

# Práctico FCEFyN Redes de computadoras

## Trabajo Práctico 1

### INTEGRANTES:

Callegari Viviana, 31450277

Casabella Martín, 39694763

Castro Petro, 36187007

Detke Ramiro, 38122152

Docente: Matías R. Cuenca del Rey

Mail: [mcuenca@unc.edu.ar](mailto:mcuenca@unc.edu.ar)

Ayudantes alumnos: Elisabeth Leonhard - Andrés Serjoy

Redes de computadoras  
Facultad de Ciencias Exactas, Físicas y Naturales  
Universidad Nacional de Córdoba

## Práctico 1: Tráfico en capa de enlace relacionado con IPv4 e IPv6

### Ejercicio 1: Tráfico IPv4 e IPv6 con CORE

Recomendaciones

Esquema

Diagrama

Tabla de asignación de direcciones IPv4 e IPv6

Links de ayuda

Consignas

Configuración de red IPv4/IPv6

### Ejercicio 2: Ruteo estático IPv4/IPv6 con Linux

Recomendaciones

Esquema

Diagrama

Tabla de asignación de direcciones IPv4 e IPv6

Links de ayuda

Consignas

Configuración de red IPv4/IPv6

### Ejercicio 3: Tecnología: namespaces

Links de ayuda

Consignas

Preguntas

# Práctico 1: Tráfico en capa de enlace relacionado con IPv4 e IPv6

Presentación teórica. Análisis de tráfico en capa de enlace cuando hay tráfico IPv4 e IPv6 en capa de red.

Presentación de consignas

Bibliografía: Douglas E. Comer hasta Capítulo 9 inclusive.

## Ejercicio 1: Tráfico IPv4 e IPv6 con CORE

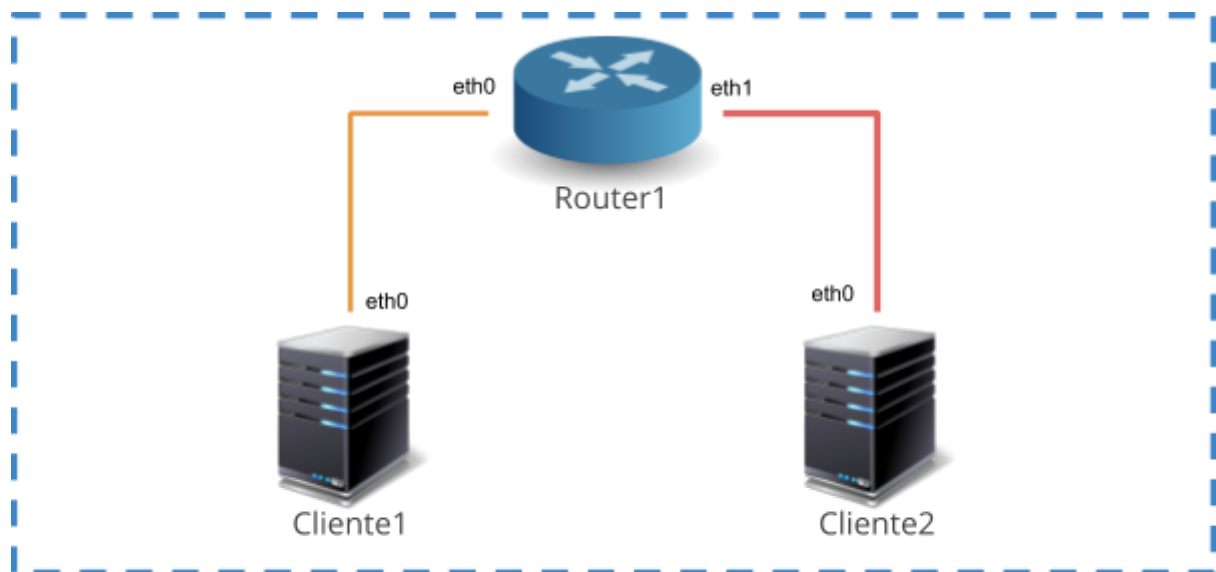
### Recomendaciones

- Lea con cuidado las consignas
- Tenga certeza de los comandos que ejecuta

### Esquema

- Uso de emulador para los tres equipos
- El router no tiene ninguna configuración particular.
- Las computadoras cliente son computadoras emuladas.

### Diagrama



Computadora física 1

### Tabla de asignación de direcciones IPv4 e IPv6

Computadora	Interfaz de red	Dirección IP
Cliente1	Interfaz 1	IPv4: 192.168.1.10/24
		IPv6: 2001:aaaa:bbbb:1::10/64

Cliente2	Interfaz 1	IPv4: 192.168.2.10/24
		IPv6: 2001:aaaa:dddd:1::10/64
Router1	Interfaz 1	IPv4: 192.168.1.1/24
		IPv6: 2001:aaaa:bbbb:1::1/64
	Interfaz 2	IPv4: 192.168.2.1/24
		IPv6: 2001:aaaa:dddd:1::1/64

Links de ayuda

Consignas

Configuración de red IPv4/IPv6

- 1.- Crear el esquema de red sobre el software de emulación CORE.
- 2.- Probar conectividad entre el Cliente1 y Cliente2 enviando 3 paquetes ICMPv4 usando el comando "ping" para IPv4.
- 3.- Probar conectividad entre el Cliente1 y Cliente2 enviando 3 paquetes ICMPv6 usando el comando "ping6" para IPv6.
- 4.- Iniciar tráfico ICMPv4 en el Cliente1 con destino Cliente2. Analizar tráfico con "tcpdump" sobre las dos redes, capturar screenshots y responder las siguientes preguntas:
  - 4.1.- ¿Cuáles son las comunicaciones ARP que suceden?
  - 4.2.- ¿Cuáles son las direcciones IPs en los datagramas IPs?
  - 4.3.- ¿Cómo sabe el router como comunicar un host con otro host?
  - 4.4.- ¿Por qué no hay necesidad de contar con un "switch" en esta topología?
  - 4.5.- ¿Qué datos contiene la tabla ARP del host origen (Cliente1)?
  - 4.6.- ¿Qué datos contiene la tabla ARP del host destino (Cliente2)?
  - 4.7.- ¿Qué datos contiene la tabla ARP del router?
  - 4.8.- ¿Qué son las direcciones de broadcast en IPv4? Cual es su utilidad?
  - 4.9.- ¿Qué son las direcciones de multicast en IPv4? Cual es su utilidad?
- 5.- Iniciar tráfico ICMPv6 en el Cliente1 con destino Cliente2. Analizar el tráfico con "tcpdump" sobre las dos redes, capturar screenshots y responder a las siguientes preguntas:
  - 5.1.- ¿Cuáles son las comunicaciones NDP que suceden?
  - 5.2.- NDP reemplaza a ARP?
  - 5.3.- ¿Cuáles son las diferencias entre NDP y ARP?
  - 5.4.- Describa todas las funciones de NDP
  - 5.5.- ¿Existen direcciones de broadcast en IPv6? Cual es su diferencia con las direcciones de broadcast de IPv4?
  - 5.6.- ¿Cuál es la diferencia entre las direcciones link-local, site-local, global? Ejemplificar.

## Ejercicio 2: Ruteo estático IPv4/IPv6 con Linux

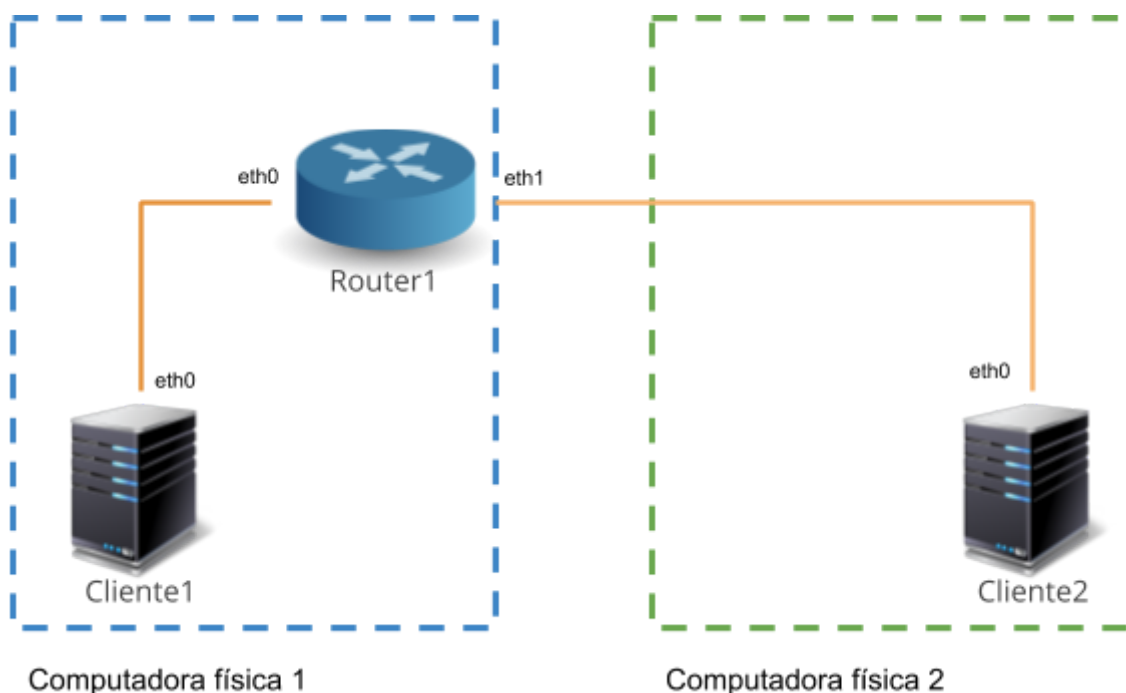
### Recomendaciones

- Lea con cuidado las consignas
- Tenga certeza de los comandos que ejecuta

### Esquema

- La máquina virtual Ubuntu Server será conocida como Router. La máquina virtual Ubuntu Desktop será conocida como cliente.
- En cada computadora, las interfaces de red de las dos máquinas virtuales estarán conectadas mediante 'adaptador puente'.
- Las máquinas virtuales Clientes tendrán una sola interfaz de red. La máquina virtual Servidor tendrá dos interfaces de red.
- Conecte las dos computadoras físicas con cable UTP cruzado.

### Diagrama



### Tabla de asignación de direcciones IPv4 e IPv6

Computadora	Interfaz de red	Dirección IP
Cliente1	eth0	IPv4: 192.168.1.10/24
		IPv6: 2001:aaaa:bbbb:1::10/64
Cliente2	eth0	IPv4: 192.168.2.10/24

		IPv6: 2001:aaaa:dddd:1::10/64
Router1	eth0	IPv4: 192.168.1.1/24
		IPv6: 2001:aaaa:bbbb:1::1/64
	eth1	IPv4: 192.168.2.1/24
		IPv6: 2001:aaaa:dddd:1::1/64

### Links de ayuda

Configuración IPv4 de manera estática en interfaces de red en Ubuntu server

<https://help.ubuntu.com/lts/serverguide/network-configuration.html>

Configuración de Ubuntu como Router

<http://opensourceforu.efytimes.com/2015/04/how-to-configure-ubuntu-as-a-router/>

Configuración IPv4 de manera estática en interfaces de red en Ubuntu desktop

<https://help.ubuntu.com/lts/ubuntu-help/net-fixed-ip-address.html>

### Consignas

Configuración de red IPv4/IPv6

- 1.- Sobre el Router: Configurar de manera permanente las interfaces de red con las direcciones IP correspondientes.
- 2.- Sobre el Router: Configurar para que realice ip\_forwarding de manera permanente.
- 3.- Sobre los Clientes: Utilizando la aplicación de configuración de red gráfica NetworkManager, asignar de manera permanente y las direcciones IPs correspondiente. Configurar como Default Gateway el Router que pertenezca a la misma red.
- 4.- Sobre los Clientes: Con la configuración hecha hasta ahora. Ejecutar los siguientes tests y responder las siguientes preguntas
  - 4.1.- Ping al Default gateway. Explicar el proceso de comunicación. Para IPv4: Protocolos ARP, IPv4 e ICMP. Para IPv6: Protocolos NDP, IPv6 e ICMPv6
  - 4.2.- Ping a el otro Cliente. Explicar el proceso de comunicación. Para IPv4: Protocolos ARP, IPv4 e ICMP. Para IPv6: Protocolos NDP, IPv6 e ICMPv6
- 5.- Restaurar Clientes y Routers a su configuración original

### Ejercicio 3: Tecnología: namespaces

#### Links de ayuda

Namespaces en Linux. Ejemplo básico:

<http://blog.scottlowe.org/2013/09/04/introducing-linux-network-namespaces/>

Como crear interfaces dummy:

<http://www.pocketnix.org/posts/Linux%20Networking:%20Dummy%20Interfaces%20and%20Virtual%20Bridges>

Linux bridge con Namespaces:

<http://www.opencloudblog.com/?p=66>

IP Forwarding:

<http://www.ducea.com/2006/08/01/how-to-enable-ip-forwarding-in-linux/>

## Consignas

### Preguntas

1.- Responda las siguientes preguntas

1.1.- ¿Qué es Linux Namespace? ¿Cómo funciona?

1.2.- Linux Bridge, ¿A qué dispositivo de red emula? ¿Por qué?

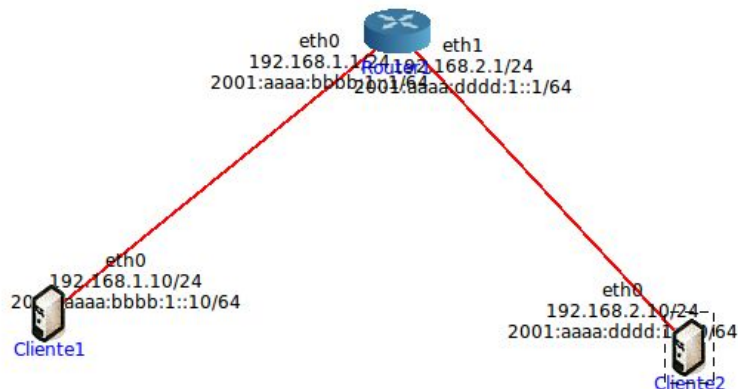
1.3.- ¿Qué es el “veth pair”?

1.4.- ¿Es capaz de crear dos namespaces y conectarlos entre sí?

1.5.- ¿Puede determinar si el software CORE usa linux namespaces. Cómo puede determinar eso?

## Ejercicio 1: Tráfico IPv4 e IPv6 con CORE

1.- Crear el esquema de red sobre el software de emulación CORE.



2.- Probar conectividad entre el Cliente1 y Cliente2 enviando 3 paquetes ICMPv4 usando el comando “ping” para IPv4.

rdc-desktop [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

Terminal

File Edit Canvas View Tools Widgets Session Help



```
root@Cliente1:/tmp/pycore.35265/Cliente1.conf# ping 192.168.2.10 -c 3
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=0.036 ms
64 bytes from 192.168.2.10: icmp_seq=2 ttl=63 time=0.033 ms
64 bytes from 192.168.2.10: icmp_seq=3 ttl=63 time=0.038 ms

--- 192.168.2.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2041ms
rtt min/avg/max/mdev = 0.033/0.035/0.038/0.007 ms
root@Cliente1:/tmp/pycore.35265/Cliente1.conf#
```



### 3.- Probar conectividad entre el Cliente1 y Cliente2 enviando 3 paquetes ICMPv6 usando el comando “ping6” para IPv6.

rac-desktop [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

Terminal

```
root@Cliente2:/tmp/pycore.35265/Cliente2.conf# ping6 2001:aaaa:bbbb:1::1 -c 3
PING 2001:aaaa:bbbb:1::1(2001:aaaa:bbbb:1::1) 56 data bytes
64 bytes from 2001:aaaa:bbbb:1::1: icmp_seq=1 ttl=64 time=0.039 ms
64 bytes from 2001:aaaa:bbbb:1::1: icmp_seq=2 ttl=64 time=0.034 ms
64 bytes from 2001:aaaa:bbbb:1::1: icmp_seq=3 ttl=64 time=0.034 ms

--- 2001:aaaa:bbbb:1::1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2042ms
rtt min/avg/max/mdev = 0.034/0.035/0.039/0.007 ms
root@Cliente2:/tmp/pycore.35265/Cliente2.conf#
```

4.- Iniciar tráfico ICMPv4 en el Cliente1 con destino Cliente2. Analizar tráfico con “tcpdump” sobre las dos redes, capturar screenshots y responder las siguientes preguntas:

```
root@Cliente1:/tmp/pycore.35265/Cliente1.conf# tcpdump -v -c 12
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
00:59:03.676961 IP (tos 0x0, ttl 64, id 18778, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.1.10 > 192.168.2.10: ICMP echo request, id 83, seq 118, length 64
00:59:03.677000 IP (tos 0x0, ttl 63, id 6162, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.2.10 > 192.168.1.10: ICMP echo reply, id 83, seq 118, length 64
00:59:04.700212 IP (tos 0x0, ttl 64, id 18854, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.1.10 > 192.168.2.10: ICMP echo request, id 83, seq 119, length 64
00:59:04.700237 IP (tos 0x0, ttl 63, id 6222, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.2.10 > 192.168.1.10: ICMP echo reply, id 83, seq 119, length 64
00:59:05.724896 IP (tos 0x0, ttl 64, id 18915, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.1.10 > 192.168.2.10: ICMP echo request, id 83, seq 120, length 64
00:59:05.724923 IP (tos 0x0, ttl 63, id 6362, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.2.10 > 192.168.1.10: ICMP echo reply, id 83, seq 120, length 64
00:59:06.370516 IP (tos 0xc0, ttl 1, id 10323, offset 0, flags [none], proto OSPF (89), length 64)
    192.168.1.1 > 224.0.0.5: OSPFv2, Hello, length 44
        Router-ID 192.168.1.1, Backbone Area, Authentication Type: none (0)
        Options [External]
        Hello Timer 10s, Dead Timer 40s, Mask 255.255.255.0, Priority 1
        Designated Router 192.168.1.1
00:59:06.717000 IP6 (class 0xc0, flowlabel 0x767d8, hlim 1, next-header OSPF (89) payload length: 36) fe80::200:ff:feaa:1 > ff02::5: OSPFv3, Hello, length 36
    Router-ID 192.168.1.1, Backbone Area
    Options [V6, External, Router]
    Hello Timer 10s, Dead Timer 40s, Interface-ID 0.0.0.6, Priority 1
    Designated Router 192.168.1.1
00:59:06.723888 IP (tos 0x0, ttl 64, id 19038, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.1.10 > 192.168.2.10: ICMP echo request, id 83, seq 121, length 64
00:59:06.723906 IP (tos 0x0, ttl 63, id 6497, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.2.10 > 192.168.1.10: ICMP echo reply, id 83, seq 121, length 64
00:59:07.740445 IP (tos 0x0, ttl 64, id 19115, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.1.10 > 192.168.2.10: ICMP echo request, id 83, seq 122, length 64
00:59:07.740473 IP (tos 0x0, ttl 63, id 6669, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.2.10 > 192.168.1.10: ICMP echo reply, id 83, seq 122, length 64
12 packets captured
12 packets received by filter
0 packets dropped by kernel
```



#### 4.1.- ¿Cuáles son las comunicaciones ARP que suceden?

Al hacer PING desde Cliente1 a Cliente2, en el IP de Cliente1 (fuente), el protocolo lo lleva en un campo, junto con la dirección MAC de fuente, de destino, y la dirección IP de destino. En este caso, debe comenzar interactuando con el router intermediario al estar ambos host en redes diferentes.

Se pueden ver las distintas instancias utilizando *tcpdump* y sus comandos de filtrado de paquetes.

Configuración Router:

```
[root@Cliente1 Cliente1.conf]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.10 netmask 255.255.255.0 broadcast 0.0.0.0
    inet6 fe80::200:ff:feaa:0 prefixlen 64 scopeid 0x20<link>
    inet6 2001:aaaa:bbbb:1::10 prefixlen 64 scopeid 0x0<global>
    ether 00:00:00:aa:00:00 txqueuelen 1000 (Ethernet)
    RX packets 488 bytes 47180 (46.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 465 bytes 44006 (42.9 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.1 netmask 255.255.255.0 broadcast 0.0.0.0
    inet6 fe80::200:ff:feaa:1 prefixlen 64 scopeid 0x20<link>
    inet6 2001:aaaa:bbbb:1::1 prefixlen 64 scopeid 0x0<global>
    ether 00:00:00:aa:00:01 txqueuelen 1000 (Ethernet)
    RX packets 493 bytes 46122 (45.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 459 bytes 44954 (43.9 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.2.1 netmask 255.255.255.0 broadcast 0.0.0.0
    inet6 2001:aaaa:dddd:1::1 prefixlen 64 scopeid 0x0<global>
    inet6 fe80::200:ff:feaa:2 prefixlen 64 scopeid 0x20<link>
    ether 00:00:00:aa:00:02 txqueuelen 1000 (Ethernet)
    RX packets 450 bytes 42064 (41.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 415 bytes 39658 (38.7 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Paquetes ARP: Se observa como el Cliente 1 pregunta por la dirección MAC de “gateway” que en este caso se equivale con la dirección IP de la interfaz eth0 del router. Y se ve el paquete de respuesta con la dirección MAC de dicha interfaz.

```
ARP, Ethernet (len 6), IPv4 (len 4), Request who-has _gateway tell Cliente1, length 28
ARP, Ethernet (len 6), IPv4 (len 4), Reply _gateway is-at 00:00:00:aa:00:01 (oui Ethernet), length 28
```

Una vez que el mensaje llega al router, este pregunta por la dirección MAC del Cliente 2 de manera similar a como lo hizo el Cliente 1 con el router, y una vez que la conoce envía el mensaje ICMP. Para la respuesta del mensaje ahora ya se conocen todas las direcciones MAC necesarias por lo tanto el mensaje de respuesta viaja sin necesidad de volver a consultar por las MACs.

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

El IP de fuente, el protocolo lo lleva en un campo, junto con la dirección MAC de fuente, de destino, y la dirección IP de destino. En este caso, debe comenzar interactuando con el router intermediario al estar ambos host en redes diferentes.

Las direcciones IP's en los datagramas IP varían según qué paquete se capture.

- Paquetes ICMP: tienen el formato **192.168.s.s > 192.168.d.d: .....**  
(source) (destiny)

#### 4.3.- ¿Cómo sabe el router como comunicar un host con otro host?

El Router sabe cómo comunicar los host gracias a las tablas de ruteo que posee, estas almacenan información sobre hacia dónde debe hacerse el siguiente salto en base a la dirección que se quiere alcanzar.

A su vez, contiene una tabla ARP que se va actualizando automáticamente. En ella, entre otras cosas, contiene en una de sus entradas contiene la dirección IP y la dirección MAC de un Cliente conectado a él y que ha intercambiado información a través de él.

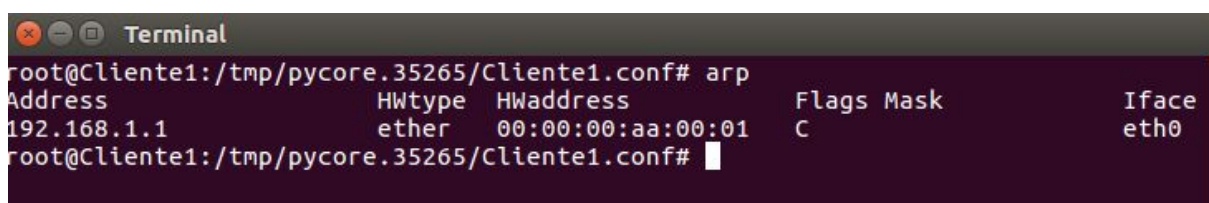
Al intercambiar mensajes ARP entre clientes y el Router, el Router agrega una entrada a su tabla ARP, es decir, la actualiza con la información recibida. Así es como el router hace de intermediario, buscando en la tabla ARP la entrada que contenga la dirección IP coincidente con la dirección IP que figura como dirección de destino en el datagrama cuando dos hosts conectados al router quieren comunicarse

#### 4.4.- ¿Por qué no hay necesidad de contar con un “switch” en esta topología?

Esta topología referencia a dos redes diferentes, conectadas mediante un router. Si se quisiera agregar más clientes a alguna de las 2 redes, si se necesitaría un switch. En este caso hay un solo host por cada red, y no se requiere dicho dispositivo.

#### 4.5.- ¿Qué datos contiene la tabla ARP del host origen (Cliente1)?

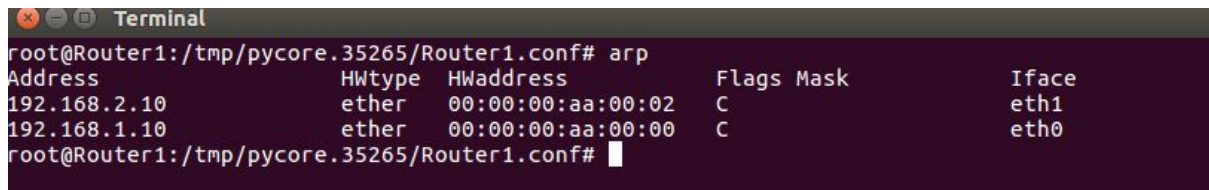
La tabla ARP del Cliente 1, luego del intercambio de mensajes ARP, contiene la dirección MAC que se corresponde con la IP de la interfaz del Router a la que se encuentra directamente conectada es decir, la interfaz eth0 en el caso de nuestra simulación, junto con el tipo de hardware empleado por la interfaz (HWType), el nombre de la interfaz, e indicadores (Flag Mask).



```
root@Cliente1:/tmp/pycore.35265/Cliente1.conf# arp
Address      HWtype  HWaddress    Flags Mask    Iface
192.168.1.1  ether   00:00:00:aa:00:01  C             eth0
root@Cliente1:/tmp/pycore.35265/Cliente1.conf#
```

#### 4.7.- ¿Qué datos contiene la tabla ARP del router?

La tabla ARP del Router contiene la MAC asociada a la dirección IP del Cliente 1 y también la MAC que se corresponde con la IP del Cliente 2, junto el resto de información adicional.



```
root@Router1:/tmp/pycore.35265/Router1.conf# arp
Address          HWtype  HWaddress      Flags Mask    Iface
192.168.2.10     ether   00:00:00:aa:00:02 C              eth1
192.168.1.10     ether   00:00:00:aa:00:00 C              eth0
root@Router1:/tmp/pycore.35265/Router1.conf#
```

#### 4.8.- ¿Qué son las direcciones de broadcast en IPv4? Cual es su utilidad?

Son las direcciones que permiten hacer llegar un mensaje a todos los nodos de una misma LAN. Es de mera utilidad cuando un nodo de la red necesite enviar información al resto de los nodos, esto se puede conseguir enviando la información a través de la dirección de broadcast.

Como utilidad a destacar se puede mencionar que sirven para encontrar una dirección especificada dentro de una subred para construir el camino por donde se va a enviar posteriormente un mensaje.

#### 4.9.- ¿Qué son las direcciones de multicast en IPv4? Cual es su utilidad?

Las direcciones multicast son identificadores lógicos para un grupo de host pertenecientes a una red determinada.

En sí, el direccionamiento multicast, es un método de envío simultáneo de paquetes (a nivel de IP) que tan sólo serán recibidos por un determinado grupo de receptores, que están interesados en los mismos.

Con la palabra interesados, se hace referencia a que para que el receptor reciba los paquetes que desea, debe haberse suscrito a ese grupo, mediante un tipo de mensaje, IGMP. Este tipo de mensaje además de apuntar al receptor mencionado para que reciba mensajes de cierta dirección, le permite al router tomar conocimiento de que en su interfaz tiene a un equipo interesado en recibir paquetes de una dirección multicast determinada.



**5.- Iniciar tráfico ICMPv6 en el Cliente1 con destino Cliente2. Analizar el tráfico con “tcpdump” sobre las dos redes, capturar screenshots y responder a las siguientes preguntas:**

```

root@Cliente1:~/tmp/pycore.35265/Cliente1.conf# tcpdump -v icmp6
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
01:28:21.244947 IP6 (FlowLabel 0xe2af2, hlin 64, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:bbbb:1::10 > 2001:aaaa:dddd:1::1: [icmp6 sum ok] ICMP6, echo request, seq 14
01:28:21.244966 IP6 (FlowLabel 0xe2af2, hlin 64, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:dddd:1::1 > 2001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo reply, seq 14
01:28:22.268232 IP6 (FlowLabel 0xe2af2, hlin 64, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:bbbb:1::10 > 2001:aaaa:dddd:1::1: [icmp6 sum ok] ICMP6, echo request, seq 15
01:28:22.268253 IP6 (FlowLabel 0xe2af2, hlin 64, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:dddd:1::1 > 2001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo reply, seq 15
01:28:23.292656 IP6 (FlowLabel 0xe2af2, hlin 64, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:bbbb:1::10 > 2001:aaaa:dddd:1::1: [icmp6 sum ok] ICMP6, echo request, seq 16
01:28:23.292676 IP6 (FlowLabel 0xe2af2, hlin 64, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:dddd:1::1 > 2001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo reply, seq 16
01:28:24.316556 IP6 (FlowLabel 0xe2af2, hlin 64, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:bbbb:1::10 > 2001:aaaa:dddd:1::1: [icmp6 sum ok] ICMP6, echo request, seq 17
01:28:24.316573 IP6 (FlowLabel 0xe2af2, hlin 64, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:dddd:1::1 > 2001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo reply, seq 17
01:28:25.340776 IP6 (FlowLabel 0xe2af2, hlin 64, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:bbbb:1::10 > 2001:aaaa:dddd:1::1: [icmp6 sum ok] ICMP6, echo request, seq 18
01:28:25.340796 IP6 (FlowLabel 0xe2af2, hlin 64, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:dddd:1::1 > 2001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo reply, seq 18
01:28:26.364318 IP6 (FlowLabel 0xe2af2, hlin 64, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:bbbb:1::10 > 2001:aaaa:dddd:1::1: [icmp6 sum ok] ICMP6, echo request, seq 19
01:28:26.364337 IP6 (FlowLabel 0xe2af2, hlin 64, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:dddd:1::1 > 2001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo reply, seq 19
01:28:27.388246 IP6 (FlowLabel 0xe2af2, hlin 64, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:bbbb:1::10 > 2001:aaaa:dddd:1::1: [icmp6 sum ok] ICMP6, echo request, seq 20
01:28:27.388264 IP6 (FlowLabel 0xe2af2, hlin 64, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:dddd:1::1 > 2001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo reply, seq 20
01:28:28.412700 IP6 (FlowLabel 0xe2af2, hlin 64, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:bbbb:1::10 > 2001:aaaa:dddd:1::1: [icmp6 sum ok] ICMP6, echo request, seq 21
01:28:28.412718 IP6 (FlowLabel 0xe2af2, hlin 64, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:dddd:1::1 > 2001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo reply, seq 21
  
```

### 5.1.- ¿Cuáles son las comunicaciones NDP que suceden?

Se comunicaron entre sí ambos clientes, y se analizó la situación. Primero, vale aclarar que tipo de mensajes maneja NDP:

Por parte del router, tenemos las siguientes direcciones:

```

ubuntu@ubuntu:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:bc:c0:d3 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.1/24 brd 192.168.1.255 scope global enp0s3
        valid_lft forever preferred_lft forever
    inet6 2001:aaaa:bbbb:1::1/64 scope global
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:febc:c0d3/64 scope link
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:09:47:27 brd ff:ff:ff:ff:ff:ff
    inet 192.168.2.1/24 brd 192.168.2.255 scope global enp0s8
        valid_lft forever preferred_lft forever
    inet6 2001:aaaa:dddd:1::1/6 scope global
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe09:4727/64 scope link
        valid_lft forever preferred_lft forever
ubuntu@ubuntu:~$ _
  
```

Por parte de Cliente1:

```
ubuntu@ubuntu:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:1d:28:b9 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.10/24 brd 192.168.1.255 scope global enp0s3
        valid_lft forever preferred_lft forever
    inet6 2001:aaaa:bbbb:1::10/64 scope global
        valid_lft forever preferred_lft forever
    inet6 fe80::a3cd:ee2c:2297:e3af/64 scope link
        valid_lft forever preferred_lft forever
ubuntu@ubuntu:~$
```

Por parte del Cliente2:

```
kzax@kzax-Aspire-4330: ~
File Edit View Search Terminal Help
21:45:55.873743 IP6 (flowlabel 0x4bbd3, hlim 63, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:bb
e-4330: [icmp6 sum ok] ICMP6, echo request, seq 1678
21:45:55.873783 IP6 (flowlabel 0xd77cb, hlim 64, next-header ICMPv6 (58) payload length: 64) kzax-Aspire-
:1::10: [icmp6 sum ok] ICMP6, echo reply, seq 1678
^C
42 packets captured
42 packets received by filter
0 packets dropped by kernel
kzax@kzax-Aspire-4330:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp5s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:1e:ec:d1:7b:97 brd ff:ff:ff:ff:ff:ff
    inet 192.168.2.10/24 brd 192.168.2.255 scope global enp5s0
        valid_lft forever preferred_lft forever
    inet6 2001:aaaa:dddd:1::10/64 scope global
        valid_lft forever preferred_lft forever
    inet6 fe80::b652:9daa:320b:26ec/64 scope link
        valid_lft forever preferred_lft forever
3: wlp4s0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 00:23:4d:d1:59:d4 brd ff:ff:ff:ff:ff:ff
kzax@kzax-Aspire-4330:~$
```

Un router empieza a enviar mensajes Router Advertisement para todo prefijo de interfaz configurado, ni bien el comando de configuración de ipv6 unicast-routing se ingresa. En una determinada interfaz, los mensajes RA (Router Advertisements) incluyen todos los prefijos IPv6 de 64 bits que estén configurados en esa interfaz. Esto permite la autoconfiguración mencionada, SLAAC (Stateless address autoconfiguration).



Para informar a los hosts acerca de los prefijos IPv6 usados en cada enlace, y también para informar a los mismos que el router está disponible como default gateway, lo que hacen los routers IPv6 es enviar periódicamente mensajes RA.

#### Router Solicitation (RS):

Los hosts, por su parte, al inicio de la conexión envían mensajes Router Solicitation a todas las direcciones multicast de todos los routers presentes.

El envío de RS ocurre periódicamente sin tiempo de espera por un RA. Si el host no tiene configurada una dirección IPv6 en su interfaz, el mensaje RS es enviado sin especificar la dirección de fuente (source address). Si dicho host si tiene configurado en su interfaz una dirección IPv6, entonces la dirección fuente del mensaje RS será esa misma.

#### Neighbor solicitation:

Nodos IPv6 envían estos mensajes para encontrar la dirección MAC de un vecino determinado. Hay 3 operaciones en las cuales este tipo de mensaje se utiliza:

- Para detectar direcciones duplicadas
- Para verificar si determinado vecino es alcanzable
- Mapeo de direcciones de capa de red a direcciones MAC (misma funcionalidad que ARP, integrada como ICMP).

#### Neighbor advertisement:

Los nodos ipv6 mandan mensajes Neighbor Advertisement periódicamente con el fin de informar su presencia a otros host presentes en la misma red y para también enviar sus direcciones de capa de enlace (MAC).

Habiendo establecido la comunicación correspondiente, se capturó lo siguiente en tcpdump:

```
e 262144 bytes
8) payload length: 64) 2001:aaaa:bbbb:1::10 > 2001:aaaa:dddd:1::10: [icmp6 sum ok] ICMP6, echo request,
8) payload length: 64) 2001:aaaa:dddd:1::10 > 2001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo reply, se
8) payload length: 64) 2001:aaaa:bbbb:1::10 > 2001:aaaa:dddd:1::10: [icmp6 sum ok] ICMP6, echo request,
8) payload length: 64) 2001:aaaa:dddd:1::10 > 2001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo reply, se
8) payload length: 64) 2001:aaaa:bbbb:1::10 > 2001:aaaa:dddd:1::10: [icmp6 sum ok] ICMP6, echo request,
8) payload length: 64) 2001:aaaa:dddd:1::10 > 2001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo reply, se
8) payload length: 64) 2001:aaaa:bbbb:1::10 > 2001:aaaa:dddd:1::10: [icmp6 sum ok] ICMP6, echo request,
8) payload length: 64) 2001:aaaa:dddd:1::10 > 2001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo reply, se
```

ICMP6 desde el Cliente1 con destino al Cliente2.



```
(58) payload length: 64) 2001:aaaa:bbbb:1::10 > 2001:aaaa:dddd:1::10: [icmp6 sum ok] ICMP6, echo request,
(58) payload length: 64) 2001:aaaa:dddd:1::10 > 2001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo reply, se
(58) payload length: 64) 2001:aaaa:bbbb:1::10 > 2001:aaaa:dddd:1::10: [icmp6 sum ok] ICMP6, echo request,
(58) payload length: 64) 2001:aaaa:dddd:1::10 > 2001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo reply, se
th: 32) fe80::a00:27ff:febc:c0d3 > 2001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, neighbor solicitation, lengt
c0:d3
th: 24) 2001:aaaa:bbbb:1::10 > fe80::a00:27ff:febc:c0d3: [icmp6 sum ok] ICMP6, neighbor advertisement, leng
(58) payload length: 64) 2001:aaaa:bbbb:1::10 > 2001:aaaa:dddd:1::10: [icmp6 sum ok] ICMP6, echo request,
(58) payload length: 64) 2001:aaaa:dddd:1::10 > 2001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo reply, se
(58) payload length: 64) 2001:aaaa:bbbb:1::10 > 2001:aaaa:dddd:1::10: [icmp6 sum ok] ICMP6, echo request,
(58) payload length: 64) 2001:aaaa:dddd:1::10 > 2001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo reply, se
```

Neighbor Solicitation con origen en la interfaz del Router que pertenece a la misma red que el Cliente1. Luego, Neighbor Advertisement con origen en el Cliente1 y destino en el Router (Default Gateway) (mapeo de MAC Address, se cortó la parte de *who has, tgt is*)

```
lim 64, next-header ICMPv6 (58) payload length: 64) kzax-Aspire-4330 > 2001:aaaa:bbbb:1::10: [icmp6
lim 63, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:bbbb:1::10 > kzax-Aspire-4330: [icmp6
lim 64, next-header ICMPv6 (58) payload length: 64) kzax-Aspire-4330 > 2001:aaaa:bbbb:1::10: [icmp6
r ICMPv6 (58) payload length: 32) kzax-Aspire-4330 > gateway: [icmp6 sum ok] ICMP6, neighbor solicit
length 8 (1): 00:1e:ec:d1:7b:97
r ICMPv6 (58) payload length: 24) gateway > kzax-Aspire-4330: [icmp6 sum ok] ICMP6, neighbor adverti
[router, solicited]
lim 63, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:bbbb:1::10 > kzax-Aspire-4330: [icmp6
lim 64, next-header ICMPv6 (58) payload length: 64) kzax-Aspire-4330 > 2001:aaaa:bbbb:1::10: [icmp6
```

(aparece el nombre de root inclusive capturando con tcpdump. No supimos porqué esas etiquetas en el ordenador conectado por ethernet externamente que simula cliente2)

Neighbor Solicitation con origen en la interfaz del Router que pertenece a la misma red que el Cliente2. Luego, Neighbor Advertisement con origen en el Cliente2 y destino en el Router (Default Gateway)(mapeo de MAC Address, se cortó la parte de *who has, tgt is*).

Se ven casos en donde se pregunta por la ip local, y otros con la ip global (NS, NA):

```
(58) payload length: 64) 2001:aaaa:dddd:1::10 > 2001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6,
(58) payload length: 64) 2001:aaaa:dddd:1::10 > 2001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6,
th: 32) 2001:aaaa:bbbb:1::10 > ff02::1:ff00:1: [icmp6 sum ok] ICMP6, neighbor solicitation, l
:28:b9
th: 32) 2001:aaaa:bbbb:1::1 > 2001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, neighbor advertisem
27:bc:c0:d3
(58) payload length: 64) ip6-localhost > 2001:aaaa:dddd:1::10: [icmp6 sum ok] ICMP6, echo re
(58) payload length: 64) ip6-localhost > 2001:aaaa:dddd:1::10: [icmp6 sum ok] ICMP6, echo re
(58) payload length: 64) 2001:aaaa:bbbb:1::10 > 2001:aaaa:dddd:1::10: [icmp6 sum ok] ICMP6,
(58) payload length: 64) 2001:aaaa:bbbb:1::10 > 2001:aaaa:dddd:1::10: [icmp6 sum ok] ICMP6,
```

Esto se debe a que el router inicialmente pregunta por la dirección IP local, ya que en principio lo único que conoce el router de cierto paquete es a que IP lo tiene que reenviar.

Si se deja la comunicacion corriendo, los mensajes NA y NS se comenzarán a realizar con las direcciones locales, ya conociendolas el router.

```

1:26:22.609687 IP6 (flowlabel 0xaf136, hlim 64, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:bbbb:1::10 >
001:aaaa:dddd:1::10: [icmp6 sum ok] ICMP6, echo reply, seq 59
1:26:23.022912 IP6 (flowlabel 0x4bbd3, hlim 64, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:bbbb:1::10 >
001:aaaa:dddd:1::10: [icmp6 sum ok] ICMP6, echo request, seq 531
1:26:23.023476 IP6 (flowlabel 0xd77cb, hlim 63, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:dddd:1::10 >
001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo reply, seq 531
1:26:23.634209 IP6 (flowlabel 0x5a438, hlim 63, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:dddd:1::10 >
001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo request, seq 60
1:26:23.634232 IP6 (flowlabel 0xaf136, hlim 64, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:bbbb:1::10 >
001:aaaa:dddd:1::10: [icmp6 sum ok] ICMP6, echo reply, seq 60
1:26:23.663116 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 32) fe80::a3cd:ee2c:2297:e3af > fe80::a00:27ff:febc:c0d3: [icmp6 sum ok] ICMP6, neighbor solicitation, length 32, who has fe80::a00:27ff:febc:c0d3
source link-address option (1), length 8 (1): 08:00:27:1d:28:b9
1:26:23.663257 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 24) fe80::a00:27ff:febc:c0d3 > fe80::a3cd:ee2c:2297:e3af: [icmp6 sum ok] ICMP6, neighbor advertisement, length 24, tgt is fe80::a00:27ff:febc:c0d3, Flags [router solicited]
1:26:24.047727 IP6 (flowlabel 0x4bbd3, hlim 64, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:bbbb:1::10 >
001:aaaa:dddd:1::10: [icmp6 sum ok] ICMP6, echo request, seq 532
1:26:24.048314 IP6 (flowlabel 0xd77cb, hlim 63, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:dddd:1::10 >
001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo reply, seq 532
1:26:24.658704 IP6 (flowlabel 0x5a438, hlim 63, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:dddd:1::10 >
001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo request, seq 61
1:26:24.658724 IP6 (flowlabel 0xaf136, hlim 64, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:bbbb:1::10 >
001:aaaa:dddd:1::10: [icmp6 sum ok] ICMP6, echo reply, seq 61
1:26:25.071122 IP6 (flowlabel 0x4bbd3, hlim 64, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:bbbb:1::10 >
001:aaaa:dddd:1::10: [icmp6 sum ok] ICMP6, echo request, seq 533
1:26:25.071752 IP6 (flowlabel 0xd77cb, hlim 63, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:dddd:1::10 >
001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo reply, seq 533
1:26:25.683219 IP6 (flowlabel 0x5a438, hlim 63, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:dddd:1::10 >
001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo request, seq 62
1:26:25.683237 IP6 (flowlabel 0xaf136, hlim 64, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:bbbb:1::10 >
001:aaaa:dddd:1::10: [icmp6 sum ok] ICMP6, echo reply, seq 62

```

Se ve en este caso, como la interfaz del router pregunta y recibe respuesta del host conectado, mediante la direcciones locales. En este caso, se realiza para saber si el vecino (host) sigue siendo alcanzable.

En cuanto a los mensajes RA, y RS, se desconectó y volvió a conectar la interfaz de un cliente, mientras la comunicación establecida y se observó lo siguiente:

```

1:10:10.000000 a.a.1.0.0 Adaptador de Ethernet Ethernet:
A 2001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo request, seq 579
21:27:11.750000 1:1:10 > 2001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo reply, seq 579
21:27:11.750000 1:1:10 > 2001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo request, seq 579
21:27:11.750000 1:1:10 > 2001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo reply, seq 579
21:27:11.950000 1:1:10 > 2001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo request, seq 579
21:27:12.000173 IP6 (flowlabel 0xd77cb, hlim 63, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:dddd:1::10 >
001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo reply, seq 579
21:27:12.234895 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 16) fe80::a412:de27:e05c:b8cd > ip6-allrouters: [icmp6 sum ok] ICMP6, router solicitation, length 16
source link-address option (1), length 8 (1): 4c:cc:6a:83:f5:b2
21:27:12.619430 IP6 (flowlabel 0x3e515, hlim 255, next-header UDP (17) payload length: 53) 2001:aaaa:dddd:1::10.mdns > ff02::fb.mdns: [udp sum ok] 0 [2q] PTR (QM)? _ipps.tcp.local. PTR (QM)? _ipp.tcp.local. (45)
21:27:12.779403 IP6 (flowlabel 0x5a438, hlim 63, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:dddd:1::10 >
001:aaaa:bbbb:1::10: [icmp6 sum ok] ICMP6, echo request, seq 108
21:27:12.779422 IP6 (flowlabel 0xaf136, hlim 64, next-header ICMPv6 (58) payload length: 64) 2001:aaaa:bbbb:1::10 >
001:aaaa:dddd:1::10: [icmp6 sum ok] ICMP6, echo reply, seq 108

```

Se capturó un paquete Router Solicitation, cuyo origen es la dirección IPv6 local real del ordenador (máquina cuyo OS es Windows 10, y mediante Virtual Box corre tanto el server como el cliente).

El destino, refleja que es una dirección multicast (ip6-all routers), manejada por la máquina virtual. Tal como se explicó previamente, los hosts al iniciar la conexión envían mensajes Router Solicitation a todas las direcciones multicast de todos los routers presentes.



## **5.2.- NDP reemplaza a ARP?**

NDP, es la nueva técnica utilizada en IPv6 para descubrir vecinos de una cierta LAN, que consta básicamente en direcciones MAC e IP de un vecino determinado.

Es el reemplazo de ARP para IPv6, ya que la eliminación del broadcast en el protocolo IPv6 afecta a aquellos protocolos que funcionaban con broadcast, tal como es el caso de ARP o DHCP. No solamente podría decirse que reemplaza a ARP, sino que también servirá en apoyo a otros protocolos como por ejemplo “autoconfigurar una IP con STATELESS”, identificar problemas de direccionamiento en nuestra red, entre otros.

NDP se basa en ICMP y utiliza direcciones Multicast (Comunicación de uno a un grupo), y estas direcciones Multicast estarán en el rango de las FF00::/8.

## **5.3.- ¿Cuáles son las diferencias entre NDP y ARP?**

NDP se distingue de ARP porque además de realizar las mismas funcionalidades que ARP, también incorpora funcionalidades de ICMP.

Utiliza mensajes especiales de ICMPv6 construyendo así una manera simple para que los terminales aprendan las direcciones IPv6 de los vecinos de la capa de enlace.

Otra de las grandes funcionalidades de este protocolo es que se ocupa de mantener limpios los cachés donde se almacena la información relativa al contexto de la red a la que está conectado un nodo. Así, cuando una ruta hacia cierto nodo falla, el router correspondiente buscará rutas alternativas. Basándose también en los mensajes ICMPv6 se permite un mecanismo de autoconfiguración

Podría decirse en forma general, que las funcionalidades de NDP en IPv6, corresponden a una combinación de 4 protocolos IPv4: ARP (Address Resolution Protocol), ICMP (Internet Control Message Protocol), Router Discovery (RDP), e ICMP redirect (redireccionamiento ICMP)..

(Se detallarán las funcionalidades en la siguiente respuesta)

## **5.4.- Describa todas las funciones de NDP**

La principal función de NDP, es la de resolver las direcciones IPv6 en direcciones MAC válidas, que son las direcciones de hardware propias de cada dispositivo.

Todas las direcciones determinadas se almacenan en el llamado neighbor cache. Este buffer informa a los componentes de la red sobre las direcciones locales de los clientes vecinos y les suministra información adicional que puede ser necesaria, por ejemplo, para verificar la disponibilidad.

Además, NDP también es un instrumento para la asignación de la puerta de enlace estándar (standard gateway). Con la incorporación del Router Advertisement Protocol (RA) es posible verificar tanto el router estándar como los prefijos de red válidos, dos parámetros fundamentales de la configuración de red. Por último, el protocolo de red, que se encarga únicamente de intercambiar datos dentro de una red, funciona también como protocolo de

apoyo para la configuración dinámica de direcciones. Este proceso también se conoce como configuración automática de direcciones sin estado (Stateless Automatic Address Configuration o SLAAC).

### **5.5.- ¿Existen direcciones de broadcast en IPv6? Cual es su diferencia con las direcciones de broadcast de IPv4?**

A diferencia de IPv4, el protocolo IPv6 no soporta direcciones Broadcast.

El nuevo protocolo IPv6 tiene servicios multicast implementados, por lo que el descubrimiento de la red funciona ya con dichos servicios. Hosts que corran estos servidores estarán escuchando direcciones multicast mientras que los restantes hosts no serán “molestados” cuando cierto cliente envíe paquetes IP a estas direcciones.

El multicast IPv6 comparte protocolos y características comunes con IPv4, pero también incorpora cambios y mejoras. Por lo tanto, no existe el concepto de una dirección de broadcast y así la dirección más alta de la red (la dirección de broadcast en una red IPv4) es considerada una dirección normal en IPv6.

### **5.6.- ¿Cuál es la diferencia entre las direcciones link-local, site-local, global? Ejemplificar.**

Las direcciones Link-Local son el equivalente a las direcciones IP privadas en IPv4. Estas son asignadas a una interfaz de manera automática a partir del momento que activamos el protocolo IPv6 en un nodo. El prefijo de estas direcciones es FE80::/10. Estas direcciones no pueden ser encaminadas a través de los Routers fuera del segmento local, de ahí deriva su nombre. El propósito principal es proporcionar direccionamiento IP automático a los nodos en caso que no exista un servidor DHCP.

Una dirección IPv6 Link-Local comienza con el prefijo FE80::/10 (los primeros 10 bits), luego los bits del 11 hasta 64 (los siguientes 54 bits) se configuran con valores de ceros (0000). De esta manera se forma la porción de red representada por los primeros 64bits. Ejemplo: FE80::211:21FF:FE6C:C86B.

Una dirección IPv6 Link-Local comienza con el prefijo FE80::/10 (los primeros 10 bits), luego los bits del 11 hasta 64 (los siguientes 54 bits) se configuran con valores de ceros (0000). De esta manera se forma la porción de red representada por los primeros 64bits. Ejemplo: FE80::211:21FF:FE6C:C86B

Las direcciones IPv6 Site-Local son también el equivalente a las direcciones IP privadas en IPv4. A diferencia de las direcciones Link-Local, estas pueden ser encaminadas fuera del segmento local, es decir, podemos enviar paquetes entre diferentes segmentos de la red pero no hacia el Internet. En las direcciones Site-Local, los primeros 10 bits se establecen con los valores 1111111011, por lo tanto, el prefijo de estas direcciones tendrá un valor en hexadecimal de FEC0 :: /10. Los siguientes 54 bits están compuestos por el ID de red. Los últimos 64 bits son el identificador de la interfaz o nodo, y estos se configuran de la misma forma que las direcciones Link-Local, tomando 48 bits de la dirección MAC y luego agregando 16 bits con los valores FFFE.

Ejemplo: FEC0::CE00:3BFF:FE85:0.

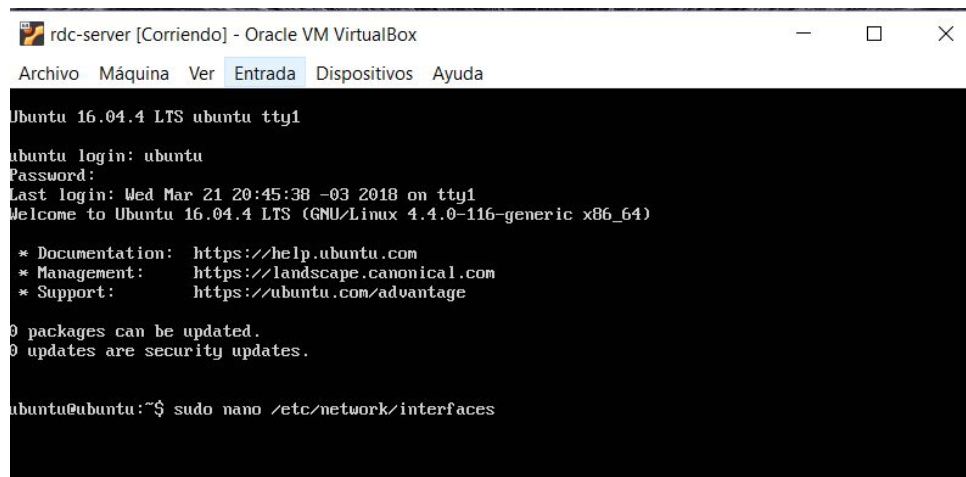
Las direcciones Global en IPv6 son el equivalente de las direcciones IP públicas en IPv4. Estas direcciones pueden ser ruteadas a través de la Internet. Los primeros 3 bits están compuestos por los valores 001 (en notación binaria), por lo tanto, el prefijo de estas direcciones IP tendrá un valor en hexadecimal de 2000 con una máscara /3. Las direcciones Global son el tipo de dirección IPv6 más utilizado.

## Ejercicio 2: Ruteo estático IPv4/IPv6 con Linux

### Consignas

#### 1.- Sobre el Router: Configurar de manera permanente las interfaces de red con las direcciones IP correspondientes.

Para realizar dicha configuración, se debe editar el archivo ubicado en `/etc/network/interfaces`, (en este caso con el editor *Nano*), y agregar en dicho archivo, las correspondientes direcciones IP que la consigna pide:



```
rdc-server [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda

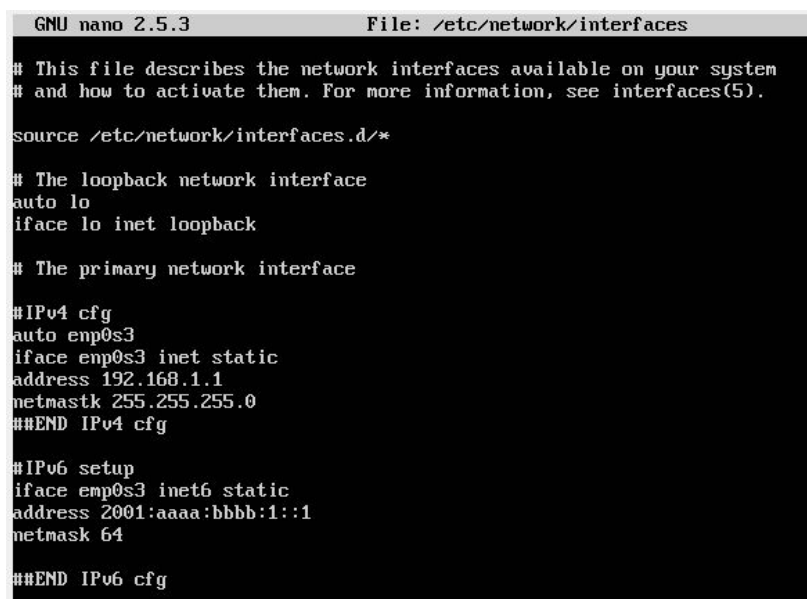
Ubuntu 16.04.4 LTS ubuntu tty1
ubuntu login: ubuntu
Password:
Last login: Wed Mar 21 20:45:38 -03 2018 on tty1
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-116-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

0 packages can be updated.
0 updates are security updates.

ubuntu@ubuntu:~$ sudo nano /etc/network/interfaces
```

Para la interfaz eth0 (enp0s3):



```
GNU nano 2.5.3      File: /etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface

#IPv4 cfg
auto enp0s3
iface enp0s3 inet static
address 192.168.1.1
netmask 255.255.255.0
##END IPv4 cfg

#IPv6 setup
iface enp0s3 inet6 static
address 2001:aaaa:bbbb:1::1
netmask 64
##END IPv6 cfg
```

Para eth1 (enp0s8):

```
# The secondary network interface
auto enp0s8

#IPv4 cfg
iface enp0s8 inet static
address 192.168.2.1
netmask 255.255.255.0
##END IPv4 cfg

##IPv6 cfg
iface enp0s8 inet6 static
address 2001:aaaa:dddd:1::1
netmask 64
##END IPv6 cfg
```

Editando el archivo mencionado, se configuran ambas interfaces de forma estática, con las correspondientes direcciones IP (IPv4 e IPv6).

El nombre asignado a la interfaz se obtuvo mediante los comandos *ip -a*, *sudo lshw -class network*.

## 2.- Sobre el Router: Configurar para que realice ip\_forwarding de manera permanente.

Para que el router pueda hacer ip forwarding de manera permanente, se debe modificar otro archivo ubicado en */etc/sysctl.conf*, descomentando las líneas correspondientes como indica el archivo (por defecto, las tiene comentadas).

```
GNU nano 2.5.3 File: /etc/sysctl.conf

# prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1

# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1

# Uncomment the next line to enable packet forwarding for IPv6
# Enabling this option disables Stateless Address Autoconfiguration
# based on Router Advertisements for this host
net.ipv6.conf.all.forwarding=1

#####
# Additional settings - these settings can improve the network
# net.ipv4.conf.all.rp_filter=1
# net.ipv4.conf.default.rp_filter=1
# net.ipv4.conf.all.arp_ignore=1
# net.ipv4.conf.default.arp_ignore=1
# net.ipv4.conf.all.arp_wanted=0
# net.ipv4.conf.default.arp_wanted=0
# net.ipv4.conf.all.arp_announce=2
# net.ipv4.conf.default.arp_announce=2
# net.ipv4.conf.all.route_hack=1
```

**3.- Sobre los Clientes: Utilizando la aplicación de configuración de red gráfica NetworkManager, asignar de manera permanente y las direcciones IPs correspondiente. Configurar como Default Gateway el Router que pertenezca a la misma red.**

Una vez configurado el Ubuntu server como router, desde el lado de los clientes, lo que se hizo fue re-configurar la dirección IP de la interfaz ethernet para poder realizar la conexión.

Editing Wired connection 1

Connection name: **Wired connection 1**

General Ethernet 802.1x Security DCB IPv4 Settings IPv6 Settings

Method: Manual

**Addresses**

Address	Netmask	Gateway
192.168.1.10	24	192.168.1.1

DNS servers:

Search domains:

DHCP client ID:

☐ Require IPv4 addressing for this connection to complete

Routes...

Editing Wired connection 1

Connection name: **Wired connection 1**

General Ethernet 802.1x Security DCB IPv4 Settings IPv6 Settings

Method: Manual

**Addresses**

Address	Prefix	Gateway
2001:aaaa:bbbb:1::10	64	2001:aaaa:bbbb:1::1

DNS servers:

Search domains:

IPv6 privacy extensions: Disabled

☐ Require IPv6 addressing for this connection to complete

Routes...



#### 4.- Sobre los Clientes: Con la configuración hecha hasta ahora. Ejecutar los siguientes tests y responder las siguientes preguntas

4.1.- Ping al Default gateway. Explicar el proceso de comunicación. Para IPv4: Protocolos ARP, IPv4 e ICMP. Para IPv6: Protocolos NDP, IPv6 e ICMPv6

```
ubuntu@ubuntu: ~  
ubuntu@ubuntu:~$ ping 192.168.1.1 -c 10  
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.  
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.267 ms  
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.309 ms  
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=0.291 ms  
64 bytes from 192.168.1.1: icmp_seq=4 ttl=64 time=0.302 ms  
64 bytes from 192.168.1.1: icmp_seq=5 ttl=64 time=0.309 ms  
64 bytes from 192.168.1.1: icmp_seq=6 ttl=64 time=0.425 ms  
64 bytes from 192.168.1.1: icmp_seq=7 ttl=64 time=0.284 ms  
64 bytes from 192.168.1.1: icmp_seq=8 ttl=64 time=0.312 ms  
64 bytes from 192.168.1.1: icmp_seq=9 ttl=64 time=0.317 ms  
64 bytes from 192.168.1.1: icmp_seq=10 ttl=64 time=0.240 ms  
  
--- 192.168.1.1 ping statistics ---  
10 packets transmitted, 10 received, 0% packet loss, time 9214ms  
rtt min/avg/max/mdev = 0.240/0.305/0.425/0.049 ms  
ubuntu@ubuntu:~$
```

```
rdc-desktop [Corriendo] - Oracle VM VirtualBox  
archivo Máquina Ver Entrada Dispositivos Ayuda  
terminal  
ubuntu@ubuntu: ~  
64 bytes from 2001:aaaa:bbbb:1::1: icmp_seq=1 ttl=64 time=0.693 ms  
64 bytes from 2001:aaaa:bbbb:1::1: icmp_seq=2 ttl=64 time=0.293 ms  
64 bytes from 2001:aaaa:bbbb:1::1: icmp_seq=3 ttl=64 time=0.169 ms  
^C  
--- 2001:aaaa:bbbb:1::1 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2039ms  
rtt min/avg/max/mdev = 0.169/0.385/0.693/0.223 ms  
ubuntu@ubuntu:~$ ping6 2001:aaaa:bbbb:1::1 -c 10  
PING 2001:aaaa:bbbb:1::1(2001:aaaa:bbbb:1::1) 56 data bytes  
64 bytes from 2001:aaaa:bbbb:1::1: icmp_seq=1 ttl=64 time=0.277 ms  
64 bytes from 2001:aaaa:bbbb:1::1: icmp_seq=2 ttl=64 time=0.295 ms  
64 bytes from 2001:aaaa:bbbb:1::1: icmp_seq=3 ttl=64 time=0.287 ms  
64 bytes from 2001:aaaa:bbbb:1::1: icmp_seq=4 ttl=64 time=0.283 ms  
64 bytes from 2001:aaaa:bbbb:1::1: icmp_seq=5 ttl=64 time=0.294 ms  
64 bytes from 2001:aaaa:bbbb:1::1: icmp_seq=6 ttl=64 time=0.293 ms  
64 bytes from 2001:aaaa:bbbb:1::1: icmp_seq=7 ttl=64 time=0.290 ms  
64 bytes from 2001:aaaa:bbbb:1::1: icmp_seq=8 ttl=64 time=0.283 ms  
64 bytes from 2001:aaaa:bbbb:1::1: icmp_seq=9 ttl=64 time=0.288 ms  
64 bytes from 2001:aaaa:bbbb:1::1: icmp_seq=10 ttl=64 time=0.290 ms  
  
--- 2001:aaaa:bbbb:1::1 ping statistics ---  
10 packets transmitted, 10 received, 0% packet loss, time 9211ms  
rtt min/avg/max/mdev = 0.277/0.288/0.295/0.005 ms  
ubuntu@ubuntu:~$
```

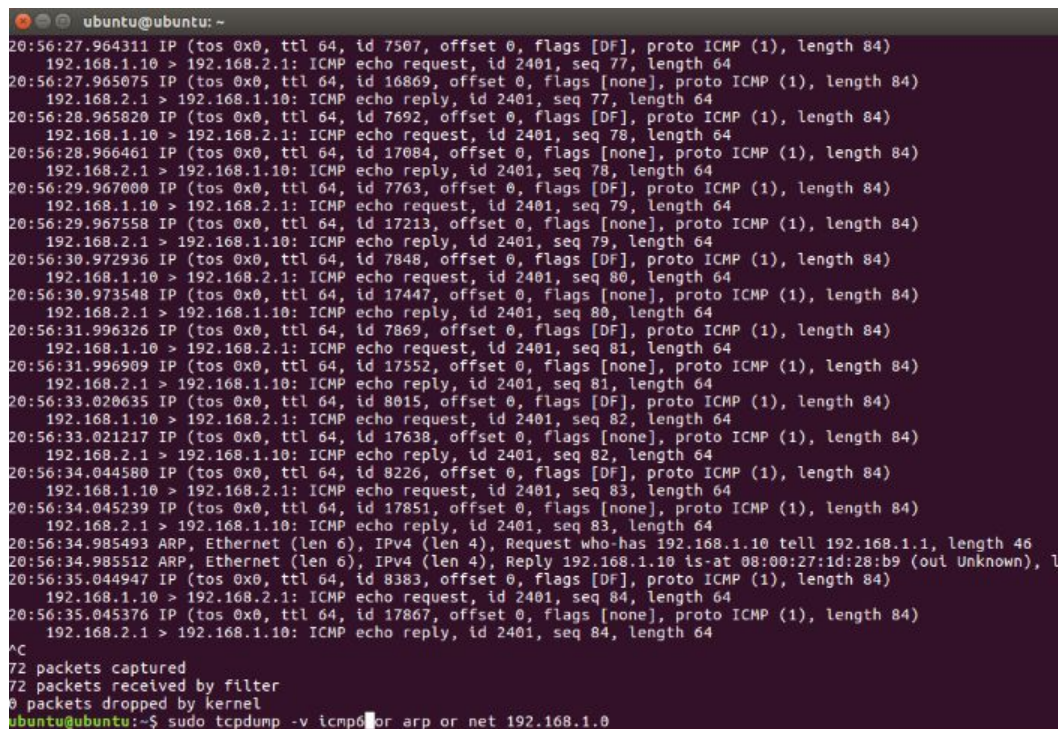
cap4.png



La siguiente secuencia responde a lo que sucede al realizar ping a nivel protocolo IPv4:

- Se envía un paquete ARP Request, cuya fuente (source) es Cliente1, y tiene como destino la dirección de broadcast
- Se envía un ARP Reply desde la interfaz del router (192.168.1.1), en respuesta al mensaje anterior, cuyo destino es Cliente1.
- Desde Cliente1, se envía un ICMP Request hacia la dirección de default gateway
- Desde la interfaz del router que se halla en la misma red que Cliente1 (default gateway), se envía un ICMP Reply con destino a Cliente1.

Se fue realizando el análisis del tráfico utilizando el comando *tcpdump*



```
ubuntu@ubuntu: ~  
20:56:27.964311 IP (tos 0x0, ttl 64, id 7507, offset 0, flags [DF], proto ICMP (1), length 84)  
192.168.1.10 > 192.168.2.1: ICMP echo request, id 2401, seq 77, length 64  
20:56:27.965075 IP (tos 0x0, ttl 64, id 16869, offset 0, flags [none], proto ICMP (1), length 84)  
192.168.2.1 > 192.168.1.10: ICMP echo reply, id 2401, seq 77, length 64  
20:56:28.965820 IP (tos 0x0, ttl 64, id 7692, offset 0, flags [DF], proto ICMP (1), length 84)  
192.168.1.10 > 192.168.2.1: ICMP echo request, id 2401, seq 78, length 64  
20:56:28.966461 IP (tos 0x0, ttl 64, id 17084, offset 0, flags [none], proto ICMP (1), length 84)  
192.168.2.1 > 192.168.1.10: ICMP echo reply, id 2401, seq 78, length 64  
20:56:29.967000 IP (tos 0x0, ttl 64, id 7763, offset 0, flags [DF], proto ICMP (1), length 84)  
192.168.1.10 > 192.168.2.1: ICMP echo request, id 2401, seq 79, length 64  
20:56:29.967558 IP (tos 0x0, ttl 64, id 17213, offset 0, flags [none], proto ICMP (1), length 84)  
192.168.2.1 > 192.168.1.10: ICMP echo reply, id 2401, seq 79, length 64  
20:56:30.972936 IP (tos 0x0, ttl 64, id 7848, offset 0, flags [DF], proto ICMP (1), length 84)  
192.168.1.10 > 192.168.2.1: ICMP echo request, id 2401, seq 80, length 64  
20:56:30.973548 IP (tos 0x0, ttl 64, id 17447, offset 0, flags [none], proto ICMP (1), length 84)  
192.168.2.1 > 192.168.1.10: ICMP echo reply, id 2401, seq 80, length 64  
20:56:31.996326 IP (tos 0x0, ttl 64, id 7869, offset 0, flags [DF], proto ICMP (1), length 84)  
192.168.1.10 > 192.168.2.1: ICMP echo request, id 2401, seq 81, length 64  
20:56:31.996909 IP (tos 0x0, ttl 64, id 17552, offset 0, flags [none], proto ICMP (1), length 84)  
192.168.2.1 > 192.168.1.10: ICMP echo reply, id 2401, seq 81, length 64  
20:56:33.020635 IP (tos 0x0, ttl 64, id 8015, offset 0, flags [DF], proto ICMP (1), length 84)  
192.168.1.10 > 192.168.2.1: ICMP echo request, id 2401, seq 82, length 64  
20:56:33.021217 IP (tos 0x0, ttl 64, id 17638, offset 0, flags [none], proto ICMP (1), length 84)  
192.168.2.1 > 192.168.1.10: ICMP echo reply, id 2401, seq 82, length 64  
20:56:34.044580 IP (tos 0x0, ttl 64, id 8226, offset 0, flags [DF], proto ICMP (1), length 84)  
192.168.1.10 > 192.168.2.1: ICMP echo request, id 2401, seq 83, length 64  
20:56:34.045239 IP (tos 0x0, ttl 64, id 17851, offset 0, flags [none], proto ICMP (1), length 84)  
192.168.2.1 > 192.168.1.10: ICMP echo reply, id 2401, seq 83, length 64  
20:56:34.985493 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.1.10 tell 192.168.1.1, length 46  
20:56:34.985512 ARP, Ethernet (len 6), IPv4 (len 4), Reply 192.168.1.10 is-at 08:00:27:1d:28:b9 (out Unknown), length 46  
20:56:35.044947 IP (tos 0x0, ttl 64, id 8383, offset 0, flags [DF], proto ICMP (1), length 84)  
192.168.1.10 > 192.168.2.1: ICMP echo request, id 2401, seq 84, length 64  
20:56:35.045376 IP (tos 0x0, ttl 64, id 17867, offset 0, flags [none], proto ICMP (1), length 84)  
192.168.2.1 > 192.168.1.10: ICMP echo reply, id 2401, seq 84, length 64  
^C  
72 packets captured  
72 packets received by filter  
0 packets dropped by kernel  
ubuntu@ubuntu:~$ sudo tcpdump -v icmp or arp or net 192.168.1.0
```

En lo que refiere a IPv6, lo que sucede es lo siguiente:

- Neighbor solicitation con origen en el Cliente1 y como destino la interfaz del router(default gateway).
- NA (neighbor advertisement) con destino en el Cliente1.
- Se envía un mensaje ICMP6 request desde el Cliente1 hacia el default gateway.
- Ahora se envía un ICMP6 reply con origen en la interfaz del router (default gateway) cuyo destino es Cliente1

**4.2.- Ping a el otro Cliente. Explicar el proceso de comunicación. Para IPv4: Protocolos ARP, IPv4 e ICMP. Para IPv6: Protocolos NDP, IPv6 e ICMPv6**

```
ubuntu@ubuntu: ~  
--- 192.168.1.1 ping statistics ---  
10 packets transmitted, 10 received, 0% packet loss, time 9203  
ms  
rtt min/avg/max/mdev = 0.261/0.620/0.786/0.182 ms  
ubuntu@ubuntu:~$ ping 192.168.2.1 -c 3  
PING 192.168.2.1 (192.168.2.1) 56(84) bytes of data.  
64 bytes from 192.168.2.1: icmp_seq=1 ttl=64 time=0.276 ms  
64 bytes from 192.168.2.1: icmp_seq=2 ttl=64 time=0.691 ms  
64 bytes from 192.168.2.1: icmp_seq=3 ttl=64 time=0.753 ms  
--- 192.168.2.1 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2038ms  
rtt min/avg/max/mdev = 0.276/0.573/0.753/0.212 ms  
ubuntu@ubuntu:~$ ping 192.168.2.10 -c 3  
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=2.04 ms  
64 bytes from 192.168.2.10: icmp_seq=2 ttl=63 time=1.84 ms  
64 bytes from 192.168.2.10: icmp_seq=3 ttl=63 time=1.37 ms  
--- 192.168.2.10 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2002ms  
rtt min/avg/max/mdev = 1.372/1.753/2.040/0.282 ms  
ubuntu@ubuntu:~$
```

Cuando se realiza ping desde un cliente hacia el otro, sucede la siguiente secuencia:

- 1- ARP Request, origen en el Cliente1 y destino FF:FF:FF:FF (dirección de broadcast)
- 2- ARP Reply, origen en la interfaz del Router que pertenece a la misma red que el Cliente1 y destino en el Cliente1.
- 3- ICMP request, origen en el Cliente1 y destino en la Default Gateway.
- 4- ARP Request, origen en la interfaz del Router que pertenece a la red de Cliente2 (192.168.2.0) y destino a la dirección de broadcast FF:FF:FF:FF.
- 5- ARP Reply con origen en el Cliente2 (respuesta al ARP Request enviado a broadcast), y destino en la interfaz del Router que pertenece a la misma red que el Cliente2 (o default gateway de Cliente2).
- 6- ICMP reply con origen en el Cliente2 y destino en el Cliente1 (respondiendo a ítem 3, y comunicando ambos clientes)

## Ejercicio 3: Tecnología: namespaces

### 1.1.- ¿Qué es Linux Namespace? ¿Cómo funciona?

Los contenedores (containers), son una tecnología que, partiendo de la idea base de chroot, extiende este concepto para conseguir ejecutar entornos aislados del sistema.

Con el uso de contenedores, en realidad no se está virtualizando nada sino que se están manteniendo en un espacio aislado (namespace) los procesos y ficheros necesarios mientras se reutiliza el kernel del sistema anfitrión.



*Contenedores vs virtualización*

Los *namespaces* constituyen el elemento base de los contenedores y es una funcionalidad del kernel que proporcionan facilidades para crear una abstracción del sistema de modo que, todo lo que sucede fuera del espacio del contenedor sea invisible al interior.

La evolución de los namespaces ha permitido pasar del “enjaulamiento” de rutas de *chroot* hasta los contenedores que proporcionan espacios aislados en todos los niveles: espacio de usuario, espacio de procesos, de red, puntos de montaje, etc.

En lo que refiere a redes, Network namespaces virtualizan la network stack (pila de red). Cuando se crea un namespace de red, contiene únicamente una interfaz loopback. Cada interfaz de red, ya sea física o virtual, esta presente en exactamente un namespace, y la misma puede ser trasladada a otro. Cada namespace tendrá un set privado de direcciones IP, junto con su propia tabla de ruteo, lista de sockets, tabla de seguimiento de conexiones, y otros recursos de redes.

Si se destruye un namespace de red, destruirá cualquier interfaz virtual que lo involucré, y moverá aquellas interfaces físicas a su namespace inicial.

### 1.2.- Linux Bridge, ¿A qué dispositivo de red emula? ¿Por qué?

Linux Bridge equivale a un switch “virtual”, es decir, emula un switch de red con su función tal cual la conocemos. Es un módulo de kernel, que incluye aún más funcionalidades, como por ejemplo, el filtrado de tráfico.

Principalmente permiten conectividad entre máquinas virtuales, contenedores y namespaces, además de proveer la conexión de las mismas hacia el exterior.

Linux bridge es un dispositivo virtual de capa 2, que por sí mismo no puede recibir o transmitir paquetes salvo que se le asignen uno o más dispositivos reales.

### **1.3.- ¿Qué es el “veth pair”? 1.4.- ¿Es capaz de crear dos namespaces y conectarlos entre sí?**

Veth refiere a Virtual Ethernet Pair, y es comúnmente usado cuando se tratan de conectar dos entidades que se linkean a una interfaz ya sea para reenviar o recibir frames.

Estas entidades pueden ser namespaces, containers, entre otros.

Las interfaces veth vienen en pares, y se conectan como tuberías (lo que sale en una veth interface, saldrá también en su par). Es de mera utilidad la existencia de las mismas para conectar un network namespace a internet, ya sea por su “default” o “global” namespace donde interfaces físicas existen.

Se podrían imaginar como un túnel que trabaja a nivel capa de enlace, y que se ve como un par de dispositivos interconectados entre sí por ethernet.

### **1.5.- ¿Puede determinar si el software CORE usa linux namespaces. Cómo puede determinar eso?**

El software CORE si usa namespaces. Buscando en la documentación, lo aclara:

“Linux network namespaces (also known as netns, LXC, or [Linux containers](#)) is the primary virtualization technique used by CORE. LXC has been part of the mainline Linux kernel since 2.6.24. Recent Linux distributions such as Fedora and Ubuntu have namespaces-enabled kernels out of the box, so the kernel does not need to be patched or recompiled. A namespace is created using the `clone()` system call. Similar to the BSD jails, each namespace has its own process environment and private network stack. Network namespaces share the same filesystem in CORE. CORE combines these namespaces with Linux Ethernet bridging to form networks. Link characteristics are applied using Linux Netem queuing disciplines. Ebtables is Ethernet frame filtering on Linux bridges. Wireless networks are emulated by controlling which interfaces can send and receive with ebtables rules.”

Hay comandos para listar los namespaces activos, tratando de correrlos y analizarlos no se pudo identificar los de CORE. Suponemos que esto se debe a algunas funcionalidades internas, pero leyendo la documentación si pudimos afirmar que utiliza namespaces.

## **Bibliografía.**

*Redes de Computadoras - 5ta Edición - James F. Kurose & Keith W. Ross*

<https://www.1and1.es/digitalguide/servidores/know-how/en-que-consiste-el-neighbor-discovery-protocol/>

<https://docs.oracle.com/cd/E19082-01/819-3000/chapter1-41/index.html>

<https://www.somosbinarios.es/que-es-multicast/>

<http://www.brianlinkletter.com/ipv6-addressing-simulator-part-1/>

<http://www.ducea.com/2006/08/01/how-to-enable-ip-forwarding-in-linux/>

[https://es.wikipedia.org/wiki/Direcci%C3%B3n\\_IPv6](https://es.wikipedia.org/wiki/Direcci%C3%B3n_IPv6)

<https://learningnetwork.cisco.com/thread/37717>

<https://docs.oracle.com/cd/E19082-01/819-3000/chapter1-41/index.html>

<https://www.1and1.es/digitalguide/servidores/know-how/en-que-consiste-el-neighbor-discovery-protocol/>

<http://opensourceforu.com/2015/04/how-to-configure-ubuntu-as-a-router/>

*Contenedores Linux y seguridad. Docker*

<https://securityinside.info/contenedores-linux-seguridad/>

<https://goyalankit.com/blog/linux-bridge>

<https://lwn.net/Articles/232688/>