

Redes de Computadoras 2021

TP1: Análisis de tráfico IPv6 en capa 3

Estudiantes:

Germán Lamberti (IComp - 38279109 - german.lamberti@unc.edu.ar)

Esteban Perez (IComp - 39026980 - esteban.perez@mi.unc.edu.ar)

8 de Abril, 2021

Objetivos

Configuración de dual stack(IPv4 e IPv6) en hosts usando el emulador CORE. Análisis de tráfico, comportamiento de *ARP*, *NDP* e *ICMP*. Asignación de direcciones de forma dinámica usando DHCP.

Requisitos

- Computadora por cada 2 personas

Consignas

Tráfico IPv4 e IPv6 con CORE

Recomendaciones

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

Diagrama de red

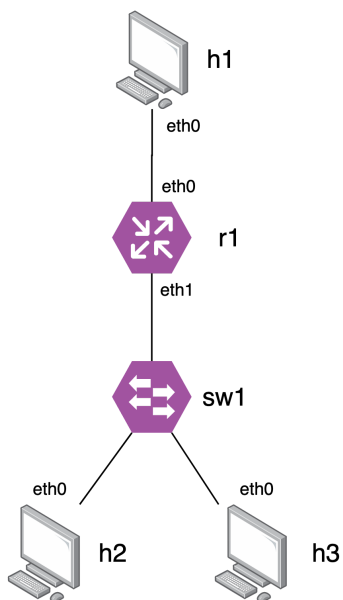


Tabla de asignación de direcciones IPv4 e IPv6

| Computadora | Interfaz de red | Dirección IP |
|-------------|-----------------|-------------------------------|
| h1 | eth0 | IPv4: 192.168.1.10/24 |
| | | IPv6: 2001:aaaa:bbbb:1::10/64 |
| h2 | eth0 | IPv4: 192.168.2.10/24 |
| | | IPv6: 2001:aaaa:cccc:1::10/64 |
| h3 | eth0 | IPv4: 192.168.2.11/24 |
| | | IPv6: 2001:aaaa:cccc:1::11/64 |
| r1 | eth0 | IPv4: 192.168.1.11/24 |
| | | IPv6: 2001:aaaa:bbbb:1::11/64 |
| | eth1 | IPv4: 192.168.2.12/24 |
| | | IPv6: 2001:aaaa:cccc:1::12/64 |

Consignas

1. Crear el esquema de red sobre el software de emulación CORE. ¿Cuál es la diferencia entre un simulador y un emulador? ¿Por qué CORE es considerado un emulador? ¿Conoce algún simulador en el área de redes?
2. Probar conectividad entre todos los hosts enviando 3 paquetes ICMPv4 usando el comando “ping” para IPv4.
3. Probar conectividad entre todos los hosts enviando 3 paquetes ICMPv6 usando el comando “ping6” para IPv6.
4. Iniciar tráfico ICMP en el Cliente1 con destino Cliente2. Analizar tráfico con “tcpdump” sobre las dos redes, capturar screenshots y responder las siguientes preguntas:
 - a. ¿Cuáles son las comunicaciones ARP que suceden? Ejemplifica brevemente y con capturas cómo funciona la traducción de direcciones lógicas a direcciones físicas.
 - b. ¿Cuáles son las direcciones IPs en los datagramas IPs?
 - c. ¿Cómo sabe el router cómo comunicar un host con otro host?

- d. ¿Para qué usamos el switch? ¿Por que el switch no tiene asignadas direcciones IP en sus interfaces ?
 - e. ¿Qué datos contiene la tabla ARP de h1?
 - f. ¿Qué datos contiene la tabla ARP de h3?
 - g. ¿Qué datos contiene la tabla ARP del router?
 - h. ¿Qué son las direcciones de broadcast en IPv4? Cual es su utilidad?
 - i. ¿Qué son las direcciones de multicast en IPv4? Cual es su utilidad?
5. Iniciar tráfico ICMPv3 (IPv6) entre h1 y h3. Analizar el tráfico con “tcpdump” sobre las dos redes, capturar screenshots y responder a las siguientes preguntas:
- 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.
 - b. NDP reemplaza a ARP?
 - c. Describa todas las funciones de NDP
 - d. ¿Existen direcciones de broadcast en IPv6? Como se reemplaza esta funcionalidad en IPv6?
 - e. ¿Cuál es la diferencia entre las direcciones link-local, unique-local, global? Ejemplificar. En qué caso usaría a cada una ?

Links de ayuda

- [core/install.md at master · coreemu/core](#)
- Capítulos 6 y 22 Comer

Autoconfiguración de direcciones IPv4 en linux namespaces

Recomendaciones

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

Diagrama de red

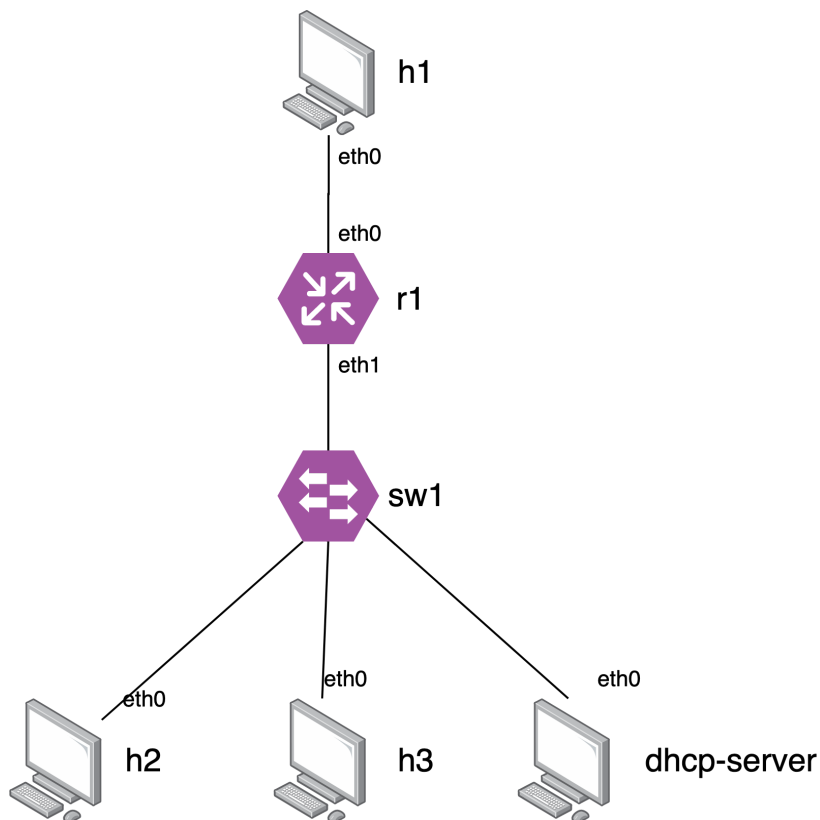


Tabla de asignación de direcciones IPv4

| Computadora | Interfaz de red | Direccion IP |
|--------------------|-----------------|-----------------------|
| h1 | eth0 | IPv4: 192.168.1.10/24 |
| dhcp-server | eth0 | Pool: 192.168.2.0/24 |
| r1 | eth0 | IPv4: 192.168.1.11/24 |
| | eth1 | IPv4: 192.168.2.12/24 |

Consignas

1. Con linux namespaces defina la topología que se muestra en el diagrama. Ayuda: puede usar el script Gist que usamos en clase como base ya que la topología es la misma solo se agrega un host a la subnet de abajo
(<https://gist.github.com/natitomattis/be26889063203c0b33b33fa25c75a5b6>)
2. Configurar un dhcp server en el nuevo host, asegurarse que no entregue la IP del router que se asignan estáticamente.
3. Usando el comando *dhclient* configurar dinámicamente la IP de h2 y h3. Qué direcciones se les asignaron?
4. Explique brevemente y con capturas (tcpdump o wireshark) cómo funciona DHCP y los mensajes que intervienen.
5. Hay conectividad entre h1 y el resto de los hosts ? Por qué ? Por qué con IPv6 no tuvimos este inconveniente ? Realice las configuraciones necesarias para que funcione el ping entre h1 y el resto de los hosts.

Links de ayuda

- [DHCP configuration file /etc/dhcp/dhcpd.conf explained](#)
- Capítulo 22 Comer

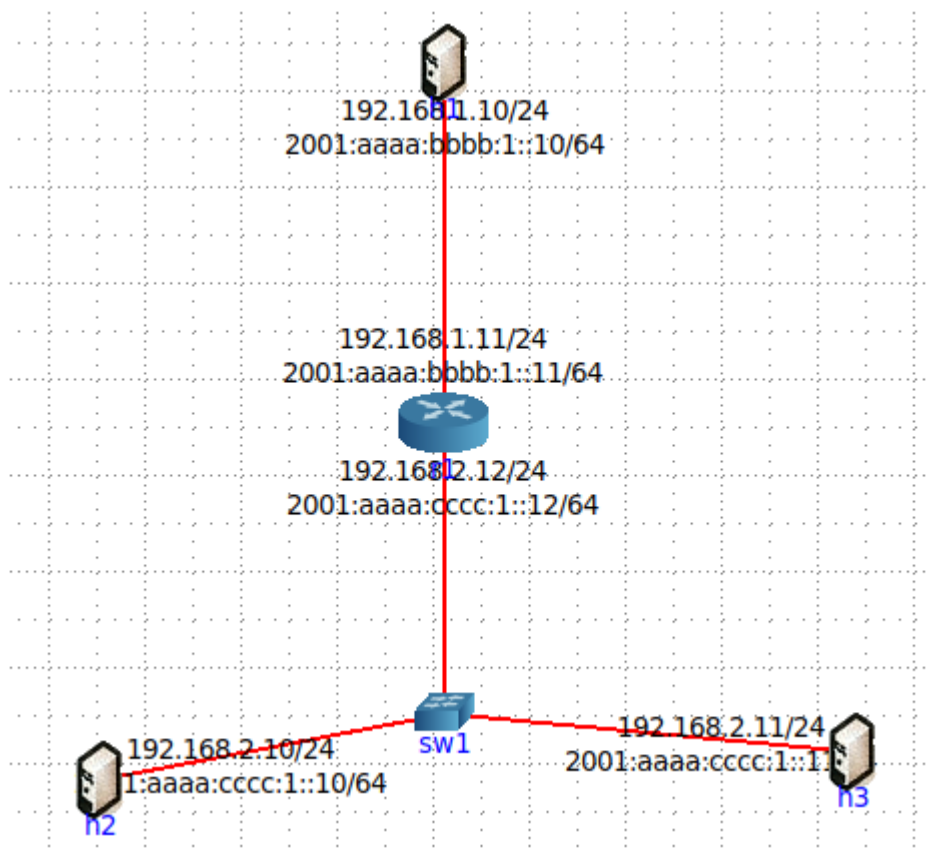
Parte 1:

Introducción

A los fines de realizar el TP1, se procedió a realizar la siguiente topología requerida en CORE network emulator, configurando las direcciones IPv4 e IPv6 tanto en los host como en el router, para posteriormente realizar el ping IPv4 e IPv6 entre los distintos host.

Desarrollo

Se llevo a cabo la siguiente topología en el emulador antes mencionado y se configuraron las direcciones IPv4 e IPv6 en cada host y router.



Posteriormente se configuraron los default gateway del router:

```
interface eth0
  ip address 192.168.1.11/24
  ipv6 address 2001:aaaa:bbbb:1::11/64
  no ipv6 nd suppress-ra
  ipv6 nd ra-interval-50
!
2 interface eth1
:   ip address 192.168.2.12/24
:   ipv6 address 2001:aaaa:cccc:1::12/64
:   no ipv6 nd suppress-ra
:   ipv6 nd ra-interval-50
!
```

Configuración del Default Gateway del Host 1:

```
#!/bin/sh
# auto-generated by DefaultRoute service (utility.py)
ip route add default via 192.168.1.11
```

Y de los Host 2 y 3:

```
#!/bin/sh
# auto-generated by DefaultRoute service (utility.py)
ip route add default via 192.168.2.12
```

1)

La diferencia entre un simulador y un emulador es que el simulador sólo trata de reproducir el comportamiento de un programa, sistema determinado, mientras que un emulador trata de modelar de forma precisa el dispositivo de manera que este funcione como si estuviese siendo usado en el artefacto original. Por citar unos ejemplos interdisciplinarios, un simulador podría ser un programa que imita la caída y flujo del agua de lluvias y un ejemplo de emulador podría ser un software que reproduzca los videojuegos arcade en una PC.

Yendo al tema que abordamos en este trabajo práctico número uno, CORE network emulador es un emulador dado a que nos permite la creación de instancias rápidas de topologías híbridas compuestas de hardware real y nodos de red virtual, puede conectar estas redes emuladas a redes en vivo.

Simuladores de redes:

- Realistic And Large Network Simulator (Real)
- Network Simulator Tesbed (NEST)
- Maryland Routing Simulator (MaRS)

2) Ping para IPv4 entre host1 y host2:

```
root@h1:/tmp/pycore.37899/h1.conf# ping 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=0.102 ms
64 bytes from 192.168.2.10: icmp_seq=2 ttl=63 time=0.081 ms
64 bytes from 192.168.2.10: icmp_seq=3 ttl=63 time=0.085 ms
64 bytes from 192.168.2.10: icmp_seq=4 ttl=63 time=0.081 ms
64 bytes from 192.168.2.10: icmp_seq=5 ttl=63 time=0.082 ms
64 bytes from 192.168.2.10: icmp_seq=6 ttl=63 time=0.061 ms
```

Ping para IPv4 entre host2 y host3:

```
root@h2:/tmp/pycore.37899/h2.conf# ping 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=0.151 ms
64 bytes from 192.168.2.11: icmp_seq=2 ttl=64 time=0.070 ms
64 bytes from 192.168.2.11: icmp_seq=3 ttl=64 time=0.067 ms
64 bytes from 192.168.2.11: icmp_seq=4 ttl=64 time=0.057 ms
64 bytes from 192.168.2.11: icmp_seq=5 ttl=64 time=0.051 ms
64 bytes from 192.168.2.11: icmp_seq=6 ttl=64 time=0.049 ms
```

Ping para IPv4 entre host1 y host3:

```
root@h1:/tmp/pycore.45357/h1.conf# ping 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=0.222 ms
64 bytes from 192.168.2.11: icmp_seq=2 ttl=63 time=0.087 ms
64 bytes from 192.168.2.11: icmp_seq=3 ttl=63 time=0.090 ms
64 bytes from 192.168.2.11: icmp_seq=4 ttl=63 time=0.090 ms
64 bytes from 192.168.2.11: icmp_seq=5 ttl=63 time=0.091 ms
64 bytes from 192.168.2.11: icmp_seq=6 ttl=63 time=0.087 ms
```

3) Ping6 para IPv6 entre el host1 y los host2 y host3:

```
root@h1:/tmp/pycore.34195/h1.conf# ping 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=0.135 ms
64 bytes from 2001:aaaa:cccc:1::10: icmp_seq=2 ttl=63 time=0.099 ms
64 bytes from 2001:aaaa:cccc:1::10: icmp_seq=3 ttl=63 time=0.079 ms
64 bytes from 2001:aaaa:cccc:1::10: icmp_seq=4 ttl=63 time=0.071 ms
^Z
[1]+  Detenido                  ping 2001:aaaa:cccc:1::10
root@h1:/tmp/pycore.34195/h1.conf# ping 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=0.208 ms
64 bytes from 2001:aaaa:cccc:1::11: icmp_seq=2 ttl=63 time=0.131 ms
64 bytes from 2001:aaaa:cccc:1::11: icmp_seq=3 ttl=63 time=0.097 ms
64 bytes from 2001:aaaa:cccc:1::11: icmp_seq=4 ttl=63 time=0.099 ms
```

Ping IPv6 entre host2 y host3

```
root@h2:/tmp/pycore.45357/h2.conf# ping 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=64 time=0.152 ms
64 bytes from 2001:aaaa:cccc:1::11: icmp_seq=2 ttl=64 time=0.076 ms
64 bytes from 2001:aaaa:cccc:1::11: icmp_seq=3 ttl=64 time=0.072 ms
64 bytes from 2001:aaaa:cccc:1::11: icmp_seq=4 ttl=64 time=0.041 ms
```

4. a)

Comunicaciones ARP: como podemos observar en la captura, el host 1 desea enviar un paquete ICMP hacia el host 2, pero este host no se encuentra dentro de la misma subred , entonces el encargado de hacer llegar este paquete es el router.

Aca podemos ver el TCPDUMP de eth0 del Host 1:

```
03:48:34.720312 ARP, Request who-has 192.168.1.11 tell 192.168.1.10, length 28
03:48:34.720378 ARP, Reply 192.168.1.11 is-at 00:00:00:aa:00:01, length 28
03:48:34.720386 IP 192.168.1.10 > 192.168.2.10: ICMP echo request, id 36, seq 1, length 64
03:48:34.720482 IP 192.168.2.10 > 192.168.1.10: ICMP echo reply, id 36, seq 1, length 64
03:48:35.722071 IP 192.168.1.10 > 192.168.2.10: ICMP echo request, id 36, seq 2, length 64
03:48:35.722147 IP 192.168.2.10 > 192.168.1.10: ICMP echo reply, id 36, seq 2, length 64
03:48:39.780169 ARP, Request who-has 192.168.1.10 tell 192.168.1.11, length 28
03:48:39.780182 ARP, Reply 192.168.1.10 is-at 00:00:00:aa:00:00, length 28
```

El host 1 a través de un mensaje ARP broadcast de petición, consulta por la dirección física del default gateway del router a la cual está conectado, el cual responde con un mensaje unicast. Al recibir esta respuesta se produce la carga en las tablas ARP de ambos dispositivos de la dirección MAC asociada a la dirección IP de cada dispositivo. Es valido mencionar que Host 1 no conoce la existencia del default gateway, este asume que es el host 2. El router será quien debe determinar por cual puerta enviar el paquete proveniente del host 1, una vez hecho esto, como no conoce la dirección MAC del host 2, envía el paquete al switch para que este a través de un mensaje ARP broadcast conocer la dirección física del host 2, el cual va a responder a este mensaje con su dirección física.

4. b)

Las direcciones IP en los datagramas IP request son :

- Dirección IPv4 de host 1: 192.168.1.10
- Dirección IPv4 de host2 : 192.168.2.10

Las direcciones IP en los datagramas IP replay son :

- Dirección IPv4 de host 2: 192.168.2.10
- Dirección IPv4 de host 1: 192.168.1.10

4. c)

El router recibe el paquete que desea enviar el host 1 hacia el host 2, el router divide la dirección de red de destino que tiene el paquete y determinar por cual default gateway debe redireccionarlo. Una vez hecho esto, en el caso que el router dentro de su tabla ARP no contenga una dirección MAC asociada a la IP de destino del paquete, este envía un mensaje al switch el cual envía un broadcast en busca del host de destino, una vez que recibe la respuesta, se dispone a enviar el paquete a la dirección MAC proporcionada y a guardar esta dirección en su tabla ARP.

4. d)

El Switch es un dispositivo de capa de Enlace (capa 2 en modelo OSI) por ende no tiene asignadas direcciones, por eso utilizamos el Switch para conectar varios dispositivos entre sí, formando así una subred o enlace local, en la cual a los dispositivos les es posible comunicarse entre sí, sin la necesidad de un router.

4. e)

Tabla ARP Host 1

```
root@h1:/tmp/pycore.46049/h1.conf# arp
```

| Dirección | TipoHW | DirecciónHW | Indic | Máscara | Interfaz |
|--------------|--------|-------------------|-------|---------|----------|
| 192.168.1.11 | ether | 00:00:00:aa:00:01 | C | | eth0 |

Tabla ARP Host 3

```
root@h3:/tmp/pycore.46049/h3.conf# arp
root@h3:/tmp/pycore.46049/h3.conf#
```

Esta tabla no está completa, debido a que no se le hizo ping a este host, por lo tanto aún no completo su tabla ARP; cuando la complete, tendrá la dirección IPv4 del default gateway del router 1 al cual está conectado y la dirección MAC del mismo.

4. g)

Tabla ARP Router 1

| Dirección | TipoHW | DirecciónHW | Indic | Máscara | Interfaz |
|--------------|--------|-------------------|-------|---------|----------|
| 192.168.1.10 | ether | 00:00:00:aa:00:00 | C | | eth0 |
| 192.168.2.10 | ether | 00:00:00:aa:00:04 | C | | eth1 |

Aquí se pueden ver las direcciones IPv4 del host 2 y del host 1 respectivamente y a cual interfaz de red (eth1 y eth0 respectivamente) están conectadas.

5. h)

Las redes o subredes tienen su propia dirección de broadcast, mediante la cual todos los participantes de la red pueden enviar datos. La dirección de broadcast es aquella en la que todos los bits del host están en valor binario 1. La utilidad radica en que nos permite enviar mensajes a todos los hosts de la red, esto suele suceder cuando las direcciones individuales de los miembros de la red son desconocidas. Otra ventaja es que puede distribuirse de forma masiva sin tener que enviarla en más de una ocasión.

Una dirección IP de grupo multicast se asigna a los dispositivos que pertenecen a un grupo multicast. Debido a que las direcciones multicast representan un grupo de direcciones (un grupo de hosts), sólo pueden utilizarse como el destino de un paquete. Son muy útiles ya que permiten a un dispositivo de origen enviar un paquete directamente a un grupo de dispositivos.

El rango de direcciones IPv4 multicast va de 224.0.0.0 a 239.255.255.255.

5. a)

Las diferentes **Comunicaciones NDP** que sucede son: un Router Solicitation (RS) , la cual es del tipo multicast, ya que desde un único dispositivo en este caso el host 1 se envía a todos los routers de la dirección multicast correspondiente (El destino del mensaje es el rango de direcciones multicast estándar “ff02::01”); Una vez encontrado un router este realiza el envío de un mensaje llamado Router Advertisement (RA) que también es del tipo multicast hacia todos los dispositivos de la dirección multicast correspondiente, o en algunos casos puede ser unicast dirigido exclusivamente al host solicitante.

Luego como se puede ver en la imagen podemos ver que hay un nuevo mensaje llamado *Neighbor Solicitation (NS)*, en el cual si los hosts quieren encontrar una dirección MAC de un vecino, pueden transmitir este tipo de mensajes ICMPv6 por multicast o por unicast si solo quieren comprobar la disponibilidad del vecino, en nuestro caso es un mensaje unicast link-local con direcciones de origen fe80::200:ff:feaa:1 y destino fe80::7cde:19ff:fe7f:458e, luego tenemos un mensaje como respuesta, llamado *Neighbor Advertisement (NA)* el cual el host que recibió el NS responde al host solicitante con un mensaje tipo unicast link-local, con direcciones de origen fe80::7cde:19ff:fe7f:458e y destino fe80::200:ff:feaa:1 .

Por último, no lo vemos en la captura pero hay otro tipo de mensaje llamado Redirect (R) el cual es enviado desde un router para informar a los hosts de un mejor primer salto para un destino.

```
04:10:04.899845 IP6 fe80::f0fc:50ff:fe24:e256 > ff02::2: ICMP6, router solicitation, length 16
04:10:04.900134 IP6 fe80::200:ff:feaa:1 > ff02::1: ICMP6, router advertisement, length 24
```

```
04:10:06.691364 IP6 fe80::200:ff:feaa:1 > fe80::7cde:19ff:fe7f:458e: ICMP6, neighbor solicitation, w
ho has fe80::7cde:19ff:fe7f:458e, length 32
04:10:06.691670 IP6 fe80::7cde:19ff:fe7f:458e > fe80::200:ff:feaa:1: ICMP6, neighbor solicitation, w
ho has fe80::200:ff:feaa:1, length 32
04:10:06.691689 IP6 fe80::200:ff:feaa:1 > fe80::7cde:19ff:fe7f:458e: ICMP6, neighbor advertisement,
tgt is fe80::200:ff:feaa:1, length 24
04:10:06.691693 IP6 fe80::7cde:19ff:fe7f:458e > fe80::200:ff:feaa:1: ICMP6, neighbor advertisement,
tgt is fe80::7cde:19ff:fe7f:458e, length 24
```

5. b)

En Capa de Red (Capa 3) tenemos 2 protocolos que son de gran importancia, uno para IPv6 como NDP y otro para IPv4: ARP. El primero no es un reemplazo de ARP, es un protocolo mucho más completo.

El protocolo de resolución de direcciones (ARP) es responsable de encontrar la dirección de hardware (Ethernet MAC) que corresponde a una determinada dirección IP mediante el envío de mensajes ARP request y como respuesta se obtiene un ARP reply ; mientras que NDP puede enviar varios tipos de mensajes como RS, RA, NS, NA y R, lo cual nos permite implementar muchas más funcionalidades que ARP.

5. c)

Algunas funcionalidades de NDP son:

- Reconocimiento del router: todos los routers en una red transmiten RA a todos los usuarios de la red a través de multicast. Estos contienen información como la dirección, el prefijo de red y el enrutamiento. Sobre la base de estos registros los clientes determinan la puerta de enlace determinada y la máscara de subred. Los anuncios del router también se pueden forzar con ayuda de las RS.
- Determinar el siguiente salto conveniente : Si se tiene que enviar un paquete, el protocolo NDP se encarga de comprobar si la caché de destino ya contiene una entrada para el host de destino previsto. Si no es así, el protocolo determina cuál es el próximo nodo intermedio. En caso de que no haya un equivalente en el neighbor cache, se genera automáticamente y se inicia la resolución de la dirección.
- Resolver direcciones IP en direcciones MAC : para determinar la dirección MAC de un host, este recibe una NS a través de IPv6 multicast en su propia dirección de multidifusión. A esta única combinación de direcciones solo puede responder el host. Como respuesta, este envía un mensaje de NA con su dirección MAC.
- Reconocer la falta de disponibilidad de un vecino: En caso de que no se hayan intercambiado datos con el dispositivo en un tiempo prolongado, para comprobar si el

host en cuestión no está disponible, se envía primero un paquete de datos a la dirección registrada. Si no hay respuesta se realiza un último test a través de una NS Unicast. Si éste confirma la inaccesibilidad, esta entrada desaparecerá del neighbor cache.

- Detectar direcciones duplicadas: si un dispositivo ha obtenido una dirección por autoconfiguración, el NDP considera a esta dirección en un primer momento como “tentative” (provisional). Antes de activarse definitivamente debe clasificarse como “inequívoca”. El cliente de red recién conectado, envía una NS a la dirección temporal que quiere usar él mismo, junto con una dirección temporal de retorno no específica. Si otro host ya utiliza dicha dirección, este responderá con un mensaje de NA. El cliente que realiza la comprobación también lo recibe, después de lo que obtiene una nueva dirección.

5. d)

No, no existen las direcciones en broadcast en IPV6, NDP implementa las direcciones multicast, las cuales se caracterizan por enviarse desde una interfaz de salida, a un grupo de interfaces de destino simultáneamente.

Multicast es similar al broadcast sólo que en multicast se envía a un grupo específico, mientras que broadcast envía a todos los nodos de red. Por eso la multicast en IPV6 es más eficiente, ya que no inunda la red de paquetes como el broadcast de IPV4.

5. e)

A continuación desarrollamos las diferencias entre los tipos de direcciones en IPV6:

Dirección global: Equivalente a las direcciones IPV4 públicas. Es una dirección globalmente única, es la que permite a los dispositivos comunicarse con otras redes. Son usadas para conectarse globalmente de un sitio a otro, es decir, es usada por dispositivos para la comunicación uno a uno a través de Internet IPV6.

Las direcciones unicast global comienzan con [2000 : : /3].

Dirección Link-local: Es una dirección que se limita a un único enlace local o subred, es decir, cada subred tiene una dirección link-local exclusiva, lo que implica que los routers no pueden reenviar paquetes de origen o destino link-local. Esta se asigna automáticamente a la interfaz. Se usan para lo siguiente:

- Los hosts utilizan la dirección link-local del router local para obtener la dirección IPV6 de gateway predeterminado.
- Los routers intercambian mensajes de protocolo de enrutamiento dinámico mediante direcciones link-local.

- Las tablas de enrutamiento de los routers utilizan la dirección link-local para identificar el router del siguiente salto al reenviar paquetes IPv6.
- Son usadas para la comunicación uno a uno dentro de los límites del router.

Las direcciones unicast link-local comienzan con [FE80 : : /10].

Dirección Unique-local: Es una dirección que se utiliza para el direccionamiento local dentro de un sitio o un grupo de sitios, no son enrutables en la IPv6 global, tiene similitud con las direcciones IPv4 privadas. Se usan dentro de dispositivos corporativos o conjunto de redes, es decir están diseñadas para ser usadas de manera local, no en comunicaciones fuera de los límites de la red corporativa. Las direcciones unicast unique-local comienzan con [FC00 : : /7].

Parte 2:

Introducción

Creamos la topología requerida con ayuda del script que nos brindaron, agregando un host, la configuración IPv4, etc.

El script utilizado fue;

[Script LinuxNamespaces - PEREZ / LAMBERTI](#)

Desarrollo

2) Para realizar esta configuración, se escribió el archivo dhcpd.conf con las siguientes líneas;

```
tee -a /etc/dhcp/dhcpd.conf <<EOF
subred 192.168.2.0 netmask 255.255.255.0 {
    range dynamic-bootp 192.168.2.10 192.168.2.30;
    option broadcast-address 192.168.2.31;
    option routers 192.168.2.12;
}
EOF
```

Donde asignamos una “pool” de IP desde 10 a 30 en la subred 2, y según lo requerido por la consigna, agregamos como opcional la IP del router 192.168.2.12 dentro del rango del pool para que el servidor no la entregue.

También colocamos la dirección de broadcast.

3) Corriendo los host h2 y h3, al solicitar una IP al DHCP nos brinda;

H2: `ip netns exec h2 dhclient -d`

```
Listening on LPF/vpeer2/1a:cd:ee:1b:39:38
Sending on   LPF/vpeer2/1a:cd:ee:1b:39:38
Sending on   Socket/fallback
DHCPDISCOVER on vpeer2 to 255.255.255.255 port 67 interval 3 (xid=0x2edbc3d)
DHCPREQUEST of 192.168.2.27 on vpeer2 to 255.255.255.255 port 67 (xid=0x3dcdb2e)
)
DHCPOFFER of 192.168.2.27 from 192.168.2.9
DHCPACK of 192.168.2.27 from 192.168.2.9
bound to 192.168.2.27 -- renewal in 1569 seconds.
```


H3: *ip netns exec h3 dhclient -d*

```
Listening on LPF/vpeer3/ee:19:91:e7:f7:15
Sending on   LPF/vpeer3/ee:19:91:e7:f7:15
Sending on   Socket/fallback
DHCPDISCOVER on vpeer3 to 255.255.255.255 port 67 interval 3 (xid=0x27174419)
DHCPREQUEST of 192.168.2.28 on vpeer3 to 255.255.255.255 port 67 (xid=0x19441727)
DHCP OFFER of 192.168.2.28 from 192.168.2.9
DHCPACK of 192.168.2.28 from 192.168.2.9
bound to 192.168.2.28 -- renewal in 1402 seconds.
```

4) Corriendo el comando

Tcpdump -i veth-dhcp

Escuchamos la interfaz veth-dhcp, correspondiente al servidor DHCP.

Luego, procedemos a solicitar una dirección IP desde el host h2.

```
root@parallels-Parallels-Virtual-Platform:/home/parallels# tcpdump -i veth-dhcp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on veth-dhcp, link-type EN10MB (Ethernet), capture size 262144 bytes
01:53:13.836787 IP 0.0.0.0.bootpc > 255.255.255.255.bootps: BOOTP/DHCP, Request from 1a:cd:ee:1b:39:38 (oui Unknown), length 303
01:53:13.841044 IP 192.168.2.9.bootps > 192.168.2.27.bootpc: BOOTP/DHCP, Reply, length 330
01:53:18.864358 ARP, Request who-has 192.168.2.27 tell 192.168.2.9, length 28
01:53:18.864426 ARP, Reply 192.168.2.27 is-at 1a:cd:ee:1b:39:38 (oui Unknown), length 28
```

Podemos observar 4 paquetes:

1. DHCP Discovery: Es una solicitud DHCP realizada por un cliente de este protocolo para que el servidor DHCP de dicha red de computadoras le asigne una dirección IP.
2. DHCP Offer: Es el paquete de respuesta del Servidor DHCP a un cliente DHCP ante su petición de la asignación de IP mediante su MAC Address.
3. DHCP Request: El cliente selecciona la configuración de los paquetes recibidos de *DHCP Offer*. Una vez más, el cliente solicita la dirección IP específica que indicó el servidor.
4. DHCP Acknowledge: Este paquete incluye la duración de la conexión y cualquier otra información de configuración que el cliente pueda tener solicitada. En este punto, el proceso de configuración TCP/IP se ha completado.

5) En primera instancia, no pudimos hacer PING entre el host h1 y los demás, pero luego configuramos el default gateway del mismo y se soluciono. En IPv6 no requerimos de esta configuración previa gracias al protocolo NPD.

Para configurar el Default GW:

```
ip netns exec h1 ip route add default via 192.168.1.11
```

Luego,

```
root@parallels-Parallels-Virtual-Platform:/home/parallels# sudo ip netns exec h2
ping 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=0.162 ms
64 bytes from 192.168.1.10: icmp_seq=2 ttl=63 time=0.084 ms
64 bytes from 192.168.1.10: icmp_seq=3 ttl=63 time=0.052 ms
```

Anexo I

Create config files

```
tee -a /etc/radvd.conf <<EOF
```

```
interface vpeer-router {
```

```
    AdvSendAdvert on;
```

```
    MinRtrAdvInterval 3;
```

```
    MaxRtrAdvInterval 10;
```

```
    prefix 2001::/64 {
```

```
        AdvOnLink on;
```

```
        AdvAutonomous on;
```

```
        AdvRouterAddr on;
```

```
    };
```

```
};
```

```

interface veth3 {

    AdvSendAdvert on;

    MinRtrAdvInterval 3;

    MaxRtrAdvInterval 10;

    prefix 2002::/64 {

        AdvOnLink on;

        AdvAutonomous on;

        AdvRouterAddr on;

    };

};

EOF

#DHCP

tee -a /etc/dhcp/dhcpd.conf <<EOF
subred 192.168.2.0 netmask 255.255.255.0 {
    range dynamic-bootp 192.168.2.10 192.168.2.30;
    option broadcast-address 192.168.2.31;
    option routers 192.168.2.12;
}

EOF
# Create resources


ip netns add h1

ip netns add h2

ip netns add h3

ip netns add dhcp

```



```
ip netns add r1
```

```
ip link add name veth1 type veth peer name vpeer1
```

```
ip link add name veth2 type veth peer name vpeer2
```

```
ip link add name veth3 type veth peer name vpeer3
```

```
ip link add name veth-dhcp type veth peer name vpeer-dhcp
```

```
ip link add name veth-router type veth peer name vpeer-router
```

```
brctl addbr sw1
```

```
# Set peer link up
```

```
ip link set veth1 up
```

```
ip link set veth2 up
```

```
ip link set veth3 up
```

```
ip link set veth-dhcp up
```

```
ip link set veth-router up
```

```
ip link set sw1 up
```

```
# Assign interfaces to namespaces
```

```
ip link set dev vpeer1 netns h1
```

```
ip link set dev vpeer2 netns h2
```

```
ip link set dev vpeer3 netns h3
```

```
ip link set dev vpeer-dhcp netns dhcp
```



```
ip link set dev vpeer-router netns r1
```

```
ip link set dev veth1 netns r1
```

```
# Connect veth to bridge
```

```
brctl addif sw1 veth2
```

```
brctl addif sw1 veth3
```

```
brctl addif sw1 veth-dhcp
```

```
brctl addif sw1 veth-router
```

```
# Configure router as router
```

```
ip netns exec r1 sysctl -w net.ipv4.conf.all.forwarding=1
```

```
# Configure IP addresses
```

```
ip netns exec r1 ip addr add 192.168.2.12/24 dev vpeer-router
```


```
ip netns exec r1 ip addr add 192.168.1.11/24 dev veth1
```

```
ip netns exec h1 ip addr add 192.168.1.10/24 dev vpeer1
```

```
ip netns exec h1 ip route add default via 192.168.1.11
```

```
ip netns exec dhcp ip addr add 192.168.2.9/24 dev vpeer-dhcp
```

```
# Set Up interfaces
```



```
ip netns exec h1 ip link set lo up
```

```
ip netns exec h2 ip link set lo up
```

```
ip netns exec h3 ip link set lo up
```

```
ip netns exec dhcp ip link set lo up
```

```
ip netns exec r1 ip link set lo up
```

```
ip netns exec h1 ip link set vpeer1 up
```

```
ip netns exec h2 ip link set vpeer2 up
```

```
ip netns exec h3 ip link set vpeer3 up
```

```
ip netns exec dhcp ip link set vpeer-dhcp up
```

```
ip netns exec r1 ip link set veth1 up
```

```
ip netns exec r1 ip link set vpeer-router up
```

```
sudo ip netns exec dhcp dnsmasq --dhcp-range=192.168.2.10,192.168.2.30,255.255.255.0  
--interface=vpeer-dhcp --no-daemon --dhcp-option=option:router,192.168.2.12
```