



# **UNIVERSIDAD NACIONAL DE CÓRDOBA**

**FACULTAD DE CIENCIAS EXACTAS, FÍSICAS, QUÍMICAS Y  
NATURALES**

Redes de Computadoras

**TRABAJO PRÁCTICO N° 1 : Grupo N° 13**

## **Análisis de tráfico IPv6 en capa 3**

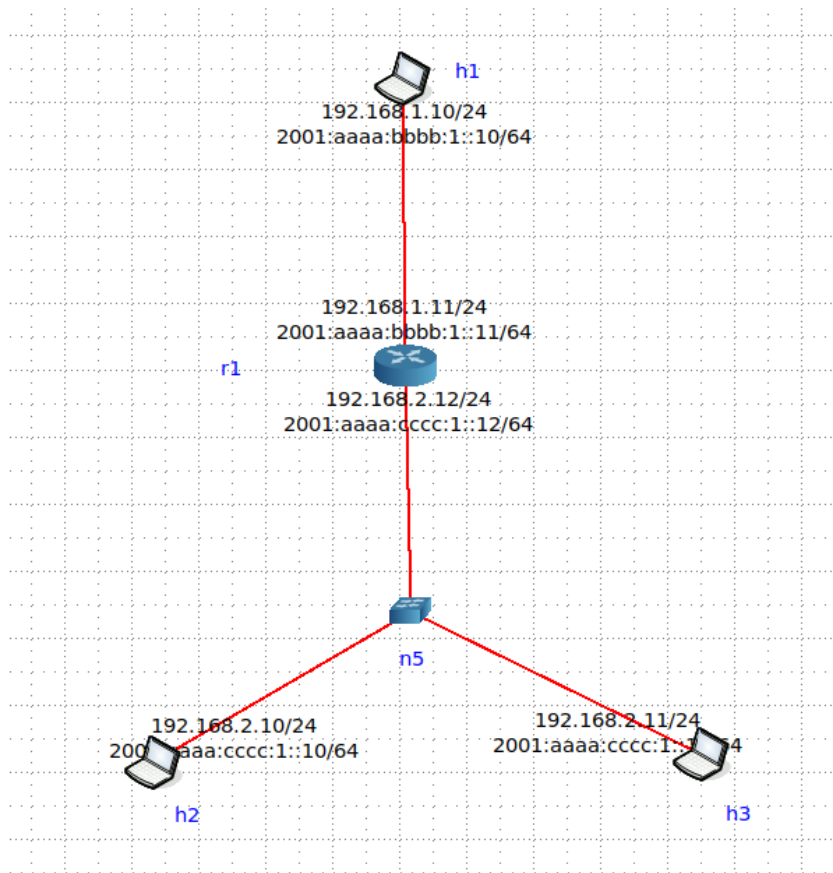
- Cancinos, Jose
- Oliva, Nahuel

**Fecha:** 02/04/2020

# DESARROLLO

## A ) Tráfico IPv4 e IPv6 con CORE

1) Creación de esquema de red en CORE.



- ¿Cual es la diferencia entre un simulador y un emulador?

Simulador	Emulador
Define un espacio definido por software (environment)	Define un espacio definido por software (environment)
Diseñado para crear un escenario que contiene todas las variables y configuraciones de software que pueden existir en el escenario real de la aplicación	Diseñado para imitar tanto las variables hardware como software.
No imitan el hardware. Solamente define escenarios software	
	Están entre medio de simuladores y

	dispositivos reales
Asegura que la aplicación cumple con performance esperada al interactuar con aplicaciones externas o escenarios	Prueba como el software interactúa con el hardware subyacente o con una combinación entre hardware/software
Permiten construir y evaluar modelos experimentales	Permiten evaluar el comportamiento real

- ¿Por que CORE es considerado un emulador ?

Es considerado un emulador ya que me permite realizar operaciones y obtener funcionalidades que son posibles de realizar en una red verdadera (definición de emulador) y no sólo demostrar el comportamiento de la red.

Permite replicar una conexión cliente/servidor sin la necesidad de dispositivos reales como routers así como también replicar tráfico de red.

- ¿Conoce algún simulador en el área de redes ?

SI, GNS3 o Cisco Packet Tracer.

## 2) Conectividad entre hosts usando ICMPv4

- Utilizo la interfaz gráfica para setear elementos y enlaces.
- Asigno direcciones estáticas para IPv4 mediante interfaz gráfica
- Configurar tablas de ruteo ipv4 en cada host.(default gateway)

Comando: *ip route add <DIRECCION DE SUBRED/MASCARA> via <DIRECCION DE SALIDA>*

Interfaces Configuradas :

```

> ip -4 addr
> ip -4 addr
> ip -4 addr
> ip -4 addr
> n1 > ip -4 addr:
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
t qlen 1000
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
21: eth0@if22: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
group default qlen 1000 link-netnsid 0
    inet 192.168.1.10/24 scope global eth0
        valid_lft forever preferred_lft forever
> n2 > ip -4 addr:
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
t qlen 1000
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
25: eth0@if26: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
group default qlen 1000 link-netnsid 0
    inet 192.168.2.10/24 scope global eth0
        valid_lft forever preferred_lft forever
> n3 > ip -4 addr:
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
t qlen 1000
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
27: eth0@if28: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
group default qlen 1000 link-netnsid 0
    inet 192.168.2.11/24 scope global eth0
        valid_lft forever preferred_lft forever
> n4 > ip -4 addr:
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
t qlen 1000
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
19: eth0@if20: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
group default qlen 1000 link-netnsid 0
    inet 192.168.1.11/24 scope global eth0
        valid_lft forever preferred_lft forever
23: eth1@if24: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
group default qlen 1000 link-netnsid 0
    inet 192.168.2.12/24 scope global eth1
        valid_lft forever preferred_lft forever

```

```
root@h1: /tmp/pycore.36743/h1.conf
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
root@h1:/tmp/pycore.36743/h1.conf# route -n
Tabla de rutas IP del núcleo
Destino      Pasarela      Genmask      Indic Métric Ref      Uso Interfaz
0.0.0.0      192.168.1.11  0.0.0.0      UG    0      0      0 eth0
192.168.1.0  0.0.0.0      255.255.255.0 U    0      0      0 eth0
root@h1:/tmp/pycore.36743/h1.conf#

root@h2: /tmp/pycore.36743/h2.conf
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
root@h2:/tmp/pycore.36743/h2.conf# route -n
Tabla de rutas IP del núcleo
Destino      Pasarela      Genmask      Indic Métric Ref      Uso Interfaz
0.0.0.0      192.168.2.12  0.0.0.0      UG    0      0      0 eth0
192.168.2.0  0.0.0.0      255.255.255.0 U    0      0      0 eth0
root@h2:/tmp/pycore.36743/h2.conf#

root@h3: /tmp/pycore.36743/h3.conf
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
root@h3:/tmp/pycore.36743/h3.conf# route -n
Tabla de rutas IP del núcleo
Destino      Pasarela      Genmask      Indic Métric Ref      Uso Interfaz
0.0.0.0      192.168.2.12  0.0.0.0      UG    0      0      0 eth0
192.168.2.0  0.0.0.0      255.255.255.0 U    0      0      0 eth0
root@h3:/tmp/pycore.36743/h3.conf#

root@r1: /tmp/pycore.36743/r1.conf
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
root@r1:/tmp/pycore.36743/r1.conf# ip route
192.168.1.0/24 dev eth1 proto kernel scope link src 192.168.1.11
192.168.2.0/24 dev eth0 proto kernel scope link src 192.168.2.12
root@r1:/tmp/pycore.36743/r1.conf#
```

Envío ping de prueba utilizando el comando:

```
ping -4 -c 3 <DESTINO>
```

```
root@n1:/tmp/pycore.33519/n1.conf# ping -4 -c 3 192.168.2.10
PING 192.168.2.10 (192.168.2.10) 56(84) bytes of data.
64 bytes from 192.168.2.10: icmp_seq=1 ttl=63 time=0.501 ms
64 bytes from 192.168.2.10: icmp_seq=2 ttl=63 time=0.098 ms
64 bytes from 192.168.2.10: icmp_seq=3 ttl=63 time=0.097 ms

--- 192.168.2.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2035ms
rtt min/avg/max/mdev = 0.097/0.232/0.501/0.190 ms
root@n1:/tmp/pycore.33519/n1.conf# ping -4 -c 3 192.168.2.11
PING 192.168.2.11 (192.168.2.11) 56(84) bytes of data.
64 bytes from 192.168.2.11: icmp_seq=1 ttl=63 time=0.179 ms
64 bytes from 192.168.2.11: icmp_seq=2 ttl=63 time=0.091 ms
64 bytes from 192.168.2.11: icmp_seq=3 ttl=63 time=0.095 ms

--- 192.168.2.11 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2046ms
rtt min/avg/max/mdev = 0.091/0.121/0.179/0.042 ms
root@n1:/tmp/pycore.33519/n1.conf#
```

```
root@n2:/tmp/pycore.33519/n2.conf# ping -4 -c 3 192.168.2.11
PING 192.168.2.11 (192.168.2.11) 56(84) bytes of data.
64 bytes from 192.168.2.11: icmp_seq=1 ttl=64 time=0.058 ms
64 bytes from 192.168.2.11: icmp_seq=2 ttl=64 time=0.099 ms
64 bytes from 192.168.2.11: icmp_seq=3 ttl=64 time=0.098 ms

--- 192.168.2.11 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2042ms
rtt min/avg/max/mdev = 0.058/0.085/0.099/0.019 ms
root@n2:/tmp/pycore.33519/n2.conf# ping -4 -c 3 192.168.1.10
PING 192.168.1.10 (192.168.1.10) 56(84) bytes of data.
64 bytes from 192.168.1.10: icmp_seq=1 ttl=63 time=0.098 ms
64 bytes from 192.168.1.10: icmp_seq=2 ttl=63 time=0.121 ms
64 bytes from 192.168.1.10: icmp_seq=3 ttl=63 time=0.119 ms

--- 192.168.1.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2052ms
rtt min/avg/max/mdev = 0.098/0.112/0.121/0.016 ms
root@n2:/tmp/pycore.33519/n2.conf#
```



```

root@n3:/tmp/pycore.33519/n3.conf# ping -4 -c 3 192.168.2.10
PING 192.168.2.10 (192.168.2.10) 56(84) bytes of data.
64 bytes from 192.168.2.10: icmp_seq=1 ttl=64 time=0.224 ms
64 bytes from 192.168.2.10: icmp_seq=2 ttl=64 time=0.099 ms
64 bytes from 192.168.2.10: icmp_seq=3 ttl=64 time=0.028 ms

--- 192.168.2.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2047ms
rtt min/avg/max/mdev = 0.028/0.117/0.224/0.081 ms
root@n3:/tmp/pycore.33519/n3.conf# ping -4 -c 3 192.168.1.10
PING 192.168.1.10 (192.168.1.10) 56(84) bytes of data.
64 bytes from 192.168.1.10: icmp_seq=1 ttl=63 time=0.124 ms
64 bytes from 192.168.1.10: icmp_seq=2 ttl=63 time=0.117 ms
64 bytes from 192.168.1.10: icmp_seq=3 ttl=63 time=0.118 ms

--- 192.168.1.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2026ms
rtt min/avg/max/mdev = 0.117/0.119/0.124/0.013 ms
root@n3:/tmp/pycore.33519/n3.conf# 

```

### 3) Conectividad entre hosts usando ICMPv6

```

root@h1:/tmp/pycore.39277/h1.conf# route -6 -n
Tabla de ruteado IPv6 del núcleo

```

Destination	Next Hop	Flag	Met	Ref	Use	If	IP6 Table
2001:aaaa:bbbb:1::/64	::	U	256	3	0	eth0	2001:aaaa:cccc:1::10
fe80::/64	::	U	256	1	0	eth0	2001:aaaa:cccc:1::11
::/0	2001:aaaa:bbbb:1::11	UG	1024	1	0	eth0	
::1/128	::	Un	0	3	0	lo	
2001:aaaa:bbbb:1::10/128	::	Un	0	2	0	eth0	
fe80::200:ff:feaa:4/128	::	Un	0	2	0	eth0	
ff00::/8	::	U	256	6	0	eth0	
::/0	::	In	-1	1	0	lo	

```

root@h1:/tmp/pycore.39277/h1.conf# 

```

Envío ping para IPv6, utilizando el siguiente comando:

**`ping -6 -c <CANT PAQUETES> <DIRECCIÓN DE DESTINO IPv6>`**

```

root@n1:/tmp/pycore.33519/n1.conf# ping -6 -c 3 2001:aaaa:cccc:1::10
PING 2001:aaaa:cccc:1::10(2001:aaaa:cccc:1::10) 56 data bytes
64 bytes from 2001:aaaa:cccc:1::10: icmp_seq=1 ttl=63 time=0.124 ms
64 bytes from 2001:aaaa:cccc:1::10: icmp_seq=2 ttl=63 time=0.116 ms
64 bytes from 2001:aaaa:cccc:1::10: icmp_seq=3 ttl=63 time=0.166 ms

--- 2001:aaaa:cccc:1::10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2024ms
rtt min/avg/max/mdev = 0.116/0.135/0.166/0.023 ms
root@n1:/tmp/pycore.33519/n1.conf# ping -6 -c 3 2001:aaaa:cccc:1::11
PING 2001:aaaa:cccc:1::11(2001:aaaa:cccc:1::11) 56 data bytes
64 bytes from 2001:aaaa:cccc:1::11: icmp_seq=1 ttl=63 time=0.053 ms
64 bytes from 2001:aaaa:cccc:1::11: icmp_seq=2 ttl=63 time=0.283 ms
64 bytes from 2001:aaaa:cccc:1::11: icmp_seq=3 ttl=63 time=0.114 ms

--- 2001:aaaa:cccc:1::11 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2031ms
rtt min/avg/max/mdev = 0.053/0.150/0.283/0.097 ms
root@n1:/tmp/pycore.33519/n1.conf# 

```

#### 4) Análisis de tráfico en IPv4

A continuación se puede observar como se traducen las direcciones Ip configuradas estáticamente para conseguir las direcciones MAC al relizar el ping entre el host h1 a h3.

```
root@r1:/tmp/pycore.39277/r1.conf
Archivo Editar Ver Buscar Terminal Ayuda
root@r1:/tmp/pycore.39277/r1.conf# arp -n
root@r1:/tmp/pycore.39277/r1.conf#
```

```
root@h1:/tmp/pycore.39277/h1.conf#
root@h1:/tmp/pycore.39277/h1.conf# arp -a
root@h1:/tmp/pycore.39277/h1.conf# tcpdump -i eth0 -v
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C11:19:20.735797 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has _gateway tel
l h1, length 28
11:19:20.735878 ARP, Ethernet (len 6), IPv4 (len 4), Reply _gateway is-at 00:00:00:
aa:00:03 (oui Ethernet), length 28
11:19:20.735889 IP (tos 0x0, ttl 64, id 38076, offset 0, flags [DF], proto ICMP (1)
, length 84)
    h1 > 192.168.2.11: ICMP echo request, id 37, seq 1, length 64
11:19:20.736044 IP (tos 0x0, ttl 63, id 53232, offset 0, flags [none], proto ICMP (
1), length 84)
    192.168.2.11 > h1: ICMP echo reply, id 37, seq 1, length 64
11:19:21.737057 IP (tos 0x0, ttl 64, id 38225, offset 0, flags [DF], proto ICMP (1)
, length 84)
    h1 > 192.168.2.11: ICMP echo request, id 37, seq 2, length 64
11:19:21.737184 IP (tos 0x0, ttl 63, id 53392, offset 0, flags [none], proto ICMP (
1), length 84)
    192.168.2.11 > h1: ICMP echo reply, id 37, seq 2, length 64
11:19:22.743573 IP (tos 0x0, ttl 64, id 38264, offset 0, flags [DF], proto ICMP (1)
, length 84)
    h1 > 192.168.2.11: ICMP echo request, id 37, seq 3, length 64
11:19:22.743701 IP (tos 0x0, ttl 63, id 53596, offset 0, flags [none], proto ICMP (
1), length 84)
    192.168.2.11 > h1: ICMP echo reply, id 37, seq 3, length 64
11:19:23.895484 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 16) fe80::70
a8:1eff:fe9e:ec52 > ip6-allrouters: [icmp6 sum ok] ICMP6, router solicitation, leng
th 16
    source link-address option (1), length 8 (1): 72:a8:1e:9e:ec:52
11:19:23.895484 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 16) fe80::28
f3:4aff:feeb:35be > ip6-allrouters: [icmp6 sum ok] ICMP6, router solicitation, leng
th 16
    source link-address option (1), length 8 (1): 72:a8:1e:9e:ec:52
10 packets captured
10 packets received by filter
0 packets dropped by kernel
root@h1:/tmp/pycore.39277/h1.conf#

root@h3:/tmp/pycore.39277/h3.conf# tcpdump -i eth0 -v
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C11:19:20.735923 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has h3 tell _gat
eway, length 28
11:19:20.735964 ARP, Ethernet (len 6), IPv4 (len 4), Reply h3 is-at 00:00:00:aa:00:
00 (oui Ethernet), length 28
11:19:20.735988 IP (tos 0x0, ttl 63, id 38076, offset 0, flags [DF], proto ICMP (1)
, length 84)
    192.168.1.10 > h3: ICMP echo request, id 37, seq 1, length 64
11:19:20.736021 IP (tos 0x0, ttl 64, id 53232, offset 0, flags [none], proto ICMP (
1), length 84)
    h3 > 192.168.1.10: ICMP echo reply, id 37, seq 1, length 64
11:19:21.591345 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 16) h3 > ip6
-allrouters: [icmp6 sum ok] ICMP6, router solicitation, length 16
    source link-address option (1), length 8 (1): 00:00:00:aa:00:00
11:19:21.737119 IP (tos 0x0, ttl 63, id 38225, offset 0, flags [DF], proto ICMP (1)
, length 84)
    192.168.1.10 > h3: ICMP echo request, id 37, seq 2, length 64
11:19:21.737160 IP (tos 0x0, ttl 64, id 53392, offset 0, flags [none], proto ICMP (
1), length 84)
    h3 > 192.168.1.10: ICMP echo reply, id 37, seq 2, length 64
11:19:22.743636 IP (tos 0x0, ttl 63, id 38264, offset 0, flags [DF], proto ICMP (1)
, length 84)
    192.168.1.10 > h3: ICMP echo request, id 37, seq 3, length 64
11:19:22.743677 IP (tos 0x0, ttl 64, id 53596, offset 0, flags [none], proto ICMP (
1), length 84)
    h3 > 192.168.1.10: ICMP echo reply, id 37, seq 3, length 64
11:19:22.871590 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 16) fe80::20
0:ff:feaa:1 > ip6-allrouters: [icmp6 sum ok] ICMP6, router solicitation, length 16
    source link-address option (1), length 8 (1): 00:00:00:aa:00:01
11:19:22.871604 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 16) fe80::3c
1b:c6ff:fe01:4021 > ip6-allrouters: [icmp6 sum ok] ICMP6, router solicitation, leng
th 16
    source link-address option (1), length 8 (1): 42:bc:2d:90:8f:fe
11 packets captured
11 packets received by filter
0 packets dropped by kernel
root@h3:/tmp/pycore.39277/h3.conf#
```

```
root@r1:/tmp/pycore.39277/r1.conf
Archivo Editar Ver Buscar Terminal Ayuda
root@r1:/tmp/pycore.39277/r1.conf# arp -n
Dirección      TipoHW  DirecciónHW      Indic Máscara      Inter
faz
192.168.2.11    ether   00:00:00:aa:00:00 C                    eth0
192.168.1.10    ether   00:00:00:aa:00:04 C                    eth1
root@r1:/tmp/pycore.39277/r1.conf#
```



```
root@h1: /tmp/pycore.39277/h1.conf
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
root@h1: /tmp/pycore.39277/h1.conf# arp -n
Dirección      TipoHW  DirecciónHW  Indic  Máscara  Inter
faz
192.168.1.11    ether   00:00:00:aa:00:03  C      eth0
root@h1: /tmp/pycore.39277/h1.conf#
```

## ICMPv3

- a. ¿Cuáles son las comunicaciones ARP que suceden? Ejemplifica brevemente y con capturas cómo funciona la traducción de direcciones lógicas a direcciones físicas.

Recordar:

- Dirección lógica: dirección IP.
- Dirección física: dirección de MAC. Es única en todo el mundo y está asociada a la NIC (Network Interface Card)

Por lo tanto, para comunicarnos entre redes separadas debemos pasar de nuestra dirección lógica usada en la red local y utilizar la dirección física. Se usa ARP. Teniendo la dirección de capa 3 (IP) podemos obtener la dirección de capa 1 (MAC) para iniciar una comunicación entre dispositivos de redes no locales.

Las comunicaciones ARP que suceden son mensajes de tipo:

- Petición (Request)
- Respuesta (Reply)

1) El host que envía el ping (host **n1**) quiere enviar el mensaje hacia el host 2. Sabe que está fuera de su red y para comunicarse con la red 192.168.2.0/24 debe enviar el mensaje a 192.168.1.12 (router) pero no conoce la dirección MAC para enviar ya que su tabla ARP está vacía. Por lo tanto, envía un mensaje peticionando a éste por su MAC así como también envía su propia dirección MAC, 00:00:00:aa:00:04

2) El router devuelve un mensaje ARP indicando al host n1 que el dispositivo con la dirección IP de próximo salto del router conectado a su red tiene como dirección MAC 00:00:00:aa:00:03.

3) Luego, h1 recibe el mensaje ARP con la dirección MAC del próximo salto y la agrega a su tabla, para luego poder así enviar el mensaje icmp al mismo.

En la otra red sucede lo siguiente:

4) El router recibe el mensaje icmp cuyo destino es el host 2 de la red contigua, que al coincidir la dirección MAC del paquete con la suya, la elimina, para luego actualizarla. Para ello, debe enviar el mensaje al enlace del router con la red 192.168.2.0/24. Como su tabla ARP sólo contiene la MAC del host 1, peticiona la MAC del host 2 teniendo en cuenta la dirección del destino del paquete IP, 192.168.2.10 (n2). Recibe un mensaje de respuesta con la MAC de host 2 y guarda en su tabla ARP.

5) El mensaje IP llega a host 2. Debe retornar el mensaje ping de respuesta a host 1, y

como su tabla ARP está vacía, peticiona al router por la dirección MAC del enlace de salida (192.168.2.12). Recibe luego, por parte del router, un mensaje ARP de respuesta con la dirección MAC del enlace. Actualiza la tabla y puede finalmente enviar el ping de respuesta.

```

root@h1:/tmp/pycore.39277/h1.conf#
root@h1:/tmp/pycore.39277/h1.conf# arp -a
root@h1:/tmp/pycore.39277/h1.conf# tcpdump -i eth0 -v
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C11:19:20.735797 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has _gateway tel
l h1, length 28
11:19:20.735878 ARP, Ethernet (len 6), IPv4 (len 4), Reply _gateway is-at 00:00:00:
aa:00:03 (oui Ethernet), length 28
11:19:20.735889 IP (tos 0x0, ttl 64, id 38076, offset 0, flags [DF], proto ICMP (1)
, length 84)
    h1 > 192.168.2.11: ICMP echo request, id 37, seq 1, length 64
11:19:20.736044 IP (tos 0x0, ttl 63, id 53232, offset 0, flags [none], proto ICMP (
1), length 84)
    192.168.2.11 > h1: ICMP echo reply, id 37, seq 1, length 64
11:19:21.737057 IP (tos 0x0, ttl 64, id 38225, offset 0, flags [DF], proto ICMP (1)
, length 84)
    h1 > 192.168.2.11: ICMP echo request, id 37, seq 2, length 64
11:19:21.737184 IP (tos 0x0, ttl 63, id 53392, offset 0, flags [none], proto ICMP (
1), length 84)
    192.168.2.11 > h1: ICMP echo reply, id 37, seq 2, length 64
11:19:22.743573 IP (tos 0x0, ttl 64, id 38264, offset 0, flags [DF], proto ICMP (1)
, length 84)
    h1 > 192.168.2.11: ICMP echo request, id 37, seq 3, length 64
11:19:22.743701 IP (tos 0x0, ttl 63, id 53596, offset 0, flags [none], proto ICMP (
1), length 84)
    192.168.2.11 > h1: ICMP echo reply, id 37, seq 3, length 64
11:19:23.387455 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 16) fe80::70
a8:1eff:fe9e:ec52 > ip6-allrouters: [icmp6 sum ok] ICMPv6, router solicitation, leng
th 16
    source link-address option (1), length 8 (1): 72:a8:1e:9e:ec:52
11:19:23.895484 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 16) fe80::28
f3:4aff:feeb:35be > ip6-allrouters: [icmp6 sum ok] ICMPv6, router solicitation, leng
th 16
    source link-address option (1), length 8 (1): 72:a8:1e:9e:ec:52

10 packets captured
10 packets received by filter
0 packets dropped by kernel
root@h1:/tmp/pycore.39277/h1.conf#

```

```

root@h3:/tmp/pycore.39277/h3.conf# tcpdump -i eth0 -v
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C11:19:20.735923 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has h3 tell _gat
eway, length 28
11:19:20.735964 ARP, Ethernet (len 6), IPv4 (len 4), Reply h3 is-at 00:00:00:aa:00:
00 (oui Ethernet), length 28
11:19:20.735988 IP (tos 0x0, ttl 63, id 38076, offset 0, flags [DF], proto ICMP (1)
, length 84)
    192.168.1.10 > h3: ICMP echo request, id 37, seq 1, length 64
11:19:20.736021 IP (tos 0x0, ttl 64, id 53232, offset 0, flags [none], proto ICMP (
1), length 84)
    h3 > 192.168.1.10: ICMP echo reply, id 37, seq 1, length 64
11:19:21.591345 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 16) h3 > ip6
-allrouters: [icmp6 sum ok] ICMPv6, router solicitation, length 16
    source link-address option (1), length 8 (1): 00:00:00:aa:00:00
11:19:21.737119 IP (tos 0x0, ttl 63, id 38225, offset 0, flags [DF], proto ICMP (1)
, length 84)
    192.168.1.10 > h3: ICMP echo request, id 37, seq 2, length 64
11:19:21.737160 IP (tos 0x0, ttl 64, id 53392, offset 0, flags [none], proto ICMP (
1), length 84)
    h3 > 192.168.1.10: ICMP echo reply, id 37, seq 2, length 64
11:19:22.743636 IP (tos 0x0, ttl 63, id 38264, offset 0, flags [DF], proto ICMP (1)
, length 84)
    192.168.1.10 > h3: ICMP echo request, id 37, seq 3, length 64
11:19:22.743677 IP (tos 0x0, ttl 64, id 53596, offset 0, flags [none], proto ICMP (
1), length 84)
    h3 > 192.168.1.10: ICMP echo reply, id 37, seq 3, length 64
11:19:22.871590 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 16) fe80::20
0f:feaa:1 > ip6-allrouters: [icmp6 sum ok] ICMPv6, router solicitation, length 16
    source link-address option (1), length 8 (1): 00:00:00:aa:00:01
11:19:22.871604 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 16) fe80::3c
1b:c6ff:fe01:4021 > ip6-allrouters: [icmp6 sum ok] ICMPv6, router solicitation, leng
th 16
    source link-address option (1), length 8 (1): 42:bc:2d:90:8f:fe

11 packets captured
11 packets received by filter
0 packets dropped by kernel
root@h3:/tmp/pycore.39277/h3.conf#

```

## b. ¿Cuáles son las direcciones IPs en los datagramas IPs?

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.1.10	192.168.2.10	ICMP	98	Echo (ping) request id=0x0031, seq=1
2	0.000037590	192.168.2.10	192.168.1.10	ICMP	98	Echo (ping) reply id=0x0031, seq=1
3	1.015775226	192.168.1.10	192.168.2.10	ICMP	98	Echo (ping) request id=0x0031, seq=2
4	1.015801039	192.168.2.10	192.168.1.10	ICMP	98	Echo (ping) reply id=0x0031, seq=2
5	2.039775491	192.168.1.10	192.168.2.10	ICMP	98	Echo (ping) request id=0x0031, seq=3
6	2.039801227	192.168.2.10	192.168.1.10	ICMP	98	Echo (ping) reply id=0x0031, seq=3

<p>Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0</p> <p>Ethernet II, Src: 00:00:00:aa:00:02 (00:00:00:aa:00:02), Dst: 00:00:00:aa:00:01 (00:00:00:aa:00:01)</p> <p>Internet Protocol Version 4, Src: 192.168.1.10, Dst: 192.168.2.10</p> <p>0100 .... = Version: 4</p> <p>.... 0101 = Header Length: 20 bytes (5)</p> <p>Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)</p> <p>Total Length: 84</p> <p>Identification: 0xe361 (58209)</p> <p>Flags: 0x4000, Don't fragment</p> <p>Time to live: 63</p> <p>Protocol: ICMP (1)</p> <p>Header checksum: 0xd3e2 [validation disabled]</p> <p>[Header checksum status: Unverified]</p> <p>Source: 192.168.1.10</p> <p>Destination: 192.168.2.10</p> <p>Internet Control Message Protocol</p>						
--	--	--	--	--	--	--

Direcciones:

Source address: dirección IPv4 del host que envía el datagrama.

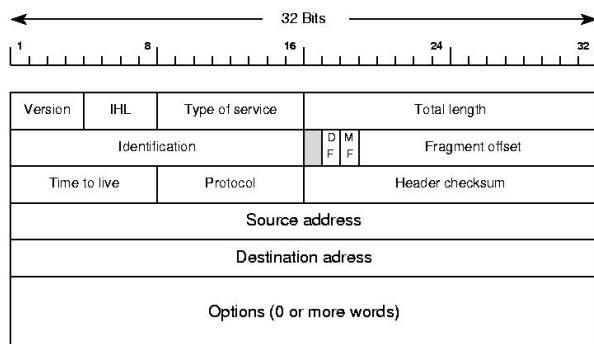
Destination address: dirección IPv4 del host al que está destinado el datagrama.

```

root@h1:/tmp/pycore.39277/h1.conf#
root@h1:/tmp/pycore.39277/h1.conf# tcpdump -i eth0 -v
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C11:39:58.845087 IP (tos 0x0, ttl 64, id 24274, offset 0, flags [DF], proto ICMP (1), length 84)
  h1 > 192.168.2.11: ICMP echo request, id 38, seq 1, length 64
11:39:58.845189 IP (tos 0x0, ttl 63, id 39960, offset 0, flags [none], proto ICMP (1), length 84)
  192.168.2.11 > h1: ICMP echo reply, id 38, seq 1, length 64
11:39:59.846406 IP (tos 0x0, ttl 64, id 24481, offset 0, flags [DF], proto ICMP (1), length 84)
  h1 > 192.168.2.11: ICMP echo request, id 38, seq 2, length 64
11:39:59.846504 IP (tos 0x0, ttl 63, id 39968, offset 0, flags [none], proto ICMP (1), length 84)
  192.168.2.11 > h1: ICMP echo reply, id 38, seq 2, length 64
11:40:00.855572 IP (tos 0x0, ttl 64, id 24491, offset 0, flags [DF], proto ICMP (1), length 84)
  h1 > 192.168.2.11: ICMP echo request, id 38, seq 3, length 64
11:40:00.855672 IP (tos 0x0, ttl 63, id 40065, offset 0, flags [none], proto ICMP (1), length 84)
  192.168.2.11 > h1: ICMP echo reply, id 38, seq 3, length 64

6 packets captured
6 packets received by filter
0 packets dropped by kernel
root@h1:/tmp/pycore.39277/h1.conf#

```



c. ¿Cómo sabe el router como comunicar un host con otro host?

El router sabe como comunicar hosts de dos redes distintas debido a que contiene en su tabla de ruteo las indicaciones sobre cual interfaz usar para comunicar mensajes que se transmiten de una red a otra.

```

File Edit View Search Terminal Help
root@r1:/tmp/pycore.36579/r1.conf# route
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0      0.0.0.0        255.255.255.0   U        0      0        0 eth1
192.168.2.0      0.0.0.0        255.255.255.0   U        0      0        0 eth0
root@r1:/tmp/pycore.36579/r1.conf#

```

d. ¿Para qué usamos el switch? ¿Por que el switch no tiene asignadas direcciones IP en sus interfaces?

Utilizamos un switch para dar comunicación los distintos dispositivos que forman una red local. Permite transportar paquetes a uno o más de uno de los dispositivos a los que está conectado.

El switch no tiene asignada direcciones IP debido a que opera en capa 2 (capa de enlace) y utiliza direcciones MAC para poder realizar el direccionamiento de los datos. Para ello, el switch busca coincidencias en su tabla de direcciones MAC con la dirección de destino que el paquete recibido contiene.

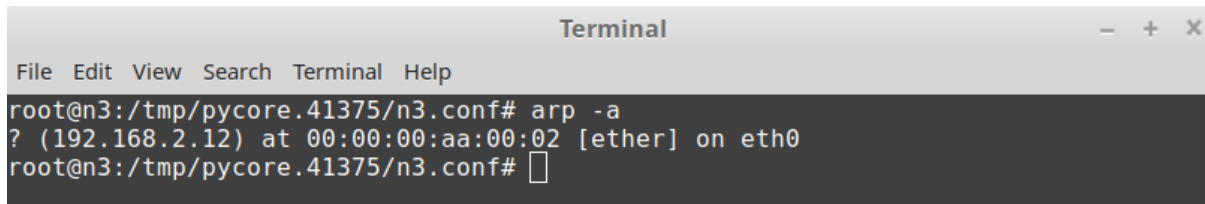
e. ¿Qué datos contiene la tabla ARP de h1?

Tabla ARP en host 1

```
0 packets dropped by kernel
root@n1:/tmp/pycore.41375/n1.conf# arp -a
? (192.168.1.11) at 00:00:00:aa:00:03 [ether] on eth0
root@n1:/tmp/pycore.41375/n1.conf#
```

f. ¿Qué datos contiene la tabla ARP de h3?

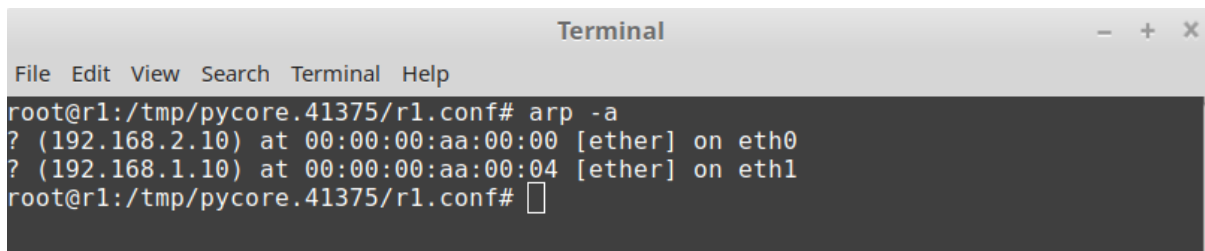
Tabla ARP en host 3

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Search, Terminal, Help). The command prompt is root@n3:/tmp/pycore.41375/n3.conf#. The command 'arp -a' has been executed, showing the ARP table entry for (192.168.2.12) at MAC address 00:00:00:aa:00:02 on interface eth0.

```
Terminal
File Edit View Search Terminal Help
root@n3:/tmp/pycore.41375/n3.conf# arp -a
? (192.168.2.12) at 00:00:00:aa:00:02 [ether] on eth0
root@n3:/tmp/pycore.41375/n3.conf#
```

g. ¿Qué datos contiene la tabla ARP del router?

Tabla ARP en router r1

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Search, Terminal, Help). The command prompt is root@r1:/tmp/pycore.41375/r1.conf#. The command 'arp -a' has been executed, showing two entries in the ARP table: (192.168.2.10) at MAC address 00:00:00:aa:00:00 on interface eth0, and (192.168.1.10) at MAC address 00:00:00:aa:00:04 on interface eth1.

```
Terminal
File Edit View Search Terminal Help
root@r1:/tmp/pycore.41375/r1.conf# arp -a
? (192.168.2.10) at 00:00:00:aa:00:00 [ether] on eth0
? (192.168.1.10) at 00:00:00:aa:00:04 [ether] on eth1
root@r1:/tmp/pycore.41375/r1.conf#
```

h. ¿Qué son las direcciones de broadcast en IPv4? Cual es su utilidad?

Las direcciones broadcast en IPv4 se usan para enviar un mismo mensaje a todos los dispositivos conectados a dicha red.

i. ¿Qué son las direcciones de multicast en IPv4? Cual es su utilidad?

Permite enviar mensajes a un conjunto de varios (no necesariamente todos) dispositivos conectados a dicha red. Es utilizado en varios protocolos tales como ospf, para la comunicación entre el router designado y los BDR.

## 5) Análisis de tráfico para IPv6

Tabla de vecinos de IPv6 en router R1

Cabe aclarar que las tablas de vecinos (neighbours table) se completa automáticamente en IPv6 al ejecutarse de forma instantánea el protocolo NDP al encender cada interfaz. Para observar los mensajes que se intercambian, se procede a borrar dicha tabla.



```

root@r1:/tmp/pycore.41375/r1.conf# ip -6 neigh
fe80::200:ff:feaa:4 dev eth1 lladdr 00:00:00:aa:00:04 STALE
2001:aaaa:cccc:1::11 dev eth0 lladdr 00:00:00:aa:00:01 STALE
fe80::bcea:f2ff:feb7:b521 dev eth0 lladdr be:ea:f2:b7:b5:21 STALE
fe80::200:ff:feaa:1 dev eth0 lladdr 00:00:00:aa:00:01 STALE
fe80::7cd2:a7ff:fe90:3a5f dev eth1 lladdr 7e:d2:a7:90:3a:5f STALE
2001:aaaa:bbbb:1::10 dev eth1 lladdr 00:00:00:aa:00:04 STALE
fe80::200:ff:feaa:0 dev eth0 lladdr 00:00:00:aa:00:00 STALE
2001:aaaa:cccc:1::10 dev eth0 lladdr 00:00:00:aa:00:00 STALE
fe80::3832:54ff:fe6f:4f3c dev eth0 lladdr ba:49:b8:f0:bd:3e STALE
fe80::8056:d8ff:fe3d:46a5 dev eth1 lladdr 7e:d2:a7:90:3a:5f STALE
root@r1:/tmp/pycore.41375/r1.conf#

```

Tabla de vecinos en h1

```

root@n1:/tmp/pycore.41375/n1.conf# ip -6 neigh
fe80::200:ff:feaa:3 dev eth0 lladdr 00:00:00:aa:00:03 router STALE
2001:aaaa:bbbb:1::11 dev eth0 lladdr 00:00:00:aa:00:03 router STALE
root@n1:/tmp/pycore.41375/n1.conf#

```

Borrar tabla de vecinos en r1

```

root@r1:/tmp/pycore.41375/r1.conf# ip neighbour
192.168.2.10 dev eth0 lladdr 00:00:00:aa:00:00 STALE
192.168.1.10 dev eth1 lladdr 00:00:00:aa:00:04 STALE
fe80::200:ff:feaa:4 dev eth1 lladdr 00:00:00:aa:00:04 STALE
2001:aaaa:cccc:1::11 dev eth0 lladdr 00:00:00:aa:00:01 STALE
fe80::bcea:f2ff:feb7:b521 dev eth0 lladdr be:ea:f2:b7:b5:21 STALE
fe80::200:ff:feaa:1 dev eth0 lladdr 00:00:00:aa:00:01 STALE
fe80::7cd2:a7ff:fe90:3a5f dev eth1 lladdr 7e:d2:a7:90:3a:5f STALE
fe80::200:ff:feaa:0 dev eth0 lladdr 00:00:00:aa:00:00 STALE
2001:aaaa:cccc:1::10 dev eth0 lladdr 00:00:00:aa:00:00 STALE
fe80::3832:54ff:fe6f:4f3c dev eth0 lladdr ba:49:b8:f0:bd:3e STALE
fe80::8056:d8ff:fe3d:46a5 dev eth1 lladdr 7e:d2:a7:90:3a:5f STALE
root@r1:/tmp/pycore.41375/r1.conf# ip neigh flush dev eth0
root@r1:/tmp/pycore.41375/r1.conf# ip neigh flush dev eth1
root@r1:/tmp/pycore.41375/r1.conf# ip neigh

```



## Con tabla limpia, inicio nuevo ping IPv6 y capturo tráfico

```
root@h1:/tmp/pycore.32815/h1.conf
Archivo Editar Ver Buscar Terminal Ayuda
bash: /opt/Xilinx/14.7/ISE_DS/settings64.sh: No existe el archivo o el directorio
root@h1:/tmp/pycore.32815/h1.conf# tcpdump -i eth0 -v
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C01:21:41.890937 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 32) h1 > ff02::1:ff00:11: [icmp6 sum ok] ICMP6, neighbor solicitation, length 32, who has _gateway
    source link-address option (1), length 8 (1): 00:00:00:aa:00:04
01:21:41.891030 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 32) _gateway > h1: [icmp6 sum ok] ICMP6, neighbor advertisement, length 32, tgt is _gateway, Flags
[router, solicited, override]
    destination link-address option (2), length 8 (1): 00:00:00:aa:00:03
01:21:41.891342 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 32) h1 > ff02::1:ff00:11: [icmp6 sum ok] ICMP6, neighbor solicitation, length 32, who has _gateway
    source link-address option (1), length 8 (1): 00:00:00:aa:00:04
01:21:41.891370 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 32) _gateway > h1: [icmp6 sum ok] ICMP6, neighbor advertisement, length 32, tgt is _gateway, Flags
[router, solicited, override]
    destination link-address option (2), length 8 (1): 00:00:00:aa:00:03
01:21:41.891376 IP6 (flowlabel 0xa0625, hlim 64, next-header ICMPv6 (58) payload length: 64) h1 > 2001:aaaa:cccc:1::10: [icmp6 sum ok] ICMP6, echo request, seq 1
01:21:41.891460 IP6 (flowlabel 0x06f1b, hlim 63, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:cccc:1::10 > h1: [icmp6 sum ok] ICMP6, echo reply, seq 1
01:21:42.893291 IP6 (flowlabel 0xa0625, hlim 64, next-header ICMPv6 (58) payload length: 64) h1 > 2001:aaaa:cccc:1::10: [icmp6 sum ok] ICMP6, echo request, seq 2
01:21:42.893434 IP6 (flowlabel 0x06f1b, hlim 63, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:cccc:1::10 > h1: [icmp6 sum ok] ICMP6, echo reply, seq 2

8 packets captured
8 packets received by filter
0 packets dropped by kernel
root@h1:/tmp/pycore.32815/h1.conf#
```

```
root@r1:/tmp/pycore.41375/r1.conf# ip -6 neigh
fe80::200:ff:feaa:4 dev eth1 lladdr 00:00:00:aa:00:04 REACHABLE
fe80::200:ff:feaa:1 dev eth0 lladdr 00:00:00:aa:00:01 REACHABLE
2001:aaaa:cccc:1::11 dev eth0 lladdr 00:00:00:aa:00:01 REACHABLE
2001:aaaa:bbbb:1::10 dev eth1 lladdr 00:00:00:aa:00:04 REACHABLE
```

- a. ¿Cuáles son las comunicaciones NDP que suceden? Identifique los distintos tipos de mensajes NDP haciendo foco en las direcciones IP de origen y destino de cada uno.

Cabe aclarar que el router posee dos direcciones IPv6 distintas en las dos interfaces del router, una dirección local y otra global. Siendo la global la configurada por nosotros. Pero, recolectando información, los usuarios de redes aconsejan no borrar las direcciones locales de interfaces ya que NDP utiliza la misma para su funcionamiento.

Se observan dos tipos de mensajes NDP (Neighbour Discovery Protocol)

- **Neighbour solicitation (NS):** enviado por host o router oara determinar la dirección MAC de un dispositivo vecino (conectado a él) o para verificar si todavía puede comunicarse con él mediante la MAC aprendida.
- **Neighbour advertisement (NA):** mensaje de respuesta a un mensaje de NS. También se usa para inidicar cambios en la dirección de capa de enlace del dispositivo.

En host 1, al intentar enviar el ping hacia el host 3, se produce el intercambio de mensajes siguiente:

i	Origen	Destino	Tipo mensaje
1	2001:aaaa:bbbb:1::10 (n1)	Ff02::1:ff00:11: (dirección multicast hacia el	NS

		router)	
2	2001:aaaa:bbbb:1::11 (r1-1)	2001:aaaa:bbbb:1::10 (n1)	NA
3	2001:aaaa:bbbb:1::10 (n1)	2001:aaaa:cccc:1::10 (n2)	ICMPV6 (ping request)
4	Fe80::200::ff:feaa:2: (interfaz r1 red 2)	Ff02::1:ff00:10: (multicast hacia n2)	NS
5	2001:aaaa:cccc:1::10 (n2)	Fe80::200::ff:feaa:2: (interfaz r1 red 2)	NA
7	2001:aaaa:cccc:1::10 (n2)	2001:aaaa:bbbb:1::10 (n1)	ICMPV6 (ping reply)

b. ¿NDP reemplaza a ARP?

Sí, el reemplazo de ARP en IPv6 es NDP. Es una nueva forma de encontrar las direcciones de capa de enlace de dispositivos en la red local. NDP utiliza mensajes ICMPv6 junto a direcciones multicast para poder lograr su cometido.

c. Describa todas las funciones de NDP

- Router solicitation:

Al activarse una interfaz de red, los hosts envían mensajes de solicitud de routers con el objetivo de que los routers respondan con mensajes de forma inmediata para poder conocer información y dirección de capa de enlace.

- Router advertisement

Un router envía periódicamente este mensaje enviando información de capa de red y de enlace a la red o para dar respuesta a mensajes de solicitud de routers.

Contiene información sobre direcciones de dispositivos de la misma red, configuración de direcciones, valores de salto, etc.

- Neighbour solicitation

Análogo al request de ARP ,enviado por hosts o routers para conocer la dirección de capa de enlace de sus vecinos,a diferencia de ARP estos son enviados mediante multicast,específicamente la dirección con el prefijo ff02::1:. También, puede ser usado para verificar si la dirección almacenada anteriormente todavía es válida para la comunicación con él.

- Neighbour advertisement

Análogo al reply de ARP,es el mensaje enviado mediante unicast como respuesta a

mensajes de solicitud. Además, el dispositivo de red puede enviar un NA para informar sobre cambios en su dirección de capa de enlace.

- Redirect

Utilizado por routers para informar a otros dispositivos de red sobre los mejores caminos disponibles hacia un destino.

- d. ¿Existen direcciones de broadcast en IPv6? Como se reemplaza esta funcionalidad en IPv6?

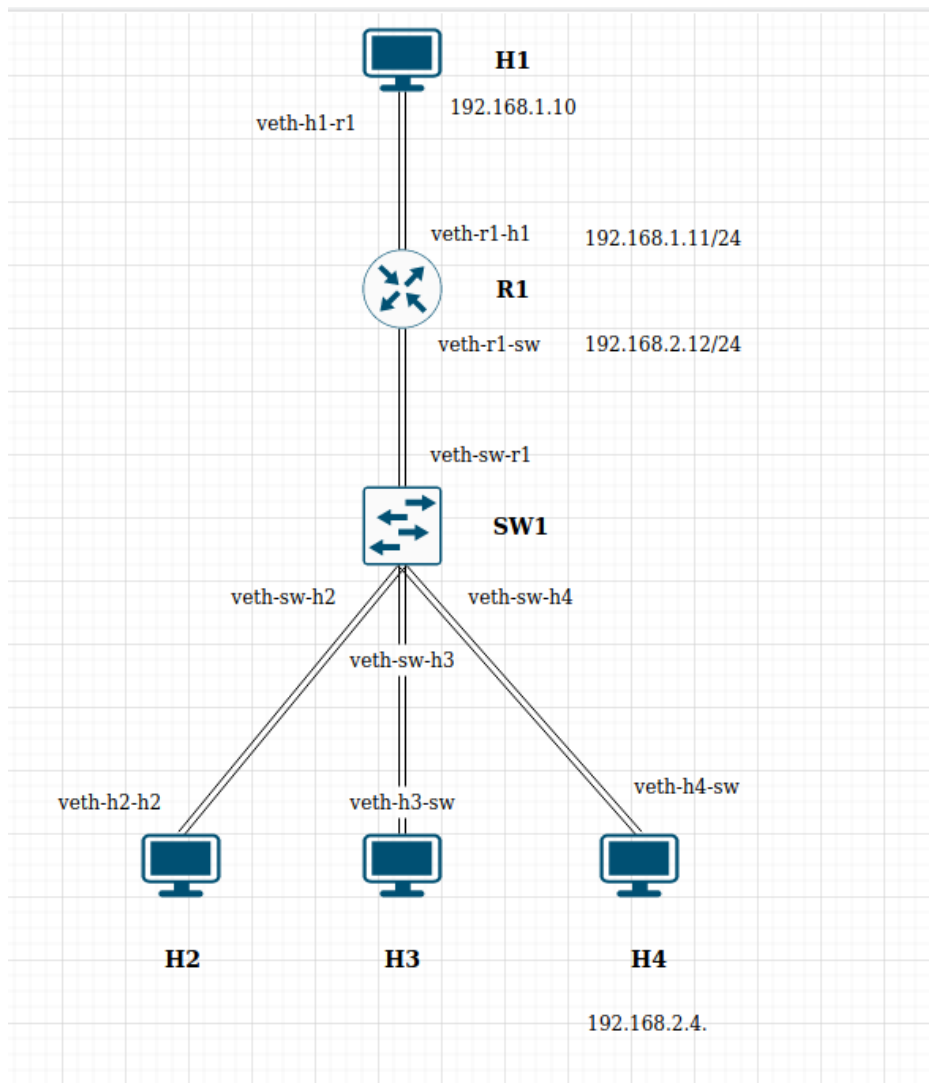
No, IPv6 no implementa direcciones broadcast. En cambio, se utilizan direcciones multicast. En IPv6, una dirección multicast permite llegar a TODOS los dispositivos de red que conforman un grupo. Así, se puede configurar una dirección multicast para varios grupos y enviar un mensaje IPv6 a ellos.

- e. ¿Cuál es la diferencia entre las direcciones link-local, unique-local, global? Ejemplificar. En qué caso usaría a cada una ?

- Direcciones link-local:
  - Empleadas para comunicarse entre dispositivos que comparten un mismo segmento de red.
  - No son ruteables. No se pueden utilizar para comunicar dispositivos de diferente red. Los routers tampoco envían los mensajes a dispositivos en otros enlaces.
  - Prefijo: `fe80::/10`
  - Uso: descubrimiento (NDP) y comunicación entre nodos de una misma red
- Direcciones unique-local:
  - Son análogas a direcciones privadas de IPv4.
  - No son ruteables en Internet.
  - Prefijo: `fc00::/8`
  - Uso en redes privadas.
- Direcciones global:
  - Análogas a direcciones públicas de IPv4
  - Son ruteables
  - Prefijo: `2000::/3`
  - Uso: para comunicación entre hosts de redes distintas a través de internet. Uso para redes públicas.

## Autoconfiguración de direcciones IPv4 en linux namespaces

1. Con linux namespaces defina la topología que se muestra en el diagrama.



2. Configurar un dhcp server en el nuevo host, asegurarse que no entregue la IP del router que se asigno estaticamente.

```
dhcpd.conf - Visual Studio Code [S
File Edit Selection View Go Run Terminal Help

etc > dhcp > dhcpd.conf
1  subnet 192.168.2.0 netmask 255.255.255.0 {
2      range 192.168.2.13 192.168.2.254;
3      default-lease-time 7200;
4      authoritative;
5      option routers 192.168.2.12;
6  }
7  |
```

Ejecucion del script `ns_start.sh` para crear y configurar la topología

```
nadaol@nadaol-Presario-21-VerK: ~/git/Fcefyn_Redes/Tp1
Archivo Editar Ver Buscar Terminal Ayuda

nadaol@nadaol-Presario-21-VerK:~/git/Fcefyn_Redes/Tp1$ ./ns_start.sh
ns created
links created
interfaces conected
links are up
net.ipv4.conf.all.forwarding = 1
ip addresses configured
interfaces are up
nadaol@nadaol-Presario-21-VerK:~/git/Fcefyn_Redes/Tp1$ ip netns list
r1 (id: 12)
h4 (id: 11)
h3 (id: 10)
h2 (id: 8)
h1 (id: 0)
nadaol@nadaol-Presario-21-VerK:~/git/Fcefyn_Redes/Tp1$
```

Comienzo del servidor DHCP mediante el daemon `dhcpd` en el host 4

```
nadaol@nadaol-Presario-21-VerK:~/git/Fcefyn_Redes/Tp1$ sudo ip netns exec h4 dhcpd -user dhcpd -group dhcpd -4 -cf /etc/dhcp/dhcpd.conf veth-h4-sw
Internet Systems Consortium DHCP Server 4.3.5
Copyright 2004-2016 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/
Config file: /etc/dhcp/dhcpd.conf
Database file: /var/lib/dhcp/dhcpd.leases
PID file: /var/run/dhcpd.pid
Wrote 16 leases to leases file.
Listening on LPF/veth-h4-sw/le:db:a6:de:e9:da/192.168.2.0/24
Sending on LPF/veth-h4-sw/le:db:a6:de:e9:da/192.168.2.0/24
Sending on Socket/fallback/fallback-net
nadaol@nadaol-Presario-21-VerK:~/git/Fcefyn_Redes/Tp1$
```

3. Usando el comando `dhclient` se comienza la configuración dinámica de



la IP de h2 y h3.

```
nadaol@nadaol-Presario-21-VerK: ~/git/Fcefyn_Redex/Tp1
Archivo  Editar  Ver  Buscar  Terminal  Ayuda

nadaol@nadaol-Presario-21-VerK:~/git/Fcefyn_Redex/Tp1$ sudo ip netns exec h3 dhclient -4 -d
Internet Systems Consortium DHCP Client 4.3.5
Copyright 2004-2016 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/veth-h3-sw/b6:5d:44:f9:ba:9e
Sending on   LPF/veth-h3-sw/b6:5d:44:f9:ba:9e
Sending on   Socket/fallback
DHCPREQUEST of 192.168.2.223 on veth-h3-sw to 255.255.255.255 port 67 (xid=0x7f3beeed)
DHCPNAK from 192.168.2.4 (xid=0xedee3b7f)
DHCPDISCOVER on veth-h3-sw to 255.255.255.255 port 67 interval 3 (xid=0xf6742d34)
DHCPREQUEST of 192.168.2.224 on veth-h3-sw to 255.255.255.255 port 67 (xid=0x342d74f6)
DHCPOFFER of 192.168.2.224 from 192.168.2.4
DHCPACK of 192.168.2.224 from 192.168.2.4
bound to 192.168.2.224 -- renewal in 3326 seconds.
^C
nadaol@nadaol-Presario-21-VerK:~/git/Fcefyn_Redex/Tp1$ sudo ip netns exec h2 dhclient -4 -d
Internet Systems Consortium DHCP Client 4.3.5
Copyright 2004-2016 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/veth-h2-sw/f2:53:2d:41:8b:ff
Sending on   LPF/veth-h2-sw/f2:53:2d:41:8b:ff
Sending on   Socket/fallback
DHCPREQUEST of 192.168.2.215 on veth-h2-sw to 255.255.255.255 port 67 (xid=0x42f99237)
DHCPNAK from 192.168.2.4 (xid=0x3792f942)
DHCPDISCOVER on veth-h2-sw to 255.255.255.255 port 67 interval 3 (xid=0xd2cf682b)
DHCPREQUEST of 192.168.2.225 on veth-h2-sw to 255.255.255.255 port 67 (xid=0x2b68cfd2)
DHCPOFFER of 192.168.2.225 from 192.168.2.4
DHCPACK of 192.168.2.225 from 192.168.2.4
bound to 192.168.2.225 -- renewal in 3259 seconds.
```

4. Qué direcciones se les asignaron?

Direcciones asignadas mediante DHCP a hosts h2 y h3	
Host	Dirección IPv4
h2	192.168.2.225/24
h3	192.168.2.224/24

```
nadaol@nadaol-Presario-21-VerK:~/git/Fcefyn_Redex/Tp1$ sudo ip netns exec h3 ifconfig
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Bucle local)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

veth-h3-sw: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.2.224 netmask 255.255.255.0 broadcast 192.168.2.255
    inet6 fe80::b45d:44ff:fe9:ba9e prefixlen 64 scopeid 0x20<link>
    ether b6:5d:44:f9:ba:9e txqueuelen 1000 (Ethernet)
    RX packets 121 bytes 15086 (15.0 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 25 bytes 2458 (2.4 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

nadaol@nadaol-Presario-21-VerK:~/git/Fcefyn_Redex/Tp1$ sudo ip netns exec h2 ifconfig
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Bucle local)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

veth-h2-sw: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.2.225 netmask 255.255.255.0 broadcast 192.168.2.255
    inet6 fe80::f053:2dff:fe41:8bff prefixlen 64 scopeid 0x20<link>
    ether f2:53:2d:41:8b:ff txqueuelen 1000 (Ethernet)
    RX packets 124 bytes 15336 (15.3 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 24 bytes 2388 (2.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

5. Explique brevemente y con capturas (tcpdump o wireshark) como funciona DHCP y los mensajes que intervienen.

dhcpcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

bootp.option.type == 53

No.	Time	Source	Destination	Protocol	Length	Info
494	16.902169722	0.0.0.0	255.255.255.255	DHCP	344	DHCP Discover - Transaction ID 0x...
495	16.902173412	0.0.0.0	255.255.255.255	DHCP	344	DHCP Discover - Transaction ID 0x...
496	16.902149282	0.0.0.0	255.255.255.255	DHCP	344	DHCP Discover - Transaction ID 0x...
505	17.738830515	192.168.2.4	192.168.2.218	DHCP	344	DHCP Offer - Transaction ID 0x...
506	17.738844466	192.168.2.4	192.168.2.218	DHCP	344	DHCP Offer - Transaction ID 0x...
507	17.739048418	0.0.0.0	255.255.255.255	DHCP	344	DHCP Request - Transaction ID 0x...
508	17.739061851	0.0.0.0	255.255.255.255	DHCP	344	DHCP Request - Transaction ID 0x...
509	17.739065732	0.0.0.0	255.255.255.255	DHCP	344	DHCP Request - Transaction ID 0x...
510	17.739068671	0.0.0.0	255.255.255.255	DHCP	344	DHCP Request - Transaction ID 0x...
511	17.739048418	0.0.0.0	255.255.255.255	DHCP	344	DHCP Request - Transaction ID 0x...
517	17.803240846	192.168.2.4	192.168.2.218	DHCP	344	DHCP ACK - Transaction ID 0x...
518	17.803255455	192.168.2.4	192.168.2.218	DHCP	344	DHCP ACK - Transaction ID 0x...
524	17.903035215	192.168.2.4	192.168.2.219	DHCP	344	DHCP Offer - Transaction ID 0x...
525	17.903049400	192.168.2.4	192.168.2.219	DHCP	344	DHCP Offer - Transaction ID 0x...
526	17.903292827	0.0.0.0	255.255.255.255	DHCP	344	DHCP Request - Transaction ID 0x...
527	17.903305415	0.0.0.0	255.255.255.255	DHCP	344	DHCP Request - Transaction ID 0x...
528	17.903311733	0.0.0.0	255.255.255.255	DHCP	344	DHCP Request - Transaction ID 0x...
529	17.903315358	0.0.0.0	255.255.255.255	DHCP	344	DHCP Request - Transaction ID 0x...
530	17.903292827	0.0.0.0	255.255.255.255	DHCP	344	DHCP Request - Transaction ID 0x...
536	17.937832205	192.168.2.4	192.168.2.219	DHCP	344	DHCP ACK - Transaction ID 0x...
537	17.937847205	192.168.2.4	192.168.2.219	DHCP	344	DHCP ACK - Transaction ID 0x...

**DHCP Discover:** Es el primer paso de la comunicación cliente/servidor DHCP. El cliente DHCP envía un mensaje mediante broadcast con el objetivo de encontrar un servidor DHCP en la red y conocer los parámetros de red que el servidor

**DHCP Offer:** el servidor DHCP responde ante el mensaje Discover con éste mensaje que ofrece al cliente una dirección IPv4, un tiempo de préstamo (lease time) y otros parámetros de red.

**DHCP ACK:** El servidor elegido por el cliente recibe un DHCPREQUEST y envía este ACK como respuesta. DHCPACK contiene campos que indican la dirección IP prestada al cliente, tiempo de préstamo y parámetros de red que se configuran para el cliente.

### Interacción host 2 con servidor DHCP en host 4: Trasacción ID 0x66x46c6d

048418	0.0.0.0	255.255.255.255	DHCP	344 DHCP Request - Transaction ID 0x
<pre> address: 0.0.0.0 ent) IP address: 0.0.0.0 er IP address: 0.0.0.0 nt IP address: 0.0.0.0 c address: a6:e3:47:10:64:90 (a6:e3:47:10:64:90) rdware address padding: 00000000000000000000 st name not given name not given ie: DHCP (3) DHCP Message Type (Request) 1 request (3) (4) DHCP Server Identifier 4 rver Identifier: 192.168.2.4 (0) Requested IP Address 4 ed IP Address: 192.168.2.218 (5) Parameter Request List (55) End 00... 00 01 00 06 a6 e3 47 10 64 90 00 00 08 00 ..... G.d.....           </pre>				

Finalmente, el servidor DHCP envía el mensaje DHCPACK.

```
▼ Dynamic Host Configuration Protocol (ACK)
  Message type: Boot Reply (2)
  Hardware type: Ethernet (0x01)
  Hardware address length: 6
  Hops: 0
  Transaction ID: 0x66c46c6d
  Seconds elapsed: 0
  ▶ Bootp flags: 0x0000 (Unicast)
  Client IP address: 0.0.0.0
  Your (client) IP address: 192.168.2.51
  Next server IP address: 192.168.2.4
  Relay agent IP address: 0.0.0.0
  Client MAC address: c6:1b:9b:95:34:ea (c6:1b:9b:95:34:ea)
  Client hardware address padding: 00000000000000000000
  Server host name not given
  Boot file name not given
```

Interacción host 3 con servidor DHCP en host 4: Trasacción ID 0xc2967d3f

Ídem a la interacción con host 2.

```
▼ Dynamic Host Configuration Protocol (ACK)
  Message type: Boot Reply (2)
  Hardware type: Ethernet (0x01)
  Hardware address length: 6
  Hops: 0
  Transaction ID: 0xc2967d3f
  Seconds elapsed: 3
  ▶ Bootp flags: 0x0000 (Unicast)
  Client IP address: 0.0.0.0
  Your (client) IP address: 192.168.2.222
  Next server IP address: 192.168.2.4
  Relay agent IP address: 0.0.0.0
  Client MAC address: 82:c4:61:ea:9d:df (82:c4:61:ea:9d:df)
  Client hardware address padding: 00000000000000000000
  Server host name not given
  Boot file name not given
  Magic cookie: DHCP
```

6. Hay conectividad entre h1 y el resto de los hosts ? Por qué ? Por que con IPv6 no tuvimos este inconveniente ? Realice las configuraciones necesarias para que funcione el ping entre h1 y el resto de los hosts.

No, en ipv4 no hay conectividad entre h1 y el resto de los hosts . Sucede que en ipv6 el protocolo ndp se encarga de encontrar las direcciones a nodos vecinos tales como el default gateway mediante los mensajes de router solicitation y advertisement. Para solucionarlo se deben configurar las rutas por default de los host's al gateway correspondiente.

```
root@asusmint:/home/adminit/RedesDeComputadoras_2020/Ip1# ip netns exec h2 ping -c 3 192.168.1.10
PING 192.168.1.10 (192.168.1.10) 56(84) bytes of data.
--- 192.168.1.10 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2043ms
```

```

root@asusmint:/home/admint/RedesDecomputadoras_2020/Tp1# ip netns exec h1 ip route add 192.168.2.0/24 via 192.168.1.11
root@asusmint:/home/admint/RedesDecomputadoras_2020/Tp1# ip netns exec h1 ip route
default via 192.168.1.11 dev veth-h1-r1
192.168.1.0/24 dev veth-h1-r1 proto kernel scope link src 192.168.1.10
192.168.2.0/24 via 192.168.1.11 dev veth-h1-r1
root@asusmint:/home/admint/RedesDecomputadoras_2020/Tp1# ip netns exec h2 ip route add 192.168.1.0/24 via 192.168.2.12
root@asusmint:/home/admint/RedesDecomputadoras_2020/Tp1# ip netns exec h2 ip route
default via 192.168.2.4 dev veth-h2-sw
192.168.1.0/24 via 192.168.2.12 dev veth-h2-sw
192.168.2.0/24 dev veth-h2-sw proto kernel scope link src 192.168.2.51
root@asusmint:/home/admint/RedesDecomputadoras_2020/Tp1# ip netns exec h3 ip route add 192.168.1.0/24 via 192.168.2.12
root@asusmint:/home/admint/RedesDecomputadoras_2020/Tp1# ip netns exec h3 ip route
default via 192.168.2.4 dev veth-h3-sw
192.168.1.0/24 via 192.168.2.12 dev veth-h3-sw
192.168.2.0/24 dev veth-h3-sw proto kernel scope link src 192.168.2.222
root@asusmint:/home/admint/RedesDecomputadoras_2020/Tp1# ip netns exec h4 ip route add 192.168.1.0/24 via 192.168.2.12
root@asusmint:/home/admint/RedesDecomputadoras_2020/Tp1# ip netns exec h4 ip route
default via 192.168.2.12 dev veth-h4-sw
192.168.1.0/24 via 192.168.2.12 dev veth-h4-sw
192.168.2.0/24 dev veth-h4-sw proto kernel scope link src 192.168.2.4

```

```

root@asusmint:/home/admint/RedesDecomputadoras_2020/Tp1# ip netns exec h2 ping -c 3 192.168.1.10
PING 192.168.1.10 (192.168.1.10) 56(84) bytes of data.
64 bytes from 192.168.1.10: icmp_seq=1 ttl=63 time=0.438 ms
64 bytes from 192.168.1.10: icmp_seq=2 ttl=63 time=0.100 ms
64 bytes from 192.168.1.10: icmp_seq=3 ttl=63 time=0.105 ms

--- 192.168.1.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2032ms
rtt min/avg/max/mdev = 0.100/0.214/0.438/0.158 ms
root@asusmint:/home/admint/RedesDecomputadoras_2020/Tp1# ip netns exec h3 ping -c 3 192.168.1.10
PING 192.168.1.10 (192.168.1.10) 56(84) bytes of data.
64 bytes from 192.168.1.10: icmp_seq=1 ttl=63 time=0.105 ms
64 bytes from 192.168.1.10: icmp_seq=2 ttl=63 time=0.094 ms
64 bytes from 192.168.1.10: icmp_seq=3 ttl=63 time=0.087 ms

--- 192.168.1.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2049ms
rtt min/avg/max/mdev = 0.087/0.095/0.105/0.010 ms
root@asusmint:/home/admint/RedesDecomputadoras_2020/Tp1# ip netns exec h4 ping -c 3 192.168.1.10
PING 192.168.1.10 (192.168.1.10) 56(84) bytes of data.
64 bytes from 192.168.1.10: icmp_seq=1 ttl=63 time=0.123 ms
64 bytes from 192.168.1.10: icmp_seq=2 ttl=63 time=0.091 ms
64 bytes from 192.168.1.10: icmp_seq=3 ttl=63 time=0.084 ms

--- 192.168.1.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2036ms
rtt min/avg/max/mdev = 0.084/0.099/0.123/0.018 ms

```