::3 (n13)
- Destination (128 bits): 200a:b8d:1014:1::2 (n11)

A continuación, se muestra la captura de la respuesta de n11 a n13:

```

1 0.000000... 200a:b8d:1014:1::3 ff02::1:ff00:2 ICMP... 86 Neighbor Solicitation for 200a:b8d:1014:1::2 ...
2 0.000070... 200a:b8d:1014:1::2 200a:b8d:1014:1::3 ICMP... 86 Neighbor Advertisement 200a:b8d:1014:1::2 (so...
3 0.000073... 200a:b8d:1014:1::3 200a:b8d:1014:1::2 ICMP... 118 Echo (ping) request id=0x0089, seq=1, hop lim...
4 0.000104... 200a:b8d:1014:1::2 200a:b8d:1014:1::3 ICMP... 118 Echo (ping) reply id=0x0089, seq=1, hop limit...
5 1.030420... 200a:b8d:1014:1::3 200a:b8d:1014:1::2 ICMP... 118 Echo (ping) request id=0x0089, seq=2, hop lim...
6 1.030446... 200a:b8d:1014:1::2 200a:b8d:1014:1::3 ICMP... 118 Echo (ping) reply id=0x0089, seq=2, hop limit...
7 2.054465... 200a:b8d:1014:1::3 200a:b8d:1014:1::2 ICMP... 118 Echo (ping) request id=0x0089, seq=3, hop lim...
8 2.054492... 200a:b8d:1014:1::2 200a:b8d:1014:1::3 ICMP... 118 Echo (ping) reply id=0x0089, seq=3, hop limit...
9 3.078161... 200a:b8d:1014:1::3 200a:b8d:1014:1::2 ICMP... 118 Echo (ping) request id=0x0089, seq=4, hop lim...
10 3.078187... 200a:b8d:1014:1::2 200a:b8d:1014:1::3 ICMP... 118 Echo (ping) reply id=0x0089, seq=4, hop limit...
11 4.102164... 200a:b8d:1014:1::3 200a:b8d:1014:1::2 ICMP... 118 Echo (ping) request id=0x0089, seq=5, hop lim...
12 4.102191... 200a:b8d:1014:1::2 200a:b8d:1014:1::3 ICMP... 118 Echo (ping) reply id=0x0089, seq=5, hop limit...
Hop Limit: 64
Source: 200a:b8d:1014:1::2
Destination: 200a:b8d:1014:1::3
- Internet Control Message Protocol v6
  Type: Echo (ping) reply (129)
  Code: 0
  Checksum: 0x1366 [correct]
  [Checksum Status: Good]
  Identifier: 0x0089
  Sequence: 1
  [Response To: 3]
  [Response Time: 0,031 ms]
  ▾ Data (56 bytes)
    0000 00 00 00 aa 00 04 00 00 00 aa 00 03 86 dd 60 00 ..... .
    0010 5a 11 00 40 3a 40 20 0a 0b 8d 10 14 00 01 00 00 Z:@@..... .
    0020 00 00 00 00 00 02 20 0a 0b 8d 10 14 00 01 00 00 ..... .
    0030 00 00 00 00 00 03 81 00 13 66 00 89 00 01 63 a0 ..... f...c.
    0040 57 67 00 00 00 00 73 5d 06 00 00 00 00 00 10 11 Wg...s]..... !
    0050 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 ..... !.
    0060 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 "#$%&(')*+,-./01
    0070 32 33 34 35 36 37 234567

```

Aquí se puede observar que el paquete resaltado en la captura corresponde al comando ping (“echo request” en ICMPv6) y lleva los siguientes campos:

- Type (1 byte): echo reply (129)
- Código (1 byte): 0
- Checksum (2 bytes)
- ID (2 bytes): 0x0089
- Data (56 bytes)

```

1 0.000000... 200a:b8d:1014:1::3 ff02::1:ff00:2 ICMP... 86 Neighbor Solicitation for 200a:b8d:1014:1::2 ...
2 0.000070... 200a:b8d:1014:1::2 200a:b8d:1014:1::3 ICMP... 86 Neighbor Advertisement 200a:b8d:1014:1::2 (so...
3 0.000073... 200a:b8d:1014:1::3 200a:b8d:1014:1::2 ICMP... 118 Echo (ping) request id=0x0089, seq=1, hop lim...
4 0.000104... 200a:b8d:1014:1::2 200a:b8d:1014:1::3 ICMP... 118 Echo (ping) reply id=0x0089, seq=1, hop limit...
5 1.030420... 200a:b8d:1014:1::3 200a:b8d:1014:1::2 ICMP... 118 Echo (ping) request id=0x0089, seq=2, hop lim...
6 1.030446... 200a:b8d:1014:1::2 200a:b8d:1014:1::3 ICMP... 118 Echo (ping) reply id=0x0089, seq=2, hop limit...
7 2.054465... 200a:b8d:1014:1::3 200a:b8d:1014:1::2 ICMP... 118 Echo (ping) request id=0x0089, seq=3, hop lim...
8 2.054492... 200a:b8d:1014:1::2 200a:b8d:1014:1::3 ICMP... 118 Echo (ping) reply id=0x0089, seq=3, hop limit...
9 3.078161... 200a:b8d:1014:1::3 200a:b8d:1014:1::2 ICMP... 118 Echo (ping) request id=0x0089, seq=4, hop lim...
10 3.078187... 200a:b8d:1014:1::2 200a:b8d:1014:1::3 ICMP... 118 Echo (ping) reply id=0x0089, seq=4, hop limit...
11 4.102164... 200a:b8d:1014:1::3 200a:b8d:1014:1::2 ICMP... 118 Echo (ping) request id=0x0089, seq=5, hop lim...
12 4.102191... 200a:b8d:1014:1::2 200a:b8d:1014:1::3 ICMP... 118 Echo (ping) reply id=0x0089, seq=5, hop limit...
..... 0. .... = LG bit: Globally unique address (factory default)
..... 0. .... = IG bit: Individual address (unicast)
Type: IPv6 (0x86dd)
- Internet Protocol Version 6, Src: 200a:b8d:1014:1::2, Dst: 200a:b8d:1014:1::3
  0110 .... = Version: 6
  > ..... 0000 0000 .... .... .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
  ..... 0000 0101 1010 0001 0001 = Flow Label: 0x05a11
  Payload Length: 64
  Next Header: ICMPv6 (58)
  Hop Limit: 64
  Source: 200a:b8d:1014:1::2
  Destination: 200a:b8d:1014:1::3
- Internet Control Message Protocol v6
  0000 00 00 00 aa 00 04 00 00 00 aa 00 03 86 dd 60 00 ..... .
  0010 5a 11 00 40 3a 40 20 0a 0b 8d 10 14 00 01 00 00 Z:@@..... .
  0020 00 00 00 00 00 02 20 0a 0b 8d 10 14 00 01 00 00 ..... .
  0030 00 00 00 00 00 03 81 00 13 66 00 89 00 01 63 a0 ..... f...c.
  0040 57 67 00 00 00 00 73 5d 06 00 00 00 00 00 10 11 Wg...s]..... !
  0050 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 ..... !.
  0060 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 "#$%&(')*+,-./01
  0070 32 33 34 35 36 37 234567

```

El mensaje ICMPv6 se incluye luego de la cabecera IPv6 mediante la opción de cabecera ICMPv6. Los campos de IPv6 son:

- Versión (4 bits): 6
- Clase de tráfico (1 byte): 0
- Etiqueta de flujo (20 bits): 0x05a11
- Longitud de payload (2 bytes): 64 bytes
- Cabecera siguiente (1 byte): ICMPv6 (58)
- Hop limit (1 byte): 64
- Source (128 bits): 200a:b8d:1014:1::2 (n11)
- Destination (128 bits): 200a:b8d:1014:1::3 (n13)

Los otros mensajes que se pueden visualizar en las capturas son semejantes a los ya explicados anteriormente.

A continuación, se muestran las capturas de pantalla luego de ejecutar el comando traceroute desde n13, que utiliza segmentos UDP junto con datagramas ICMPv6, el cual se encarga de ir trazando un camino a través de la red hasta llegar al destino (n11).

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94	55387 → 33434 Len=32
2	0.0000349...	200a:b8d:1014:1...	200a:b8d:1014:1...	ICMP...	142	Destination Unreachable (Port unreachable)
3	0.0000474...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94	58137 → 33435 Len=32
4	0.0000516...	200a:b8d:1014:1...	200a:b8d:1014:1...	ICMP...	142	Destination Unreachable (Port unreachable)
5	0.0000589...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94	37444 → 33436 Len=32
6	0.0000627...	200a:b8d:1014:1...	200a:b8d:1014:1...	ICMP...	142	Destination Unreachable (Port unreachable)
7	0.0000692...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94	60510 → 33437 Len=32
8	0.0000729...	200a:b8d:1014:1...	200a:b8d:1014:1...	ICMP...	142	Destination Unreachable (Port unreachable)
9	0.0000791...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94	59553 → 33438 Len=32
10	0.0000828...	200a:b8d:1014:1...	200a:b8d:1014:1...	ICMP...	142	Destination Unreachable (Port unreachable)
11	0.0000890...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94	37904 → 33439 Len=32
12	0.0000935...	200a:b8d:1014:1...	200a:b8d:1014:1...	ICMP...	142	Destination Unreachable (Port unreachable)
13	0.0000999...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94	37106 → 33440 Len=32
14	0.0001091...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94	46479 → 33441 Len=32
15	0.0001160...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94	37204 → 33442 Len=32
16	0.0001239...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94	33214 → 33443 Len=32
17	0.0001311...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94	52842 → 33444 Len=32
18	0.0001384...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94	50871 → 33445 Len=32
19	0.0001467...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94	49313 → 33446 Len=32
20	0.0001544...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94	35924 → 33447 Len=32
21	0.0001619...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94	37613 → 33448 Len=32
22	0.0001703...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94	36417 → 33449 Len=32
► Frame 1: 94 bytes on wire (752 bits), 94 bytes captured (752 bits) on interface vethd.0.2b, id 0						
► Ethernet II, Src: 00:00:00_aa:00:04 (00:00:00:aa:00:04), Dst: 00:00:00_aa:00:03 (00:00:00:aa:00:03)						
► Internet Protocol Version 6, Src: 200a:b8d:1014:1::3, Dst: 200a:b8d:1014:1::2						
► User Datagram Protocol, Src Port: 55387, Dst Port: 33434						
Data (60 bytes):						
0000	00 00 00 aa 00 03 00 00 00 aa 00 04 <b>86 dd</b> 60 0c	.....	.....	.....	.....	.....
0010	4e 21 00 28 11 01 20 0a 0b 8d 10 14 00 01 00 00	N!	..(	.....	.....	.....
0020	00 00 00 00 00 03 20 0a 0b 8d 10 14 00 01 00 00	.....	.....	.....	.....	.....
0030	00 00 00 00 00 02 d8 5b 82 9a 00 28 38 46 40 41	.....	[	.....	(8F@A	.....
0040	42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f 50 51	BCDEFGHI	JKLMNOPQ	.....	.....	.....
0050	52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f	RSTUVWXY	Z[\]^_	.....	.....	.....

El segmento UDP es un mensaje genérico que es enviado para que el receptor envíe un mensaje ICMP, que es usado para ir armando la ruta. Traceroute deliberadamente empieza a enviar los paquetes UDP con un valor bajo en el campo "hop limit" y lo va incrementando en 1 cada tres mensajes. Como no debe atravesarse ningún router para que el paquete llegue de n13 a n11, el UDP llega bien en el primer intento.

Cuando el mensaje llega a su destino, n11 le responde a n13 con un mensaje ICMPv6 de “destino inalcanzable”. De no ser así, n13 debería recibir una respuesta de “tiempo excedido” pues irían expirando los paquetes de bajo “hop limit”.

1 0.00000000...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94 55387 → 33434 Len=32
2 0.0000349...	200a:b8d:1014:1...	200a:b8d:1014:1...	ICMP...	142 Destination Unreachable (Port unreachable)
3 0.0000474...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94 58137 → 33435 Len=32
4 0.0000516...	200a:b8d:1014:1...	200a:b8d:1014:1...	ICMP...	142 Destination Unreachable (Port unreachable)
5 0.0000589...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94 37444 → 33436 Len=32
6 0.0000627...	200a:b8d:1014:1...	200a:b8d:1014:1...	ICMP...	142 Destination Unreachable (Port unreachable)
7 0.0000692...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94 60510 → 33437 Len=32
8 0.0000729...	200a:b8d:1014:1...	200a:b8d:1014:1...	ICMP...	142 Destination Unreachable (Port unreachable)
9 0.0000791...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94 59553 → 33438 Len=32
10 0.0000828...	200a:b8d:1014:1...	200a:b8d:1014:1...	ICMP...	142 Destination Unreachable (Port unreachable)
11 0.0000890...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94 37904 → 33439 Len=32
12 0.0000935...	200a:b8d:1014:1...	200a:b8d:1014:1...	ICMP...	142 Destination Unreachable (Port unreachable)
13 0.0000999...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94 37106 → 33440 Len=32
14 0.0001691...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94 46479 → 33441 Len=32
15 0.0001160...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94 37204 → 33442 Len=32
16 0.0001239...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94 33214 → 33443 Len=32
17 0.0001311...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94 52842 → 33444 Len=32
18 0.0001384...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94 50871 → 33445 Len=32
19 0.0001467...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94 49313 → 33446 Len=32
20 0.0001544...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94 35924 → 33447 Len=32
21 0.0001619...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94 37613 → 33448 Len=32
22 0.0001703...	200a:b8d:1014:1...	200a:b8d:1014:1...	UDP	94 36417 → 33449 Len=32
Frame 2: 142 bytes on wire (1136 bits), 142 bytes captured (1136 bits) on interface vethd.0.2b, id 0				
Ethernet II, Src: 00:00:00_aa:00:03 (00:00:00:aa:00:03), Dst: 00:00:00_aa:00:04 (00:00:00:aa:00:04)				
Internet Protocol Version 6, Src: 200a:b8d:1014:1::2, Dst: 200a:b8d:1014:1::3				
Internet Control Message Protocol v6				
0000	00 00 00 aa 00 04 00 00 00 aa 00 03 86 dd 60 04	.....	.....	.....
0010	5a 47 00 58 3a 40 20 0a 0b 8d 10 14 00 01 00 00	ZG-X:@	.....	.....
0020	00 00 00 00 00 00 20 0a 0b 8d 10 14 00 01 00 00	.....	.....	.....
0030	00 00 00 00 00 03 01 04 c7 ee 00 00 00 00 60 0c	.....	.....	.....
0040	4e 21 00 28 11 01 20 0a 0b 8d 10 14 00 01 00 00	N!(...	.....	.....
0050	00 00 00 00 00 03 20 0a 0b 8d 10 14 00 01 00 00	.....	.....	.....
0060	00 00 00 00 00 02 d8 5b 82 9a 00 28 38 46 40 41	.....[ ... (8F@A	.....	.....
0070	42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f 50 51	BCDEFGHI JKLMNOPQ	RSTUVWXY Z[\]^_`	.....
0080	52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f	.....	.....	.....

Para este mensaje ICMPv6, los campos llevan estos valores:

- Tipo (1 byte): 1 (destino inalcanzable)
- Código (1 byte): 4 (puerto inalcanzable)
- Checksum (2 bytes)
- 4 bytes nulos
- Cabecera IPv6 y datos del mensaje original (80 bytes)

Al igual que con el ping, se tiene múltiples mensajes ICMP y múltiples segmentos UDP capturados, pero son muy similares, cambiando el puerto de entrada y el número de secuencia de UDP. Traceroute asigna un puerto a cada segmento UDP desde 33434 a 33449.

No.	Time	Source	Destination	Protocol	Length Info
1	0.0000000...	200a:b8d:1014:1...	200a:b8d:1014:1...	ICMP...	118 Echo (ping) request id=0x0024, seq=1, hop limit=...
2	0.0000628...	200a:b8d:1014:1...	200a:b8d:1014:1...	ICMP...	118 Echo (ping) reply id=0x0024, seq=1, hop limit=64...
3	1.0244445...	200a:b8d:1014:1...	200a:b8d:1014:1...	ICMP...	118 Echo (ping) request id=0x0024, seq=2, hop limit=...
4	1.0244661...	200a:b8d:1014:1...	200a:b8d:1014:1...	ICMP...	118 Echo (ping) reply id=0x0024, seq=2, hop limit=64...
5	1.1208495...	fe80::200:ff:fe... ff02::2		ICMP...	70 Router Solicitation from 00:00:00:aa:00:03
6	2.0484814...	200a:b8d:1014:1...	200a:b8d:1014:1...	ICMP...	118 Echo (ping) request id=0x0024, seq=3, hop limit=...
7	2.0485762...	200a:b8d:1014:1...	200a:b8d:1014:1...	ICMP...	118 Echo (ping) reply id=0x0024, seq=3, hop limit=64...
8	3.0725269...	200a:b8d:1014:1...	200a:b8d:1014:1...	ICMP...	118 Echo (ping) request id=0x0024, seq=4, hop limit=...
9	3.0725515...	200a:b8d:1014:1...	200a:b8d:1014:1...	ICMP...	118 Echo (ping) reply id=0x0024, seq=4, hop limit=64...
10	4.0981703...	200a:b8d:1014:1...	200a:b8d:1014:1...	ICMP...	118 Echo (ping) request id=0x0024, seq=5, hop limit=...
11	4.0982097...	200a:b8d:1014:1...	200a:b8d:1014:1...	ICMP...	118 Echo (ping) reply id=0x0024, seq=5, hop limit=64...
► Frame 1: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface vethd.0.9c, id 0					
► Ethernet II, Src: 00:00:00_aa:00:04 (00:00:00:aa:00:04), Dst: 00:00:00_aa:00:02 (00:00:00:aa:00:02)					
► Internet Protocol Version 6, Src: 200a:b8d:1014:1::3, Dst: 200a:b8d:1014:1::1					
0110 .... = Version: 6					
.... 0000 0000 .... .... .... .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)					
.... .... .... 1010 0111 0101 0000 0010 = Flow Label: 0xa7502					
Payload Length: 64					
Next Header: ICMPv6 (58)					
Hop Limit: 64					
Source: 200a:b8d:1014:1::3					
Destination: 200a:b8d:1014:1::1					
► Internet Control Message Protocol v6					
0000	00 00 00 aa 00 02 00 00 00 aa 00 04 86 dd 60 0a				.....
0010	75 02 00 40 3a 40 20 0a 0b 8d 10 14 00 01 00 00				u...@:0 ..
0020	00 00 00 00 00 03 20 0a 0b 8d 10 14 00 01 00 00				.....
0030	00 00 00 00 00 01 80 00 f5 b3 00 24 00 01 d8 d6				..... \$ .....
0040	58 67 00 00 00 00 1a 3f 08 00 00 00 00 00 10 11				Xg ..... ? .....
0050	12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21				..... ! .....
0060	22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31				"#\$%&'() *+,.-./01
0070	32 33 34 35 36 37				234567

Para probar la conexión entre n13 y el router n5, se utilizó el comando ping. En la interfaz eth0 de n5 se captaron mensajes ICMPv6, donde se pueden observar entrantes (echo request) y salientes (echo reply). Los paquetes de echo request tienen el siguiente formato:

- Tipo (1 byte): 128 (echo request)
- Código (1 byte): 0
- Checksum (2 bytes)
- ID (2 bytes)
- No Secuencia (2 bytes): 1
- Datos (56 bytes)

Los paquetes de echo reply tienen el siguiente formato:

- Tipo (1 byte): 129 (echo reply)
- Código (1 byte): 0
- Checksum (2 bytes)
- ID (2 bytes)
- No Secuencia (2 bytes): 1
- Datos (56 bytes)

Lo único que cambia entre los mensajes de ping es el número de secuencia, que se incrementa en uno en cada echo request.

Estos mensajes van en una cabecera opcional detrás de la cabecera IPv6, cuyos valores son:

- Versión (4 bits): 6
- Clase de tráfico (1 byte): 0
- Etiqueta de flujo (20 bits): 0xa7502
- Longitud de payload (2 bytes): 64 bytes
- Cabecera siguiente (1 byte): ICMPv6 (58)
- Hop limit (1 byte): 64
- Source (128 bits): 200a:b8d:1014:1::3 (n13)
- Destination (128 bits): 200a:b8d:1014:1::1 (n5)

Para los mensajes de echo reply, sólo cambian los campos de direcciones origen y destino del mensaje:

- Source (128 bits): 200a:b8d:1014:1::1 (n5)
- Destination (128 bits): 200a:b8d:1014:1::3 (n13)

También se ha ejecutado traceroute para probar la conexión desde n13 a n5 y se ha capturado el tráfico en la interfaz eth0 de n5. Como en el caso de la conexión de n13 a n11, se intercambian segmentos UDP y mensajes ICMPv6 de “destino inalcanzable” porque el camino es directo (sin saltos intermedios).

```

8 9.0215110... 200a:b8d:1014:1... 200a:b8d:1014:1... UDP      86 40217 → 33434 Len=24
9 9.0215250... 200a:b8d:1014:1... 200a:b8d:1014:1... ICMP... 134 Destination Unreachable (Port unreachable)
10 9.0215408... 200a:b8d:1014:1... 200a:b8d:1014:1... UDP     86 40217 → 33434 Len=24
11 9.0215461... 200a:b8d:1014:1... 200a:b8d:1014:1... ICMP... 134 Destination Unreachable (Port unreachable)
12 14.196301... fe80::200:ff:fe... 200a:b8d:1014:1... ICMP... 86 Neighbor Solicitation for 200a:b8d:1014:1::3 fro...
13 14.196319... 200a:b8d:1014:1... fe80::200:ff:fe... ICMP... 78 Neighbor Advertisement 200a:b8d:1014:1::3 (sol)
14 19.316820... fe80::200:ff:fe... fe80::200:ff:fe... ICMP... 86 Neighbor Solicitation for fe80::200:ff:feaa:2 fr...
15 19.316870... fe80::200:ff:fe... fe80::200:ff:fe... ICMP... 78 Neighbor Advertisement fe80::200:ff:feaa:2 (rtr,...)
16 24.436225... fe80::200:ff:fe... fe80::200:ff:fe... ICMP... 86 Neighbor Solicitation for fe80::200:ff:feaa:4 fr...
17 24.436239... fe80::200:ff:fe... fe80::200:ff:fe... ICMP... 78 Neighbor Advertisement fe80::200:ff:feaa:4 (sol)

▶ Frame 6: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface vethd.0.9c, id 0
▶ Ethernet II, Src: 00:00:00_aa:00:04 (00:00:00:aa:00:04), Dst: 00:00:00_aa:00:02 (00:00:00:aa:00:02)
└ Internet Protocol Version 6, Src: 200a:b8d:1014:1::3, Dst: 200a:b8d:1014:1::1
    0110 .... = Version: 6
    ⋮ .... 0000 0000 .... .... .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... .... 0101 1111 0000 1010 0110 = Flow Label: 0x5f0a6
    Payload Length: 32
    Next Header: UDP (17)
    Hop Limit: 1
    Source: 200a:b8d:1014:1::3
    Destination: 200a:b8d:1014:1::1
    User Datagram Protocol, Src Port: 40217, Dst Port: 33434
    Data (24 bytes)

0000 00 00 00 aa 00 02 00 00 00 aa 00 04 86 dd 60 05  .....
0010 f0 a6 00 20 11 01 20 0a 0b 8d 10 14 00 01 00 00  .....
0020 00 00 00 00 03 20 0a 0b 8d 10 14 00 01 00 00  .....
0030 00 00 00 00 00 01 9d 19 82 9a 00 20 91 81 00 00  .....
0040 00 25 00 00 00 01 3a 13 00 00 00 00 00 00 f7 e0  %.....
0050 a5 02 00 00 00 00  .....


```

El formato de los segmentos UDP es igual al de las otras pruebas que se han hecho con esta herramienta, sólo cambian las direcciones IPv6 del datagrama IPv6:

- Source (128 bits): 200a:b8d:1014:1::3 (n13)

- Destination (128 bits): 200a:b8d:1014:1::1 (n5)

El formato de los mensajes ICMP es el mismo que en la prueba de n13/n11, pero cambian las IPv6 de origen y destino por tratarse de una respuesta:

- Source (128 bits): 200a:b8d:1014:1::1 (n5)
- Destination (128 bits): 200a:b8d:1014:1::3 (n13)

## RED C

Para la red C se probó la conexión de n9 a n14 y se capturó el tráfico en las interfaces de cada uno de los dispositivos conectados a la red. Se ejecutaron ping y traceroute.

### Ping n9 a n14:

```

1 0.0000000... fe80::200:ff:fe.. fe80::200:ff:fe... ICMP... 86 Neighbor Solicitation for fe80::200:ff:fea:19 f...
2 0.0000007... fe80::200:ff:fe.. fe80::200:ff:fe... ICMP... 78 Neighbor Advertisement fe80::200:ff:fea:19 (rtr...
3 4.1855423... 200a:b9d:1014:f.. 200a:b9d:1014:f.. ICMP... 118 Echo (ping) request id=0x002d, seq=1, hop limit=...
4 4.1855893... 200a:b9d:1014:f.. 200a:b9d:1014:f.. ICMP... 118 Echo (ping) reply id=0x002d, seq=1, hop limit=63...
5 5.2160558... 200a:b9d:1014:f.. 200a:b9d:1014:f.. ICMP... 118 Echo (ping) request id=0x002d, seq=2, hop limit=...
6 5.2160921... 200a:b9d:1014:f.. 200a:b9d:1014:f.. ICMP... 118 Echo (ping) reply id=0x002d, seq=2, hop limit=63...
7 6.2402454... 200a:b9d:1014:f.. 200a:b9d:1014:f.. ICMP... 118 Echo (ping) request id=0x002d, seq=3, hop limit=...
8 6.2402808... 200a:b9d:1014:f.. 200a:b9d:1014:f.. ICMP... 118 Echo (ping) reply id=0x002d, seq=3, hop limit=63...
9 7.2636742... 200a:b9d:1014:f.. 200a:b9d:1014:f.. ICMP... 118 Echo (ping) request id=0x002d, seq=4, hop limit=...
10 7.2637119... 200a:b9d:1014:f.. 200a:b9d:1014:f.. ICMP... 118 Echo (ping) reply id=0x002d, seq=4, hop limit=63...
11 8.2875900... 200a:b9d:1014:f.. 200a:b9d:1014:f.. ICMP... 118 Echo (ping) request id=0x002d, seq=5, hop limit=...
Frame 1: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface veth9.0.f8, id 0
Ethernet II, Src: 00:00:00_aa:00:1a (00:00:00:aa:00:1a), Dst: 00:00:00_aa:00:19 (00:00:00:aa:00:19)
Internet Protocol Version 6, Src: fe80::200:ff:fea:1a, Dst: fe80::200:ff:fea:19
Internet Control Message Protocol v6

```

0000	00 00 00 aa 00 19 00 00 00 aa 00 1a 86 dd 60 00	.....
0010	00 00 00 20 3a ff fe 80 00 00 00 00 00 02 00	.... : .....
0020	00 ff fe aa 00 1a fe 80 00 00 00 00 00 02 00	.....
0030	00 ff fe aa 00 19 87 00 76 13 00 00 00 00 fe 80	..... v .....
0040	00 00 00 00 00 02 00 00 ff fe aa 00 19 01 01	.....
0050	00 00 00 aa 00 1a	.....

Los paquetes de echo request tienen el siguiente formato:

- Tipo (1 byte): 128 (echo request)
- Código (1 byte): 0
- Checksum (2 bytes)
- ID (2 bytes)
- No Secuencia (2 bytes): 1
- Datos (56 bytes)

Los paquetes de echo reply tienen el siguiente formato:

- Tipo (1 byte): 129 (echo reply)

- Código (1 byte): 0
- Checksum (2 bytes)
- ID (2 bytes)
- No Secuencia (2 bytes): 1
- Datos (56 bytes)

Traceroute de n9 a n14:

```

6 3.2078260... 200a:b9d:1014:f... 200a:b9d:1014:f... ICMP... 134 Time Exceeded (hop limit exceeded in transit)
7 3.2087977... 200a:b9d:1014:f... 200a:b9d:1014:f... UDP 86 48190 → 33434 Len=24
8 3.2088111... 200a:b9d:1014:f... 200a:b9d:1014:f... ICMP... 134 Time Exceeded (hop limit exceeded in transit)
9 3.2088234... 200a:b9d:1014:f... 200a:b9d:1014:f... UDP 86 48190 → 33434 Len=24
10 3.2088274... 200a:b9d:1014:f... 200a:b9d:1014:f... ICMP... 134 Time Exceeded (hop limit exceeded in transit)
11 3.2088367... 200a:b9d:1014:f... 200a:b9d:1014:f... UDP 86 48190 → 33434 Len=24
12 3.2088666... 200a:b9d:1014:f... 200a:b9d:1014:f... ICMP... 134 Destination Unreachable (Port unreachable)
13 3.2089184... 200a:b9d:1014:f... 200a:b9d:1014:f... UDP 86 48190 → 33434 Len=24
14 3.2089265... 200a:b9d:1014:f... 200a:b9d:1014:f... ICMP... 134 Destination Unreachable (Port unreachable)
15 3.2089349... 200a:b9d:1014:f... 200a:b9d:1014:f... UDP 86 48190 → 33434 Len=24
16 3.2089400... 200a:b9d:1014:f... 200a:b9d:1014:f... ICMP... 134 Destination Unreachable (Port unreachable)

▶ Frame 1: 180 bytes on wire (1440 bits), 180 bytes captured (1440 bits) on interface veth9.0.f8, id 0
▶ Ethernet II, Src: 6e:ca:44:a3:bb:ce (6e:ca:44:a3:bb:ce), Dst: IPv6mcast_fb (33:33:00:00:00:fb)
└ Internet Protocol Version 6, Src: fe80::6cca:44ff:fea3:bbce, Dst: ff02::fb
    0110 .... = Version: 6
    .... 0000 0000 .... .... .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... .... 0101 1011 1111 1100 1111 = Flow Label: 0x5bfcf
    Payload Length: 126
    Next Header: UDP (17)
    Hop Limit: 255
    Source: fe80::6cca:44ff:fea3:bbce
    Destination: ff02::fb
└ User Datagram Protocol, Src Port: 5353, Dst Port: 5353
    Source Port: 5353
    Destination Port: 5353
    Length: 126
    Checksum: 0x6b1a [Unverified]
0000 33 33 00 00 00 fb 6e ca 44 a3 bb ce 86 dd 60 05 33 .....n. D ..... .
0010 bf cf 00 7e 11 ff fe 80 00 00 00 00 00 00 6c ca .....~....1.
0020 44 ff fe a3 bb ce ff 02 00 00 00 00 00 00 00 00 D..... .
0030 00 00 00 00 00 fb 14 e9 14 e9 00 7e 6b 4a 00 00 ..... .~kJ.
0040 00 00 00 07 00 00 00 00 00 00 04 5f 66 74 70 04 ..... ._ftp.
0050 5f 74 63 70 05 6c 6f 63 61 6c 00 00 0c 00 01 04 _tcp.loc.al. .....
0060 5f 6e 66 73 c0 11 00 0c 00 01 0b 5f 61 66 70 6f _nfs. ....afpo
0070 76 65 72 74 63 70 c0 11 00 0c 00 01 04 5f 73 6d vertcp. ....sm
0080 62 c0 11 00 0c 00 01 09 5f 73 66 74 70 2d 73 73 b. ...._sftp-ss
0090 68 c0 11 00 0c 00 01 08 5f 77 65 62 64 61 76 73 h. ...._webdavs
00a0 c0 11 00 0c 00 01 07 5f 77 65 62 64 61 76 c0 11 ..... webdav. .
00b0 00 0c 00 01 ..... .

```

El formato de los segmentos UDP es igual al de las otras pruebas que se han hecho con esta herramienta, sólo cambian las direcciones IPv6 del datagrama IPv6:

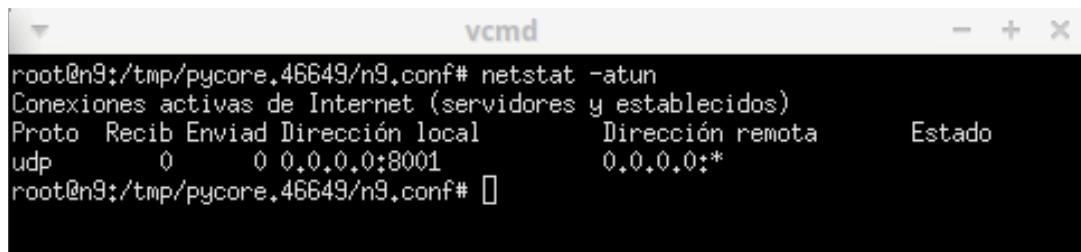
- Source (128 bits): 200a:b9d:1014:f0::2 (n9)
- Destination (128 bits): 200a:b9d:1014:f1::2 (n14)

## Práctica 4 - Ejercicio 22

a) Levantar un servicio UDP con la herramienta nc (netcat) en el host n9 y enviar información desde n13 usando el mismo comando. Ver el estado de los sockets en ambos extremos. ¿Qué significa el estado ESTABLISHED en UDP?

En n9, desde la consola, se ejecutó el siguiente comando para levantar el servidor UDP en modo escucha en el puerto 8001 mediante “nc -ul 8001”. La opción -l indica que el puerto se pone en modo escucha y -u indica que se va a utilizar el protocolo UDP.

Por otra parte, en n13 se ejecutó “nc -u 14.200.12.2 8001”, que se conectó en modo UDP (-u) al puerto 8001 de 14.200.12.2 (n9). Se abrió una nueva terminal en cada uno de los nodos y se ejecutó netstat para ver el estado de las conexiones. Las opciones -t y -u muestran los puertos TCP y UDP, respectivamente.

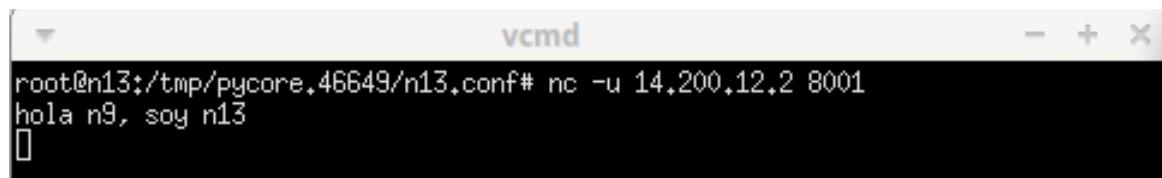


```
root@n9:/tmp/pycore.46649/n9.conf# netstat -atun
Conexiones activas de Internet (servidores y establecidos)
Proto Recib Enviad Dirección local      Dirección remota      Estado
udp    0      0 0.0.0.0:8001          0.0.0.0:*
root@n9:/tmp/pycore.46649/n9.conf#
```

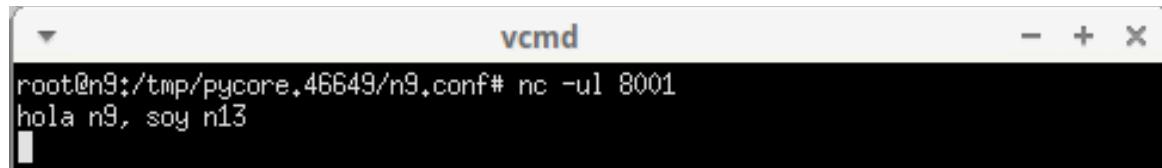


```
root@n13:/tmp/pycore.46649/n13.conf# netstat -atun
Conexiones activas de Internet (servidores y establecidos)
Proto Recib Enviad Dirección local      Dirección remota      Estado
udp    0      0 14.200.8.3:42424     14.200.12.2:8001    ESTABLECIDO
root@n13:/tmp/pycore.46649/n13.conf#
```

Una vez establecida la conexión entre los hosts, se envía un mensaje desde n13 que es recibido por n9.



```
root@n13:/tmp/pycore.46649/n13.conf# nc -u 14.200.12.2 8001
hola n9, soy n13
[]
```



```
root@n9:/tmp/pycore.46649/n9.conf# nc -ul 8001
hola n9, soy n13
[]
```

En UDP el estado ESTABLISHED indica que la conexión fue establecida entre ambos extremos. Un estado vacío indica la falta de conexión o que el puerto se encuentra en modo escucha (la utilidad netstat sólo tiene dos estados para UDP: vacío y establecido).

**b) Levantar en el super daemon inetd o similar con el servicio UDP echo y probar el cliente programado en lenguaje de su elección mediante la API socket contra este servicio. Inspeccionar el estado. Ver de generar datos hacia el “servidor” desde más de un Nodo.**

Para levantar el super daemon inetd se abrió ”/etc/inetd.conf” desde la terminal del nodo n13 y se agregó la línea “echo dgram udp nowait root internal”. Luego se reinició inet

para aplicar los cambios y se comprobó que el servicio udp estaba funcionando en el puerto 7 de n13.

```
root@n10:/tmp/pycore.42999/n10.conf# echo "prueba echo" | nc -u 14.200.8.3 7
prueba echo
```

A continuación se programó el cliente “udp\_cliente.py” en lenguaje python en el nodo n10, el cual envía un mensaje al puerto 7 del nodo n13 y espera recibir una respuesta.

```
import socket

server_address = ('14.200.8.3', 7)
message = b'Hola desde el cliente UDP'

with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as sock:
    print(f'Enviando: {message}')
    sock.sendto(message, server_address)

    data, server = sock.recvfrom(4096)
    print(f'Recibido: {data}')


~
```

Luego se ejecutó el cliente mediante “python3 udp\_cliente.py” y se observó lo siguiente:

```
root@n10:/tmp/pycore.42999/n10.conf# python3 cliente_udp.py
Enviando: b'Hola desde el cliente UDP'
Recibido: b'Hola desde el cliente UDP'
root@n10:/tmp/pycore.42999/n10.conf#
```

Para generar datos hacia el servidor desde más de un nodo se creó el mismo programa “udp\_cliente.py” en los nodos n9 y n11 y al ejecutarlos se obtuvo:

```
root@n9:/tmp/pycore.42999/n9.conf# python3 cliente_udp.py
Enviando: b'Hola desde el cliente UDP'
Recibido: b'Hola desde el cliente UDP'
root@n9:/tmp/pycore.42999/n9.conf#
```

```
root@n11:/tmp/pycore.42999/n11.conf# python3 cliente_udp.py
Enviando: b'Hola desde el cliente UDP'
Recibido: b'Hola desde el cliente UDP'
root@n11:/tmp/pycore.42999/n11.conf#
```

**c) Enviar información desde n13 a n9 a un port UDP donde no existe un proceso esperando por recibir datos. ¿Cómo notifica el stack TCP/IP de este hecho? Investigue la herramienta traceroute que ports utiliza y cómo usa estos mensajes (Ver ejercicio de IP con ruteo estático).**

Estas pruebas se realizan ejecutando el siguiente comando para conectar el host n13 al puerto 8008 de n9, que no tiene conexiones vigentes.

```
echo "hola" | nc -u 14.200.14.2 8008
```

Este comando envía un mensaje para probar la conexión. El control es devuelto a la terminal porque no existe el enlace.

En el fondo, n9 devuelve a n13 un mensaje ICMP de tipo 3, código 3 (destino inalcanzable, puerto inalcanzable) como respuesta al segmento UDP que n13 le había enviado al puerto 8008. Este mensaje contiene el datagrama IP completo del segmento UDP recibido encapsulado en el campo de datos del ICMP.

Cuando se intentan enviar datos a un puerto UDP donde no hay un proceso esperando, el stack TCP/IP no envía ninguna notificación al emisor de manera activa, ya que UDP es un protocolo sin conexión. UDP no realiza ninguna verificación sobre la recepción del paquete, no hay una confirmación ni una retransmisión si no se recibe una respuesta. Simplemente, el paquete se envía y si no hay un servicio que lo reciba, el mismo se pierde.

Traceroute es una herramienta que permite rastrear la ruta que siguen los paquetes desde un host hasta un destino a través de una red IP. Utiliza mensajes ICMP o, en algunas implementaciones, mensajes UDP para identificar los saltos intermedios. Las implementaciones de Linux utilizan UDP en los puertos 33434 a 33463.

Traceroute envía paquetes con un TTL (Time-To-Live) que se incrementa con cada salto. Inicialmente, el TTL se establece en 1, lo que significa que el primer router intermedio devolverá un paquete ICMP "Time Exceeded". Luego, el TTL se incrementa y se repite el proceso hasta que el paquete alcanza el destino o el número máximo de saltos es alcanzado. Cuando un paquete llega a un router intermedio, el router decrementa el TTL. Si el TTL llega a 0, el router envía un mensaje ICMP "Time Exceeded" de vuelta al origen. Esto permite que el traceroute registre el salto y la dirección IP del router. Cuando el paquete llega al destino, si es un paquete UDP, el destino devolverá un mensaje ICMP "Destination Unreachable" con el código "Port Unreachable" si no hay ningún servicio escuchando en el puerto solicitado (por ejemplo, el puerto 33434).

**d) Para las pruebas anteriores capturar tráfico y ver el formato de los datagramas UDP y como se encapsulan en IP.**

#### PRUEBAS DEL INCISO A

Se capturó el tráfico en la interfaz eth0 de n9 y se filtraron las capturas para que solo aparezcan los paquetes UDP. En la primera imagen puede verse el contenido del segmento

UDP con dirección IP origen 14.200.8.3 (n13) y dirección IP destino 14.200.12.2 (n9) , cuyos campos son:

- Puerto origen: 58916 (n13)
- Puerto destino: 8001 (n9)
- Longitud: 25 bytes (cabecera UDP + datos)
- Checksum (2 bytes)
- Datos (17 bytes): mensaje enviado por n13

Este segmento se encapsula en un datagrama IPv4, donde sus campos tienen los siguientes valores:

- Versión: 4 (IPv4)
- IHL (Header Length): 20 bytes
- Longitud del datagrama: 45 bytes
- ID: 48735
- Flags: M = 0, D = 1
- Fragment Offset: 0
- TTL: 59
- Protocolo: 17 (UDP)
- Checksum
- IP origen: 14.200.8.3 (n13)
- IP destino: 14.200.12.2 (n9)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	14.200.8.3	14.200.12.2	UDP	59	58916 → 8001 Len=17

Frame 1: 59 bytes on wire (472 bits), 59 bytes captured (472 bits) on interface veth9.0.33, id 0  
Ethernet II, Src: 00:00:00\_aa:00:19 (00:00:00:aa:00:19), Dst: 00:00:00\_aa:00:1a (00:00:00:aa:00:1a)  
Internet Protocol Version 4, Src: 14.200.8.3, Dst: 14.200.12.2  
    0100 .... = Version: 4  
    .... 0101 = Header Length: 20 bytes (5)  
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)  
    Total Length: 45  
    Identification: 0xbe5f (48735)  
    Flags: 0x4000, Don't fragment  
    Fragment offset: 0  
    Time to live: 59  
    Protocol: UDP (17)  
    Header checksum: 0x4fcc [validation disabled]  
    [Header checksum status: Unverified]  
    Source: 14.200.8.3  
    Destination: 14.200.12.2  
User Datagram Protocol, Src Port: 58916, Dst Port: 8001  
    Source Port: 58916  
    Destination Port: 8001  
    Length: 25  
    Checksum: 0xaec8 [unverified]  
    [Checksum Status: Unverified]  
    [Stream index: 0]  
    [Timestamps]  
    Data (17 bytes)

## PRUEBA INCISO B

Se capturó el tráfico en la interfaz eth0 de n13 (servidor), y se enviaron los mensajes desde n9, n10 y n11 (cliente). En la captura pueden observarse los tres pares de paquetes UDP ECHO.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	14.200.11.10	14.200.8.3	ECHO	67	Request
2	0.000047393	14.200.8.3	14.200.11.10	ECHO	67	Response
3	5.001669318	00:00:00_aa:00:04	00:00:00_aa:00:02	ARP	42	Who has 14.200.8.1? Tell 14.200.8.3
4	5.002752186	00:00:00_aa:00:02	00:00:00_aa:00:04	ARP	42	Who has 14.200.8.3? Tell 14.200.8.1
5	5.002873224	00:00:00_aa:00:02	00:00:00_aa:00:04	ARP	42	14.200.8.1 is at 00:00:00:aa:00:04
6	5.002849086	00:00:00_aa:00:04	00:00:00_aa:00:02	ARP	42	14.200.8.3 is at 00:00:00:aa:00:04
7	38.765282306	14.200.12.2	14.200.8.3	ECHO	67	Request
8	38.765322071	14.200.8.3	14.200.12.2	ECHO	67	Response
9	43.913712778	00:00:00_aa:00:04	00:00:00_aa:00:02	ARP	42	Who has 14.200.8.1? Tell 14.200.8.3
10	43.913808979	00:00:00_aa:00:02	00:00:00_aa:00:04	ARP	42	Who has 14.200.8.3? Tell 14.200.8.1
11	43.913900053	00:00:00_aa:00:02	00:00:00_aa:00:04	ARP	42	14.200.8.1 is at 00:00:00:aa:00:04
12	43.913866404	00:00:00_aa:00:04	00:00:00_aa:00:02	ARP	42	14.200.8.3 is at 00:00:00:aa:00:04
13	86.320989931	14.200.8.2	14.200.8.3	ECHO	67	Request
14	86.321023019	14.200.8.3	14.200.8.2	ECHO	67	Response
15	91.529754955	00:00:00_aa:00:04	00:00:00_aa:00:03	ARP	42	Who has 14.200.8.2? Tell 14.200.8.3
16	91.529872301	00:00:00_aa:00:03	00:00:00_aa:00:04	ARP	42	Who has 14.200.8.3? Tell 14.200.8.2
17	91.529900013	00:00:00_aa:00:03	00:00:00_aa:00:04	ARP	42	14.200.8.2 is at 00:00:00:aa:00:04
18	91.529896456	00:00:00_aa:00:04	00:00:00_aa:00:03	ARP	42	14.200.8.3 is at 00:00:00:aa:00:04

Al observar los campos de uno de los paquetes enviados por los clientes puede verse que el puerto destino de n13 es el 7, el cual es estándar para el servicio ECHO.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	14.200.11.10	14.200.8.3	ECHO	67	Request
2	0.000047393	14.200.8.3	14.200.11.10	ECHO	67	Response
Frame 1: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface vethd.0.50, id 0						
Ethernet II, Src: 00:00:00_aa:00:02 (00:00:00:aa:00:02), Dst: 00:00:00_aa:00:04 (00:00:00:aa:00:04)						
Internet Protocol Version 4, Src: 14.200.11.10, Dst: 14.200.8.3						
User Datagram Protocol, Src Port: 47241, Dst Port: 7						
Source Port: 47241						
Destination Port: 7						
Length: 33						
Checksum: 0xa540 [unverified]						
[Checksum Status: Unverified]						
[Stream index: 0]						
[Timestamps]						
Echo						

### PRUEBA INCISO C

Se capturó el tráfico en la interfaz eth0 de n9 y se obtuvieron dos paquetes: el paquete UDP con el mensaje que n13 envía a n9 y el paquete ICMP “destino/puerto inalcanzable” que se envía a n13 como respuesta porque el puerto al que mandó el mensaje no está escuchando.

Los campos del segmento UDP tienen los siguientes valores:

- Puerto origen: 36033
- Puerto destino: 8008
- Longitud: 13 bytes (cabecera UDP + datos)
- Checksum
- Datos (5 bytes): mensaje enviado por n13

Este segmento se encapsula en un datagrama IPv4, cuyos campos tiene los siguientes valores:

- Versión: 4 (IPv4)
- IHL (Header Length): 20 bytes
- Longitud del datagrama: 33 bytes
- ID: 7760
- Flags: M = 0, D = 1

- TTL: 59
- Protocolo: 17 (UDP)
- Checksum
- IP origen: 14.200.8.3 (n13)
- IP destino: 14.200.12.2 (n9)

No.	Time	Source	Destination	Protocol	Length	Info
3	0.000024924	14.200.8.3	14.200.12.2	UDP	47	36033 → 8008 Len=5
4	0.000037509	14.200.12.2	14.200.8.3	ICMP	75	Destination unreachable

Frame 3: 47 bytes on wire (376 bits), 47 bytes captured (376 bits) on interface veth9.0.1d, id 0  
Ethernet II, Src: 00:00:00\_aa:00:19 (00:00:00:aa:00:19), Dst: 00:00:00\_aa:00:1a (00:00:00:aa:00:1a)  
Internet Protocol Version 4, Src: 14.200.8.3, Dst: 14.200.12.2  
    0100 .... = Version: 4  
    .... 0101 = Header Length: 20 bytes (5)  
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)  
    Total Length: 33  
    Identification: 0x1e50 (7760)  
    Flags: 0x4000, Don't fragment  
    Fragment offset: 0  
    Time to live: 59  
    Protocol: UDP (17)  
    Header checksum: 0xeafe7 [validation disabled]  
    [Header checksum status: Unverified]  
    Source: 14.200.8.3  
    Destination: 14.200.12.2  
User Datagram Protocol, Src Port: 36033, Dst Port: 8008  
    Source Port: 36033  
    Destination Port: 8008  
    Length: 13  
    Checksum: 0x4365 [unverified]  
    [Checksum Status: Unverified]  
    [Stream index: 0]  
    [Timestamps]  
Data (5 bytes)

Por otro lado, los campos del mensaje ICMP tiene los siguientes valores:

- Tipo: 3 (destino inalcanzable)
- Código: 3 (puerto inalcanzable)
- Checksum
- Cabecera IPv4 (20 bytes)
- Cabecera UDP (8 bytes)
- Datos (5 bytes)

El mensaje ICMP se encapsula en un datagrama IP, con los valores de sus campos de la siguiente manera:

- Versión: 4 (IPv4)
- IHL: 20 bytes
- Longitud del datagrama: 61 bytes (cabecera IP + datos)
- ID (2 bytes): 12525
- Flags (3 bits): M = 0, D = 0
- TTL: 64
- Protocolo: 1 (ICMP)
- Checksum
- IP origen: 14.200.12.2 (n9)
- IP destino: 14.200.8.3 (n13)

No.	Time	Source	Destination	Protocol	Length	Info
3	0.000024924	14.200.8.3	14.200.12.2	UDP	47	36033 → 8008 Len=5
4	0.000037509	14.200.12.2	14.200.8.3	ICMP	75	Destination unreachable

Frame 4: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface veth9.0.1d, id 0  
 Ethernet II, Src: 00:00:00\_aa:00:1a (00:00:00:aa:00:1a), Dst: 00:00:00\_aa:00:19 (00:00:00:aa:00:19)  
 Internet Protocol Version 4, Src: 14.200.12.2, Dst: 14.200.8.3  
 0100 .... = Version: 4  
 .... 0101 = Header Length: 20 bytes (5)  
 Differentiated Services Field: 0xc0 (DSCP: CS6, ECN: Not-ECT)  
 Total Length: 61  
 Identification: 0x201b (8219)  
 Flags: 0x0000  
 Fragment offset: 0  
 Time to live: 64  
 Protocol: ICMP (1)  
 Header checksum: 0x2851 [validation disabled]  
 [Header checksum status: Unverified]  
 Source: 14.200.12.2  
 Destination: 14.200.8.3  
 Internet Control Message Protocol  
 Type: 3 (Destination unreachable)  
 Code: 3 (Port unreachable)  
 Checksum: 0x2eb0 [correct]  
 [Checksum Status: Good]  
 Unused: 00000000  
 Internet Protocol Version 4, Src: 14.200.8.3, Dst: 14.200.12.2  
 User Datagram Protocol, Src Port: 36033, Dst Port: 8008  
 Data (5 bytes)

## Práctica 4 - Ejercicio 23

a) Programar en el lenguaje de su elección mediante la API socket un servidor TCP que escuche conexiones en el puerto 9000 y que los datos que reciba los descarte. Correr en el nodo n9 y enviar datos desde otro nodo usando la herramienta nc (netcat) o telnet.

Se programó y ejecutó el siguiente servidor de nombre “tcp\_servidor.py” en lenguaje python en el nodo n9. El mismo escucha conexiones en el puerto 9000 y descarta los datos que recibe, pero escribe en la terminal la dirección IPv4 y el puerto del que recibió el mensaje.

```

import socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = '0.0.0.0'
port = 9000
server_socket.bind((host, port))
server_socket.listen(5)
print(f"Servidor escuchando en {host}:{port}...")

while True:
    client_socket, client_address = server_socket.accept()
    print(f"Conexion recibida de {client_address}")
    data=client_socket.recv(1024)
    client_socket.close()

```

Al ejecutar el comando “echo “hola” | nc 14.200.12.2 9000” en la terminal del nodo n13 se obtuvo el siguiente resultado, en el cual puede observarse que n9 recibe una conexión de 14.200.8.3 (n13).

The image displays two terminal windows side-by-side. Both windows have a title bar labeled "Terminal -" and a menu bar with "Archivo", "Editar", "Ver", "Terminal", "Pestañas", and "Ayuda".

**Terminal on Node n13:**

```

root@n13:/tmp/pycore.35985/n13.conf# echo "hola" | nc 14.200.12.2 9000
root@n13:/tmp/pycore.35985/n13.conf#

```

**Terminal on Node n9:**

```

root@n9:/tmp/pycore.35985/n9.conf# sudo python3 servidor_tcp.py
sudo: imposible resolver el anfitrión n9: Fallo temporal en la resolución del nombre
Servidor escuchando en 0.0.0.0:9000...
Conexion recibida de ('14.200.8.3', 59452)

```

Se realizó una prueba similar desde n10, cuya dirección IPv4 es 14.200.11.10 y se obtuvo:

The image shows two terminal windows. The top window, titled 'Terminal -', has the command 'echo "hola servidor" | nc 14.200.12.2 9000' and its output 'root@n10:/tmp/pycore.35985/n10.conf#'. The bottom window, also titled 'Terminal -', has the command 'sudo python3 servidor\_tcp.py' and its output: 'root@n9:/tmp/pycore.35985/n9.conf# sudo: imposible resolver el anfitrión n9: Fallo temporal en la resolución del nombre', followed by 'Servidor escuchando en 0.0.0.0:9000...', 'Conexion recibida de ('14.200.8.3', 59452)', and 'Conexion recibida de ('14.200.11.10', 43904)'.

**b) En el nodo n9 levantar con el super daemon inetd algunos servicios extras, como tcp echo, discard y otros. Chequear los servicios TCP y UDP activos.**

Se abrió el archivo “/etc/inetd.conf” y se verificó que estuvieran las siguientes líneas para activar los servicios ECHO y DISCARD de tcp y udp, de las cuales algunas fueron agregadas y otras descomentadas:

```
echo stream tcp nowait root internal
echo dgram udp wait root internal
discard stream tcp nowait root internal
discard dgram udp wait root internal
```

Con el comando “netstat -tuln” se comprobó que los servicios echo (puerto 7) y discard (puerto 9) están activos para tcp y udp.

The terminal window shows the command 'root@n9:/tmp/pycore.42999/n9.conf# sudo netstat -tuln' and its output: 'root@n9:/tmp/pycore.42999/n9.conf# sudo: imposible resolver el anfitrión n9: Fallo temporal en la resolución del nombre', 'Conexiones activas de Internet (solo servidores)', and a table of active connections:

Proto	Recib	Enviad	Dirección local	Dirección remota	Estado
tcp	0	0	0.0.0.0:7	0.0.0.0:*	ESCUCHAR
tcp	0	0	0.0.0.0:9	0.0.0.0:*	ESCUCHAR
udp	0	0	0.0.0.0:7	0.0.0.0:*	
udp	0	0	0.0.0.0:9	0.0.0.0:*	

Se realizaron pruebas desde n11 para probar el funcionamiento del ECHO tanto en tcp como udp. Puede observarse que se devuelve el nodo origen el mismo mensaje que fue enviado al destino.A

```
root@n11:/tmp/pycore.42999/n11.conf# echo "prueba tcp" | nc 14.200.12.2 7
prueba tcp
```

```
root@n11:/tmp/pycore.42999/n11.conf# echo "prueba udp" | nc -u 14.200.12.2 7
prueba udp
```

También se probó el DISCARD mediante “echo “prueba TCP” | nc 14.200.12.2 9” y “echo “prueba UDP” | nc -u 14.200.12.2 9”. No se obtuvo ninguna respuesta porque los mensajes fueron descartados por n9.

**c) Desde el nodo n13 realizar una conexión TCP y enviar datos mediante un programa cliente de su elección al servicio discard, y al echo. Capturar el tráfico con la herramienta tcpdump o wireshark y analizar la cantidad de segmentos, los flags utilizados y las opciones extras que llevan los encabezados tcp.**

Se programó en n13 en lenguaje python el programa “tcp\_cliente.py”, el cual envía mensajes a los puertos 7 (ECHO) y 9 (DISCARD) del nodo n9 y, en el caso del puerto 7, espera una respuesta.

```
#Se debe poner en primer lugar la dirección IP origen
#y luego poner entre espacios los puertos con los que se
#quiera hacer la conexión en orden ascendente.
#Solo están configurados los puertos 7 y 9.

import socket
import argparse

#Configurar el parser de argumentos
parser = argparse.ArgumentParser(description="Se debe indicar IP
origen")
parser.add_argument("servidor", type=str, help="Dirección IP del
servidor de origen")
parser.add_argument("puertos", type=int, nargs="+", help="Puertos a los
que conectarse (puede ser más de uno)")

args = parser.parse_args();
source_servidor = args.servidor;
# Dirección IP y puertos del nodo servidor (n9)
servidor = "14.200.12.2" # Dirección IP de n9
puertos = args.puertos;
if (len(puertos) == 1):
    puerto=puertos[0]
    if puertos[0]==7:
```

```
print(f"Conectando al puerto {puerto}...")
# Crear socket TCP
cliente_7 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
cliente_7.connect((servidor, puerto))

# Enviar datos al servidor en puerto 7
mensaje_7 = f"Prueba desde {source_servidor} al servicio en
puerto {puerto}\n"
cliente_7.sendall(mensaje_7.encode())

# Recibir respuesta (solo para echo)
respuesta_7 = cliente_7.recv(1024)
print(f"Respuesta del servidor en puerto {puerto}:
{respuesta_7.decode()}")
input("Conexión abierta. Presione Enter para cerrar la
conexión...")

# Cerrar conexión
cliente_7.close()

else:
    # Crear y conectar al puerto 9 (discard)

    print(f"Conectando al puerto {puerto}...")
    cliente_9 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    cliente_9.connect((servidor, puerto))

    # Enviar datos al servidor en puerto 9
    mensaje_9 = f"Prueba al servidor {servidor} en puerto
{puerto}\n"
    cliente_9.sendall(mensaje_9.encode())
    # Mantener ambas conexiones abiertas hasta que el usuario
decida cerrarlas
    input("Conexión abierta. Presione Enter para cerrar la
conexión...")

    # Cerrar las conexiones
    cliente_9.close()

else:
    puerto = puertos[0]
    print(f"Conectando al puerto {puerto}...")
    # Crear socket TCP
    cliente_7 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    cliente_7.connect((servidor, puerto))
```

```

# Enviar datos al servidor en puerto 7
mensaje_7 = f"Prueba desde {source_servidor} al servicio en puerto {puerto}\n"
cliente_7.sendall(mensaje_7.encode())

# Recibir respuesta (solo para echo)
respuesta_7 = cliente_7.recv(1024)
print(f"Respuesta del servidor en puerto {puerto}: {respuesta_7.decode() }")

# Crear y conectar al puerto 9 (discard)
puerto = puertos[1]
print(f"Conectando al puerto {puerto}...")
cliente_9 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
cliente_9.connect((servidor, puerto))

# Enviar datos al servidor en puerto 9
mensaje_9 = f"Prueba al servidor {servidor} en puerto {puerto}\n"
cliente_9.sendall(mensaje_9.encode())

# Mantener ambas conexiones abiertas hasta que el usuario decida cerrarlas
input("Conexiones abiertas. Presione Enter para cerrar las conexiones...")

# Cerrar las conexiones
cliente_7.close()
cliente_9.close()

```

Al ejecutarlo se obtiene la siguiente respuesta, donde puede verse que n9 devuelve a n13 el mismo mensaje que este le envía por el puerto 7 y no envía una respuesta al mensaje que n13 envía hacia el puerto

```

root@n13:/tmp/pycore.43805/n13.conf# python3 tcp_cliente_open.py 14.200.8.3
Conectando al puerto 7...
Respuesta del servidor en puerto 7: Prueba desde 14.200.8.3 al servicio en puerto 7

Conectando al puerto 9...
Conexiones abiertas. Presione Enter para cerrar las conexiones...■

```

Se capturó mediante wireshark el tráfico en la interfaz eth0 de n9 al momento de ejecutar “tcp\_cliente.py” desde n13. A continuación se muestran y analizan los segmentos capturados.

No.	Time	Source	Destination	Protocol	Length Info
1	0.0000000000	14.200.8.3	14.200.12.2	TCP	74 56376 - 7 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2687462156 TSecr=0 WS=128
2	0.0000034395	14.200.8.3	14.200.12.2	TCP	74 7 - 56376 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=4044611191 TSecr=2687462156...
3	0.000141999	14.200.8.3	14.200.12.2	TCP	66 56376 - 7 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2687462157 TSecr=4044611191
4	0.001423127	14.200.8.3	14.200.12.2	ECHO	114 Request
5	0.001441568	14.200.12.2	14.200.8.3	TCP	66 7 - 56376 [ACK] Seq=1 Ack=49 Win=65152 Len=0 TSval=4044611192 TSecr=2687462157
6	0.010992366	14.200.12.2	14.200.8.3	ECHO	114 Response
7	0.011186462	14.200.8.3	14.200.12.2	TCP	66 56376 - 7 [ACK] Seq=49 Ack=49 Win=64240 Len=0 TSval=2687462168 TSecr=4044611202
8	0.015277418	14.200.8.3	14.200.12.2	TCP	74 35652 - 9 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2687462172 TSecr=0 WS=128
9	0.015290444	14.200.12.2	14.200.8.3	TCP	74 9 - 35652 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=4044611206 TSecr=2687462172...
10	0.015327502	14.200.8.3	14.200.12.2	TCP	66 35652 - 9 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2687462172 TSecr=4044611206
11	0.015646302	14.200.8.3	14.200.12.2	TCP	109 35652 - 9 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=43 TSval=2687462172 TSecr=4044611206
12	0.015652622	14.200.12.2	14.200.8.3	TCP	66 9 - 35652 [ACK] Seq=1 Ack=44 Win=65152 Len=0 TSval=4044611206 TSecr=2687462172

Los segmentos 1, 2 y 3 establecen la conexión TCP entre el puerto 56376 de n13 (cliente) y el puerto 7 de n9 (servidor). Estos tienen los flags SYN, SYN y ACK, ACK activos respectivamente, y ambos procesos comienzan con el número de secuencia 0. Los segmentos 1 y 2 tienen en su encabezado las opciones:

- Maximum segment Size: 1460 bytes
- SACK permitted
- Timestamps: TSval = 2687462156 y TSecr = 0 para segmento 1. TSval = 4044611191 y TSecr = 2687462156 para segmento 2.
- No-Operation (NOP)
- Window scale: 7 (debe multiplicarse por 128 para obtener el tamaño en bytes)

El segmento 3 tiene las opciones:

- No-Operation (NOP)
- Timestamps: TSval = 2687462156 y TSecr = 4044611191

Los timestamps son números de 32 bits que se incluyen como parte de las opciones TCP en los segmentos TCP. Estas marcas de tiempo permiten medir el tiempo que tarda un paquete en viajar desde un punto de la red hasta otro.

El TSval (Timestamp Value) es el valor de la marca de tiempo generada por el emisor del paquete, mientras que el TSecr (Timestamp Echo Reply) contiene el valor del timestamp enviado por el otro extremo en el último segmento que fue recibido.

El segmento 4 es el request de ECHO de n13 hacia n9 y tiene los flags ACK y PUSH activos y las mismas opciones que el segmento 3. El segmento 5 es el ACK del request y el 6 es la respuesta ECHO con los flags ACK y PUSH activos. Estos dos últimos segmentos tienen las opciones:

- No-Operation (NOP)
- Timestamps: TSval = 4044611192 y TSecr = 2687462157 para segmento 5. TSval = 4044611202 y TSecr = 2687462157 para segmento 6.

Tanto el ECHO request como el ECHO response tienen un payload de 41 bytes. Luego n13 envía el ACK al ECHO response de n9 en el segmento 7, con el flag ACK activado y las mismas opciones:

- No-Operation (NOP)
- Timestamps: TSval = 2687462158 y TSecr = 4044611202

En el segmento 8, n13 comienza la apertura de otra sesión TCP, esta vez al puerto 9 de n9 desde su puerto 35652. Los segmentos 8, 9 y 10 son similares a los segmentos 1, 2 y 3.

Los segmentos 11 y 12 son el mensaje que n13 envía a n9 y su respectivo ACK. Como DISCARD descarta los datos que recibe y no envía respuesta, no hay un segmento con datos enviado desde n9 hacia n13.

13	91.362512017	14.200.8.3	14.200.12.2	TCP	66 56376 → 7	[FIN, ACK]	Seq=49 Ack=49 Win=64256 Len=0 TSval=2687553519 TSecr=4044611202
14	91.362556188	14.200.8.3	14.200.12.2	TCP	66 35652 → 9	[FIN, ACK]	Seq=44 Ack=1 Win=64256 Len=0 TSval=2687553519 TSecr=4044611206
15	91.366844821	14.200.12.2	14.200.8.3	TCP	66 7 → 56376	[FIN, ACK]	Seq=49 Ack=0 Win=65152 Len=0 TSval=4044702557 TSecr=2687553519
16	91.366957738	14.200.8.3	14.200.12.2	TCP	66 56376 → 7	[ACK]	Seq=50 Ack=50 Win=64256 Len=0 TSval=2687553524 TSecr=4044702557
17	91.367424571	14.200.12.2	14.200.8.3	TCP	66 9 → 35652	[FIN, ACK]	Seq=1 Ack=45 Win=65152 Len=0 TSval=4044702558 TSecr=2687553519
18	91.367465980	14.200.8.3	14.200.12.2	TCP	66 35652 → 9	[ACK]	Seq=45 Ack=2 Win=64256 Len=0 TSval=2687553524 TSecr=4044702558

A continuación se indica al programa que deben cerrarse ambas sesiones. Para esto, n13 comienza a cerrar ambas sesiones con los segmentos 13 y 14, enviados al puerto 7 y 9 de n9 respectivamente. Ambos segmentos con los flags FIN y ACK activados. Luego se termina el cierre de conexión ECHO en los segmentos 15 y 16, que tienen los flags FIN y ACK, ACK activos. Finalmente se cierra la sesión DISCARD con los segmentos 17 y 18, los cuales tiene activados los mismos flags que los segmentos 15 y 16.

En las siguientes imágenes pueden verse los mensajes enviados por n13 hacia n9 y el mensaje que n9 responde al ECHO de n13. Puede verse que en el caso del ECHO n9 responde a n13 con el mismo mensaje que este le había enviado.

ECHO request de n13 a n9:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	14.200.8.3	14.200.12.2	TCP	74 56376 → 7	[SYN] Seq=0 Win=64240 Len=0 MSS=1460
2	0.0000034305	14.200.12.2	14.200.8.3	TCP	74 7 → 56376	[SYN, ACK] Seq=0 Ack=1 Win=65160 Len=1460
3	0.000141999	14.200.8.3	14.200.12.2	TCP	66 56376 → 7	[ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2687553524 TSecr=4044702557
4	0.0001423127	14.200.8.3	14.200.12.2	ECHO	114 Request	
5	0.001441568	14.200.12.2	14.200.8.3	TCP	66 7 → 56376	[ACK] Seq=1 Ack=49 Win=65152 Len=0 TSval=4044702558 TSecr=2687553519
6	0.010992366	14.200.12.2	14.200.8.3	ECHO	114 Response	
7	0.011186462	14.200.8.3	14.200.12.2	TCP	66 56376 → 7	[ACK1] Seq=49 Ack=49 Win=64256 Len=0 TSval=2687553524 TSecr=4044702558

Frame 4: 114 bytes on wire (912 bits), 114 bytes captured (912 bits) on interface veth9.0.b6, id 0						
Frame	Start	End	Length	Source	Destination	Info
0000	00 00 00 aa 00 1a 00 00	00 aa 00 19 08 00 45 00	.....E.....			
0010	00 64 1f b5 40 00 3b 06	ee 4a 0e c8 08 03 0e c8	·d·@;·J.....			
0020	0c 02 dc 38 00 07 7c d3	ed 95 0e c1 21 b1 80 18	··8· ··!··			
0030	01 f6 78 6c 00 00 01 01	08 0a a0 2f 6b 0d f1 13	··x1···/k··			
0040	de 77 50 72 75 65 62 61	20 64 65 73 64 65 20 31	wPrueba desde 1			
0050	34 2e 32 30 30 2e 38 2e	33 20 61 6c 20 73 65 72	4.200.8. 3 al ser			
0060	76 69 63 69 6f 20 65 6e	20 70 75 65 72 74 6f 20	vicio en puerto			
0070	37 0a		7·			

ECHO response de n9 a n13:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	14.200.8.3	14.200.12.2	TCP	74	56376 → 7 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
2	0.000034305	14.200.12.2	14.200.8.3	TCP	74	7 → 56376 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 TS
3	0.000141999	14.200.8.3	14.200.12.2	TCP	66	56376 → 7 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TS
4	0.001423127	14.200.8.3	14.200.12.2	ECHO	114	Request
5	0.001441568	14.200.12.2	14.200.8.3	TCP	66	7 → 56376 [ACK] Seq=1 Ack=49 Win=65152 Len=0 TS
6	0.010992366	14.200.12.2	14.200.8.3	ECHO	114	Response
7	0.011186462	14.200.8.3	14.200.12.2	TCP	66	56376 → 7 [ACK] Seq=49 Ack=49 Win=64256 Len=0 TS

Frame 6: 114 bytes on wire (912 bits), 114 bytes captured (912 bits) on interface veth9.0.b6, id 0

0000	00 00 00 aa 00 19 00 00 00 aa 00 1a 08 00 45 00	.....E
0010	00 64 48 7f 40 00 40 06 c0 80 0e c8 0c 02 0e c8	dH @@ .....
0020	08 03 00 07 dc 38 0e c1 21 b1 7c d3 ed c5 00 18	...8..!.. ....
0030	01 fd 78 2a 00 00 01 01 08 0a f1 13 de 82 a0 2f	.x*...../.....
0040	6b 0d 50 72 75 65 62 61 20 64 65 73 64 65 20 31	k-Prueba desde 1
0050	34 2e 32 30 30 2e 38 2e 33 20 61 6c 20 73 65 72	4.200.8.3 al ser
0060	76 69 63 69 6f 20 65 6e 20 70 75 65 72 74 6f 20	vicio en puerto
0070	37 0a	7

Mensaje que n13 envía a n9 y que este último descarta:

6 0.010992366	14.200.12.2	14.200.8.3	ECHO	114 Response
7 0.011186462	14.200.8.3	14.200.12.2	TCP	66 56376 → 7 [ACK] Seq=49 Ack=49 Win=64256 Len=0
8 0.015277418	14.200.8.3	14.200.12.2	TCP	74 35652 → 9 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
9 0.015290445	14.200.12.2	14.200.8.3	TCP	74 9 → 35652 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0
10 0.015327502	14.200.8.3	14.200.12.2	TCP	66 35652 → 9 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TS
11 0.015646302	14.200.8.3	14.200.12.2	TCP	109 35652 → 9 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=0
12 0.015652622	14.200.12.2	14.200.8.3	TCP	66 9 → 35652 [ACK] Seq=1 Ack=44 Win=65152 Len=0 TS
12 01 262512017	14.200.8.2	14.200.12.2	TCP	66 56276 → 7 [FIN, ACK] Seq=40 Ack=40 Win=64256

Frame 11: 109 bytes on wire (872 bits), 109 bytes captured (872 bits) on interface veth9.0.b6, id 0

Ethernet II, Src: 00:00:00\_aa:00:19 (00:00:00:aa:00:19), Dst: 00:00:00\_aa:00:1a (00:00:00:aa:00:1a)

Internet Protocol Version 4, Src: 14.200.8.3, Dst: 14.200.12.2

Transmission Control Protocol, Src Port: 35652, Dst Port: 9, Seq: 1, Ack: 1, Len: 43

Source Port: 35652  
Destination Port: 9

0000	00 00 00 aa 00 1a 00 00 00 aa 00 19 08 00 45 00	.....E
0010	00 5f 99 ec 40 00 3b 06 74 18 0e c8 08 03 0e c8	..@; t.....
0020	0c 02 8b 44 00 09 ab cd 5f bc d6 7a 3b 23 80 18	..D....z;#..
0030	01 f6 32 fe 00 00 01 01 08 0a a0 2f 6b 1c f1 13	..2...../k...
0040	de 86 50 72 75 65 62 61 20 61 6c 20 73 65 72 76	..Prueba al serv
0050	69 64 6f 72 20 31 34 2e 32 30 30 2e 31 32 32 2e 32	idor 14.200.12.2
0060	20 65 6e 20 70 75 65 72 74 6f 20 39 0a	en puer to 9.

#### d) Sin cerrar las conexiones chequear los servicios activos y ver los Estados.

Sin cerrar las conexiones y ejecutando “netstat -atun” en n9 y n13, se obtuvieron los siguientes resultados:

```
root@n9:/tmp/pycore.43805/n9.conf# netstat -atun
Conexiones activas de Internet (servidores y establecidos)
Proto Recib Enviad Dirección local           Dirección remota      Estado
tcp      0      0 0.0.0.0:9                 0.0.0.0:*                ESCUCHAR
tcp      0      0 0.0.0.0:7                 0.0.0.0:*                ESCUCHAR
tcp      0      0 14.200.12.2:9              14.200.8.3:50216        ESTABLECIDO
tcp      0      0 14.200.12.2:7              14.200.8.3:35224        ESTABLECIDO
udp      0      0 0.0.0.0:7                 0.0.0.0:*                ESCUCHAR
udp      0      0 0.0.0.0:9                 0.0.0.0:*                ESCUCHAR
root@n9:/tmp/pycore.43805/n9.conf#
```

```
root@n13:/tmp/pycore.43805/n13.conf# netstat -atun
Conexiones activas de Internet (servidores y establecidos)
Proto Recib Enviad Dirección local           Dirección remota      Estado
tcp      0      0 14.200.8.3:35224          14.200.12.2:7          ESTABLECIDO
tcp      0      0 14.200.8.3:50216          14.200.12.2:9          ESTABLECIDO
root@n13:/tmp/pycore.43805/n13.conf#
```

En la imagen superior, se puede observar que el nodo n9 mantiene dos conexiones abiertas (estado ESTABLECIDO): una en el puerto 7 y otra en el puerto 9, ambas con 14.200.8.3 (n13). También podemos notar que sigue escuchando (estado ESCUCHAR) cualquier mensaje que vaya al puerto 9000, tal como se pedía en el inciso a, y en los propios puertos 7 y 9. En cambio, en la salida del nodo n13 vemos las conexiones que mantiene con n9.

Se puede ver que se corresponden los puertos que aparecen en ambas salidas:

- 14.200.8.3:35224 - 14.200.12.2:7
- 14.200.8.3:50216 - 14.200.12.2:9

**e) Generar nuevas conexiones hacia el nodo n9 e inspeccionar los estados. Por ejemplo realizar varias conexiones simultáneas al servicio tcp echo desde el mismo origen y desde otros nodos.**

En el nodo n13 se abrieron tres conexiones: dos al puerto 7 (echo) y una al puerto 9 (discard) del nodo n9:

```
root@n13:/tmp/pycore.43805/n13.conf# python3 tcp_cliente_open.py 14.200.8.3 7
Conectando al puerto 7...
Respuesta del servidor en puerto 7: Prueba desde 14.200.8.3 al servicio en puerto 7

Conexión abierta. Presione Enter para cerrar la conexión...[]

root@n13:/tmp/pycore.43805/n13.conf# netstat -atun
Conexiones activas de Internet (servidores y establecidos)
Proto Recib Enviad Dirección local           Dirección remota      Estado
tcp    0      0 14.200.8.3:38696            14.200.12.2:7        ESTABLECIDO
tcp    0      0 14.200.8.3:60820            14.200.12.2:7        ESTABLECIDO
tcp    0      0 14.200.8.3:54512            14.200.12.2:9        ESTABLECIDO
root@n13:/tmp/pycore.43805/n13.conf#
```

En el nodo n10 se abrieron 2 conexiones: una al puerto 7 (echo) y una al puerto 9 (discard) del nodo n9.

```
<n10.conf# python3 ./tcp_cliente_open.py 14.200.11.10 7 9
Conectando al puerto 7...
Respuesta del servidor en puerto 7: Prueba desde 14.200.11.10 al servicio en puerto 7

Conectando al puerto 9...
Conexiones abiertas. Presione Enter para cerrar las conexiones...[]

root@n10:/tmp/pycore.43805/n10.conf# netstat -atun
Conexiones activas de Internet (servidores y establecidos)
Proto Recib Enviad Dirección local           Dirección remota      Estado
tcp    0      0 14.200.11.10:40294          14.200.12.2:7        ESTABLECIDO
tcp    0      0 14.200.11.10:51452          14.200.12.2:9        ESTABLECIDO
root@n10:/tmp/pycore.43805/n10.conf#
```

En el nodo n11 se abrió 1 conexión en el puerto 7 (echo) del nodo n9.

```

<n11.conf# python3 ./tcp_cliente_open.py 14.200.8.2 7
Conectando al puerto 7...
Respuesta del servidor en puerto 7: Prueba desde 14.200.8.2 al servicio en puerto 7

Conexión abierta. Presione Enter para cerrar la conexión...■

root@n11:/tmp/pycore.43805/n11.conf# netstat -atun
Conexiones activas de Internet (servidores y establecidos)
Proto Recib Enviad Dirección local           Dirección remota     Estado
tcp      0      0 14.200.8.2:52826          14.200.12.2:7      ESTABLECIDO
root@n11:/tmp/pycore.43805/n11.conf# ■

```

En el nodo n9 se pueden ver todas las conexiones establecidas:

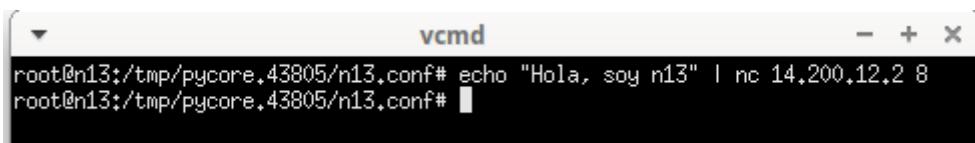
```

root@n9:/tmp/pycore.43805/n9.conf# netstat -atun
Conexiones activas de Internet (servidores y establecidos)
Proto Recib Enviad Dirección local           Dirección remota     Estado
tcp      0      0 0.0.0.0:9                0.0.0.0:*
tcp      0      0 0.0.0.0:7                0.0.0.0:*
tcp      0      0 14.200.12.2:9            14.200.8.3:54512    ESTABLECIDO
tcp      0      0 14.200.12.2:7            14.200.8.3:38696    ESTABLECIDO
tcp      0      0 14.200.12.2:7            14.200.11.10:40294   ESTABLECIDO
tcp      0      0 14.200.12.2:7            14.200.8.2:52826    ESTABLECIDO
tcp      0      0 14.200.12.2:9            14.200.11.10:51452    ESTABLECIDO
tcp      0      0 14.200.12.2:7            14.200.8.3:60820    ESTABLECIDO
udp      0      0 0.0.0.0:7                0.0.0.0:*
udp      0      0 0.0.0.0:9                0.0.0.0:*
root@n9:/tmp/pycore.43805/n9.conf# ■

```

**f) Intentar generar conexiones a un puerto donde no existe un proceso esperando por recibir datos. ¿Cómo notifica TCP de este hecho (ver flags)?**

Se quiso establecer una conexión desde el cliente n13 (14.200.8.3) con el puerto 8 de n9 (14.200.12.2). Como que en este puerto de n9 no se encontraba activo ningún servicio, no se obtuvo ningún resultado en la terminal de n13.



```

root@n13:/tmp/pycore.43805/n13.conf# echo "Hola, soy n13" | nc 14.200.12.2 8
root@n13:/tmp/pycore.43805/n13.conf# ■

```

Al capturar los segmentos TCP con wireshark en la interfaz eth0 de n13, se obtuvo lo siguiente:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	14.200.8.3	14.200.12.2	TCP	74	40694 → 8 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2691368475 TSecr=0 WS=128
2	0.000324957	14.200.12.2	14.200.8.3	TCP	54	8 → 40694 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
3	5.235788795	00:00:00 aa:00:02	00:00:00 aa:00:04	ARP	42	Who has 14.200.8.3? Tell 14.200.8.1
4	5.235771385	00:00:00 aa:00:04	00:00:00 aa:00:02	ARP	42	Who has 14.200.8.1? Tell 14.200.8.3
5	5.235848463	00:00:00 aa:00:04	00:00:00 aa:00:02	ARP	42	14.200.8.3 is at 00:00:00:aa:00:04
6	5.235898251	00:00:00 aa:00:02	00:00:00 aa:00:04	ARP	42	14.200.8.1 is at 00:00:00:aa:00:02

Se puede ver que en el segundo segmento (que tiene origen en n9 y destino n13), junto a ACK, se hace uso del flag RST. Este flag es utilizado para informar al cliente que no se pudo establecer una sesión debido a que (en este caso) no se encontró un servicio activo, por lo que la conexión debe terminar inmediatamente.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	14.200.8.3	14.200.12.2	TCP	74	40604 → 8 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
L	2 0.000324957	14.200.12.2	14.200.8.3	TCP	54	8 → 40604 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
3	5.235788795	00:00:00_aa:00:02	00:00:00_aa:00:04	ARP	42	Who has 14.200.8.3? Tell 14.200.8.1
4	5.235771385	00:00:00_aa:00:04	00:00:00_aa:00:02	ARP	42	Who has 14.200.8.1? Tell 14.200.8.3
5	5.235848463	00:00:00_aa:00:04	00:00:00_aa:00:02	ARP	42	14.200.8.3 is at 00:00:00_aa:00:04
6	5.235898251	00:00:00_aa:00:02	00:00:00_aa:00:04	ARP	42	14.200.8.1 is at 00:00:00_aa:00:02
▶ Frame 2: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface vethd.0.b6, id 0						
▶ Ethernet II, Src: 00:00:00_aa:00:02 (00:00:00_aa:00:04) (00:00:00_aa:00:04)						
▶ Internet Protocol Version 4, Src: 14.200.12.2, Dst: 14.200.8.3						
▼ Transmission Control Protocol, Src Port: 8, Dst Port: 40604, Seq: 1, Ack: 1, Len: 0						
Source Port: 8						
Destination Port: 40604						
[Stream index: 0]						
[TCP Segment Len: 0]						
Sequence number: 1 (relative sequence number)						
Sequence number (raw): 0						
[Next sequence number: 1 (relative sequence number)]						
Acknowledgment number: 1 (relative ack number)						
Acknowledgment number (raw): 2635700496						
0101 .... = Header Length: 20 bytes (5)						
Flags: 0x014 (RST, ACK)						
000. .... = Reserved: Not set						
...0 .... = Nonce: Not set						
.... 0.... = Congestion Window Reduced (CWR): Not set						
.... 0.. = ECN-Echo: Not set						
.... .0.. = Urgent: Not set						
.... .1.. = Acknowledgment: Set						
.... .0... = Push: Not set						
.... .1.. = Reset: Set						
.... ..0.. = Syn: Not set						
.... ...0.. = Fin: Not set						
TCP Flags: ....A.R..1						

También puede verse que en el segmento enviado por n9, el tamaño de la ventana es 0, indicando que no pueden recibirse datos.

### g) Cerrar las conexiones y ver el estado de los servicios en ambos lados. ¿En qué estado queda el que hace el cierre activo?

En n13 se cerraron las conexiones entre el puerto 38696 del n13 y el puerto 7 del n9, y entre el puerto 54512 de n13 y el puerto 9 de n9. Inmediatamente se ejecutó el comando “netstat -atun” y se obtuvo la siguiente salida:

```
root@n13:/tmp/pycore.43805/n13.conf# netstat -atun
Conexiones activas de Internet (servidores y establecidos)
Proto Recib Enviad Dirección local           Dirección remota         Estado
tcp      0      0 14.200.8.3:38696          14.200.12.2:7        TIME_WAIT
tcp      0      0 14.200.8.3:50820          14.200.12.2:7        ESTABLECIDO
tcp      0      0 14.200.8.3:54512          14.200.12.2:9        TIME_WAIT
root@n13:/tmp/pycore.43805/n13.conf#
```

Se puede observar que las conexiones que se cerraron están en el estado TIME\_WAIT. Este estado indica que la sesión ha sido cerrada por n13, pero debe esperar por los paquetes que podría haber mandado n9 y aún no recibió.

Luego de un tiempo, se cierra la sesión entre el puerto 60820 de n13 y el puerto 7 del nodo n9. Al ejecutar el comando “netstat -atun”, se ve que esta conexión está en estado TIME\_WAIT, y que las conexiones anteriores ya no aparecen.

```
root@n13:/tmp/pycore.43805/n13.conf# netstat -atun
Conexiones activas de Internet (servidores y establecidos)
Proto Recib Enviad Dirección local           Dirección remota         Estado
tcp      0      0 14.200.8.3:60820          14.200.12.2:7        TIME_WAIT
root@n13:/tmp/pycore.43805/n13.conf#
```

Luego se cerraron las conexiones en n11 y n10. Al ejecutar el comando “netstat -atun” se observa que las sesiones también se encuentran en estado TIME\_WAIT.

```
root@n11:/tmp/pycore.43805/n11.conf# netstat -atun
Conexiones activas de Internet (servidores y establecidos)
Proto Recib Enviad Dirección local      Dirección remota      Estado
tcp      0      0 14.200.8.2:52826      14.200.12.2:7      TIME_WAIT
root@n11:/tmp/pycore.43805/n11.conf# []
```

```
root@n10:/tmp/pycore.43805/n10.conf# netstat -atun
Conexiones activas de Internet (servidores y establecidos)
Proto Recib Enviad Dirección local      Dirección remota      Estado
tcp      0      0 14.200.11.10:40294     14.200.12.2:7      TIME_WAIT
tcp      0      0 14.200.11.10:51452     14.200.12.2:9      TIME_WAIT
root@n10:/tmp/pycore.43805/n10.conf# []
```

Luego de realizar todos los cierres de conexión, al ejecutar “netstat -atun” en n9 inmediatamente se puede ver que no hay más conexiones pendientes.

```
root@n9:/tmp/pycore.43805/n9.conf# netstat -atun
Conexiones activas de Internet (servidores y establecidos)
Proto Recib Enviad Dirección local      Dirección remota      Estado
tcp      0      0 0.0.0.0:9          0.0.0.0:*
tcp      0      0 0.0.0.0:7          0.0.0.0:*
udp      0      0 0.0.0.0:7          0.0.0.0:*
udp      0      0 0.0.0.0:9          0.0.0.0:*
root@n9:/tmp/pycore.43805/n9.conf# []
```

**h) Observando la captura indicar la cantidad de segmentos y los flags utilizados. ¿Con cuántos segmentos se cerró la conexión? ¿Existen otras variantes de cierre?**

En la siguiente imagen pueden observar los segmentos enviados al cerrar dos sesiones TCP. En este caso ambas tienen como cliente al nodo n13 y al nodo n9 como servidor.

13 91.362512017 14.200.8.3	14.200.12.2	TCP	66 56376 → 7 [FIN, ACK] Seq=49 Ack=49 Win=64256 Len=0 TStamp=2687553519 TSectr=4044611202
14 91.362556188 14.200.8.3	14.200.12.2	TCP	66 35652 → 9 [FIN, ACK] Seq=44 Ack=1 Win=64256 Len=0 TStamp=2687553519 TSectr=4044611206
15 91.366864821 14.200.12.2	14.200.8.3	TCP	66 7 → 56376 [FIN, ACK] Seq=49 Ack=50 Win=65152 Len=0 TStamp=4044702557 TSectr=2687553519
16 91.366957738 14.200.8.3	14.200.12.2	TCP	66 56376 → 7 [ACK] Seq=50 Ack=50 Win=64256 Len=0 TStamp=2687553524 TSectr=4044702557
17 91.367424571 14.200.12.2	14.200.8.3	TCP	66 9 → 35652 [FIN, ACK] Seq=1 Ack=45 Win=65152 Len=0 TStamp=4044702558 TSectr=2687553519
18 91.367465980 14.200.8.3	14.200.12.2	TCP	66 35652 → 9 [ACK] Seq=45 Ack=2 Win=64256 Len=0 TStamp=2687553524 TSectr=4044702558

Para realizar el cierre de conexión se utilizaron 3 segmentos:

- el primero, enviado por el cliente con los flags FIN y ACK
- el segundo, enviado por el servidor también con las flags FIN y ACK
- el tercero, también enviado por el cliente únicamente con la flag ACK.

Los segmentos 13, 15 y 16 pertenecen al cierre de la conexión establecida con el puerto 7 de n9, mientras que los segmentos 14, 17 y 18 pertenecen a la conexión abierta en el puerto 9 del mismo nodo.

Existe otra variante de cierre, que hace uso de 4 segmentos: el llamado handshake de 4 pasos. En este caso, el cliente comienza enviando un segmento con flag FIN (el termination request), y espera por un segmento ACK. Cuando el servidor recibe el

segmento FIN, envía el ACK al cliente, confirmando que está listo para cerrar la sesión. Después, envía un segmento FIN indicando que se ha aprobado el cierre de la sesión. Cuando el cliente recibe el FIN, envía un ACK, y en cuanto el servidor lo recibe la sesión es finalizada.

En este último tipo de cierre, el nodo que no haya iniciado el cierre puede seguir enviando segmentos hasta que decida enviar el segundo segmento con el flag FIN. Por otro lado, el nodo que inició el cierre solo puede mandar segmentos ACK luego de enviar el primer segmento con el flag FIN.

**i) Hacer un diagrama de los segmentos intercambiados con los números de secuencia absolutos para una de las sesiones TCP (Se puede usar la herramienta wireshark u otra).**

En la siguiente imagen se puede observar el flujo de los segmentos intercambiados en la comunicación TCP establecida entre n13 (cliente) y n9 (servidor), con su inicio de sesión, el envío de la frase “Hola desde n13” y la finalización de la sesión.

Por defecto, la herramienta Wireshark muestra el número de secuencia relativo de los segmentos (es decir, desde que se establece la sesión y se está escuchando). Sin embargo, es posible cambiar la configuración ingresando a “edit” > “preferences” > “protocols” > “TCP” y desmarcando el ítem “relative sequence numbers”. Entonces, se pasarán a mostrar los números de secuencia absolutos.

39422	39422 → 9000 [SYN] Seq=3155597995 Win=64240 Len=0 M... 9000
39422	9000 → 39422 [SYN, ACK] Seq=2033006282 Ack=315559799... 9000
39422	39422 → 9000 [ACK] Seq=3155597996 Ack=2033006283 Wi... 9000
39422	39422 → 9000 [PSH, ACK] Seq=3155597996 Ack=203300628... 9000
39422	9000 → 39422 [ACK] Seq=2033006283 Ack=3155598011 Wi... 9000
39422	39422 → 9000 [FIN, ACK] Seq=3155598011 Ack=203300628... 9000
39422	9000 → 39422 [FIN, ACK] Seq=2033006283 Ack=315559801... 9000
39422	39422 → 9000 [ACK] Seq=3155598012 Ack=2033006284 Wi... 9000

La imagen muestra un análisis detallado del intercambio de segmentos TCP entre los hosts n13 (cliente) y n9 (servidor).

**Establecimiento de conexión:**

El cliente (puerto 39422) envía un segmento SYN al servidor (puerto 9000) para iniciar la conexión, con un número de secuencia inicial (Seq=3155597995).

El servidor responde con un segmento SYN, ACK confirmado la solicitud, utilizando también un número de secuencia inicial (Seq=2033006282) y un reconocimiento del segmento previo (Ack=3155597996).

El cliente confirma el establecimiento de la conexión con un segmento ACK=2033006283.

#### Transferencia de datos:

El cliente envía un segmento PSH, ACK con datos (longitud = 15) al servidor, avanzando su número de secuencia (Seq=3155597996).

El servidor responde con un segmento ACK= 315559798011, confirmado la recepción de los datos.

#### Finalización de la conexión:

El cliente inicia la finalización de la conexión enviando un segmento FIN, ACK, con el número de secuencia en Seq=315559798011.

El servidor confirma con un segmento FIN, ACK, avanzando su número de reconocimiento (Ack=315559798012).

Finalmente, el cliente cierra la sesión con un segmento ACK. Seq= 315559798012.

**j) Alternativo: Realizar una conexión mediante nc indicando un puerto específico para el cliente. Luego cerrar la conexión desde el cliente e intentar abrirla nuevamente. ¿En qué estado está el socket? Investigar el valor del 2MSL en la plataforma sobre la cual está haciendo los tests.**

Se comenzó configurando a n14 para escuchar en el puerto 9001 y conectando n10 a ese puerto desde su puerto 50000. Se comprobó desde n14 que la conexión fue establecida mediante el comando netstat y se envió un mensaje desde n10, que fue recibido correctamente por n14.

```
root@n14:/tmp/pycore.39591/n14.conf# nc -l -p 9001
holo n14
^C
root@n10:/tmp/pycore.39591/n10.conf# nc -p 50000 14.200.14.2 9001
holo n14
^C
root@n14:/tmp/pycore.39591/n14.conf# netstat -atun
Conexiones activas de Internet (servidores y establecidos)
Proto Recib Enviad Dirección local          Dirección remota      Estado
tcp      0      0 0.0.0.0:9001            0.0.0.0:*
tcp      0      0 14.200.14.2:9001        14.200.11.10:50000    ESTABLECIDO
```

Luego se cerró la conexión desde n10 con las teclas ctrl+c e inmediatamente se intentó volver a abrirla. Esto no fue posible, ya que el puerto 50000 de n10 ya estaba en uso. Al verificar el estado de las conexiones con el comando netstat, se observó que el mismo era TIME\_WAIT.

```
^C
root@n10:/tmp/pycore.39591/n10.conf# nc -p 50000 14.200.14.2 9001
nc: bind failed: Address already in use
root@n10:/tmp/pycore.39591/n10.conf# netstat -atun
Conexiones activas de Internet (servidores y establecidos)
Proto Recib Enviad Dirección local           Dirección remota      Estado
tcp      0      0 14.200.11.10:50000        14.200.14.2:9001      TIME_WAIT
root@n10:/tmp/pycore.39591/n10.conf#
```

El estado TIME\_WAIT ocurre para asegurar que cualquier segmento retrasado en la red sea descartado antes de reutilizar el socket.

El servidor espera un tiempo (2MSL) antes de liberar completamente los recursos asociados con el socket, el cual evita problemas como la recepción de paquetes viejos en una nueva conexión. El socket se cierra completamente después de que el período TIME\_WAIT expira.

El 2MSL (Maximum Segment Lifetime) es el tiempo que un paquete TCP puede permanecer en la red antes de descartarse. Es el tiempo que el socket permanece en TIME\_WAIT.

Mediante el comando “cat /proc/sys/net/ipv4/tcp\_fin\_timeout” se comprobó que el 2MSL en este caso es 60 (segundos).