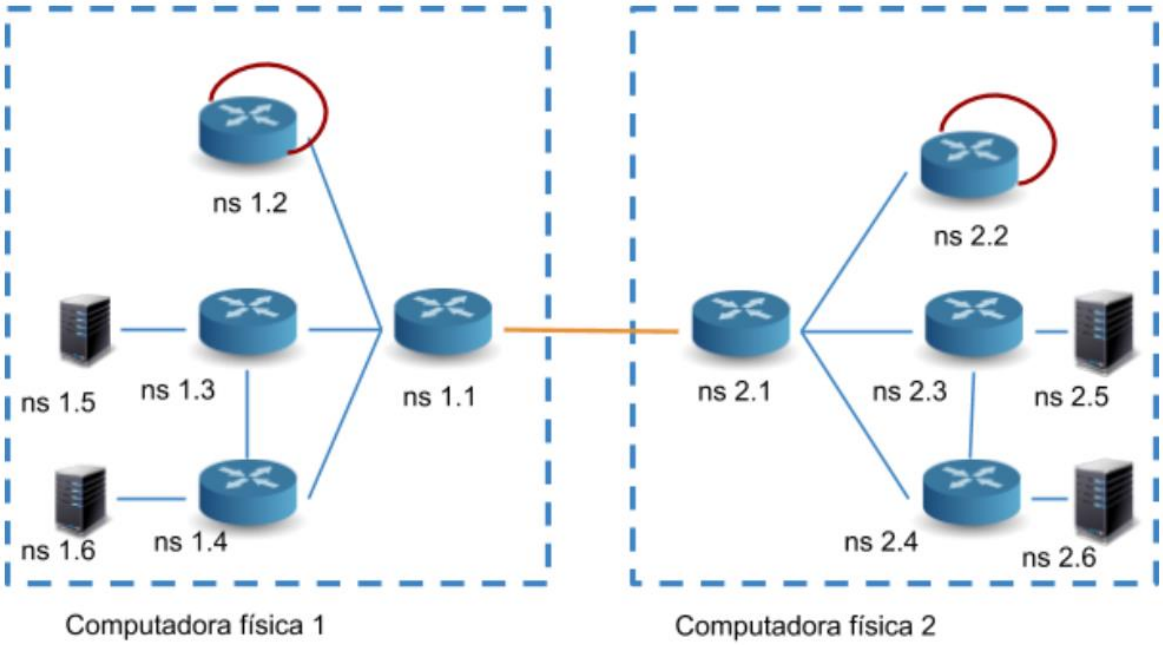
UNIVERSIDAD NACIONAL DE CÓRDOBA
Facultad de Ciencias Exactas, Físicas y Naturales

Redes de Computadoras

Práctico 3: Ruteo estático. Fragmentación

Contrera, Ivan
Malano, Leandro



1 y 8. Se utilizaran los siguientes comandos para crear los namespaces en cada máquina física (en modo root), mostrando todos los namespaces necesarios para configurar routers y hosts de ambas maquinas fisicas:

```
#agrega namespaces
ip netns add ns1.1
ip netns add ns1.2
ip netns add ns1.3
ip netns add ns1.4
ip netns add ns1.5
ip netns add ns1.6
```

```
#agrega namespaces
ip netns add ns2.1
ip netns add ns2.2
ip netns add ns2.3
ip netns add ns2.4
ip netns add ns2.5
ip netns add ns2.6
```

2. A continuacion se creará un bridge por cada maquina fisica, para comunicarse entre namespaces de estas dos. Antes, hay que instalar el paquete *bridge-utils* que contiene la herramienta *brctl* para poder crearlas:

```
#agrega el bridge que conectara con la interfaz fisica
brctl addbr bridge-1
```

```
#agrega el bridge que conectara con la interfaz fisica
brctl addbr bridge-2
```

3 y 4) Para conectar un bridge con, por ejemplo, el namespace ns1.1, se crea un cable virtual asignando dos nombres de interfaces, utilizando el comando *ip link add int11-4 type veth peer name brdg-1*, cuyos nombres marcados en rojo van a pertenecer a ns1.1 y a bridge-1,

respectivamente. Luego se asocia la interfaz correspondiente al bridge, y a su vez tambien la fisica (enp3s0 por ejemplo), cuyos comandos se muestran en las siguientes capturas:

PC1:

```
#se desactiva la interfaz fisica y se asocia con el bridge, y se asocia la interfaz virtual brdg-2
#ifconfig enp3s0 down
brctl addif bridge-1 enp0s3
brctl addif bridge-1 brdg-1
```

PC2:

```
#se desactiva la interfaz fisica y se asocia con el bridge, y se asocia la interfaz virtual brdg-2
#ifconfig enp3s0 down
brctl addif bridge-2 enp3s0
brctl addif bridge-2 brdg-2
```

5. Como se crearon los cables virtuales para unir un namespace con un bridge, se crean los cables restantes para unir el resto de los mismos, según se muestra en la topología del comienzo del informe. Luego se asocian con el comando *ip link set int15-1 netns ns1.5* (ejemplo para asociar una interfaz), y posteriormente se asignan las direcciones ip para cada una, como se ve a continuacion:

PC1:

```
#agrega las direcciones ip a cada interfaz de cada namespace
ip netns exec ns1.1 ip address add 2001:AAAA:BBBB:5::2/64 dev int11-1
ip netns exec ns1.1 ip address add 2001:AAAA:BBBB:2::2/64 dev int11-2
ip netns exec ns1.1 ip address add 2001:AAAA:BBBB:6::1/64 dev int11-3
ip netns exec ns1.1 ip address add 2001:AAAA:DDDD:1::1/64 dev int11-4
ip netns exec ns1.2 ip address add 2001:AAAA:BBBB:6::2/64 dev int12-1
ip netns exec ns1.3 ip address add 2001:AAAA:BBBB:1::1/64 dev int13-1
ip netns exec ns1.3 ip address add 2001:AAAA:BBBB:4::1/64 dev int13-2
ip netns exec ns1.3 ip address add 2001:AAAA:BBBB:2::1/64 dev int13-3
ip netns exec ns1.4 ip address add 2001:AAAA:BBBB:3::1/64 dev int14-1
ip netns exec ns1.4 ip address add 2001:AAAA:BBBB:4::2/64 dev int14-2
ip netns exec ns1.4 ip address add 2001:AAAA:BBBB:5::1/64 dev int14-3
ip netns exec ns1.5 ip address add 2001:AAAA:BBBB:1::10/64 dev int15-1
ip netns exec ns1.6 ip address add 2001:AAAA:BBBB:3::20/64 dev int16-1
```

PC2:

```
#agrega las direcciones ip a cada interfaz de cada namespace
ip netns exec ns2.1 ip address add 2001:AAAA:CCCC:5::2/64 dev int21-1
ip netns exec ns2.1 ip address add 2001:AAAA:CCCC:2::2/64 dev int21-2
ip netns exec ns2.1 ip address add 2001:AAAA:CCCC:6::1/64 dev int21-3
ip netns exec ns2.1 ip address add 2001:AAAA:DDDD:1::2/64 dev int21-4
ip netns exec ns2.2 ip address add 2001:AAAA:CCCC:6::2/64 dev int22-1
ip netns exec ns2.3 ip address add 2001:AAAA:CCCC:1::1/64 dev int23-1
ip netns exec ns2.3 ip address add 2001:AAAA:CCCC:4::1/64 dev int23-2
ip netns exec ns2.3 ip address add 2001:AAAA:CCCC:2::1/64 dev int23-3
ip netns exec ns2.4 ip address add 2001:AAAA:CCCC:3::1/64 dev int24-1
ip netns exec ns2.4 ip address add 2001:AAAA:CCCC:4::2/64 dev int24-2
ip netns exec ns2.4 ip address add 2001:AAAA:CCCC:5::1/64 dev int24-3
ip netns exec ns2.5 ip address add 2001:AAAA:CCCC:1::10/64 dev int25-1
ip netns exec ns2.6 ip address add 2001:AAAA:CCCC:3::20/64 dev int26-1
```

6. Una vez configurados correctamente los bridges y namespaces ns1.1 y ns2.1, se hace una prueba de conectividad, obteniendo un resultado positivo:

```
root@leandro-Lenovo-B590:/home/leandro/scripts# ip netns exec ns2.1 ping6 2001:aaaa:dddd:1::1
PING 2001:aaaa:dddd:1::1(2001:aaaa:dddd:1::1) 56 data bytes
64 bytes from 2001:aaaa:dddd:1::1: icmp_seq=1 ttl=64 time=0.477 ms
64 bytes from 2001:aaaa:dddd:1::1: icmp_seq=2 ttl=64 time=0.293 ms
64 bytes from 2001:aaaa:dddd:1::1: icmp_seq=3 ttl=64 time=0.316 ms
64 bytes from 2001:aaaa:dddd:1::1: icmp_seq=4 ttl=64 time=0.283 ms
64 bytes from 2001:aaaa:dddd:1::1: icmp_seq=5 ttl=64 time=0.282 ms
64 bytes from 2001:aaaa:dddd:1::1: icmp_seq=6 ttl=64 time=0.277 ms
^C
--- 2001:aaaa:dddd:1::1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 4998ms
rtt min/avg/max/mdev = 0.277/0.321/0.477/0.072 ms
root@leandro-Lenovo-B590:/home/leandro/scripts#
```

7. Al configurar el MTU = 500 utilizando el comando *ip netns exec ns1.1 ip link int11-4 MTU 500* (para establecer el nuevo mtu en la interfaz mencionada), se genero conflicto ya que se borraba su direccion ip, e impedia la comunicacion con la otra pc. Segun el rfc 2460, seccion 5, no se pueden configurar MTU inferiores a 1280, por lo que se configuró con ese valor, utilizando el mismo comando mencionado antes.

9. Se configura el enrutamiento estatico en ambas pc, tal cual se ve en las capturas:

PC1:

```
#se agregan rutas a los distintos enrutadores
ip netns exec ns1.1 ip -6 route add 2001:AAAA:BBBB:1::0/64 via 2001:AAAA:BBBB:2::1
ip netns exec ns1.1 ip -6 route add 2001:AAAA:BBBB:3::0/64 via 2001:AAAA:BBBB:5::1
ip netns exec ns1.1 ip -6 route add 2001:AAAA:BBBB:4::0/64 via 2001:AAAA:BBBB:5::1
ip netns exec ns1.1 ip -6 route add 2001:AAAA::0/8 via 2001:AAAA:DDDD:1::2
ip netns exec ns1.2 ip -6 route add 2001:AAAA:BBBB:1::0/64 via 2001:AAAA:BBBB:6::1
ip netns exec ns1.2 ip -6 route add 2001:AAAA:BBBB:2::0/64 via 2001:AAAA:BBBB:6::1
ip netns exec ns1.2 ip -6 route add 2001:AAAA:BBBB:3::0/64 via 2001:AAAA:BBBB:6::1
ip netns exec ns1.2 ip -6 route add 2001:AAAA:BBBB:4::0/64 via 2001:AAAA:BBBB:6::1
ip netns exec ns1.2 ip -6 route add 2001:AAAA:BBBB:5::0/64 via 2001:AAAA:BBBB:6::1
ip netns exec ns1.2 ip -6 route add 2001:AAAA::0/8 via 2001:AAAA:BBBB:6::1
ip netns exec ns1.3 ip -6 route add 2001:AAAA:BBBB:3::0/64 via 2001:AAAA:BBBB:4::2
ip netns exec ns1.3 ip -6 route add 2001:AAAA:BBBB:5::0/64 via 2001:AAAA:BBBB:4::2
ip netns exec ns1.3 ip -6 route add 2001:AAAA:BBBB:6::0/64 via 2001:AAAA:BBBB:2::2
ip netns exec ns1.3 ip -6 route add 2001:AAAA::0/8 via 2001:AAAA:BBBB:2::2
ip netns exec ns1.4 ip -6 route add 2001:AAAA:BBBB:1::0/64 via 2001:AAAA:BBBB:4::1
ip netns exec ns1.4 ip -6 route add 2001:AAAA:BBBB:2::0/64 via 2001:AAAA:BBBB:4::1
ip netns exec ns1.4 ip -6 route add 2001:AAAA:BBBB:6::0/64 via 2001:AAAA:BBBB:5::2
ip netns exec ns1.4 ip -6 route add 2001:AAAA::0/8 via 2001:AAAA:BBBB:5::2
```

PC2:

```
#se agregan rutas a los distintos enrutadores
ip netns exec ns2.1 ip -6 route add 2001:AAAA:CCCC:1::0/64 via 2001:AAAA:CCCC:2::1
ip netns exec ns2.1 ip -6 route add 2001:AAAA:CCCC:3::0/64 via 2001:AAAA:CCCC:5::1
ip netns exec ns2.1 ip -6 route add 2001:AAAA:CCCC:4::0/64 via 2001:AAAA:CCCC:5::1
ip netns exec ns2.1 ip -6 route add 2001:AAAA::0/8 via 2001:AAAA:DDDD:1::1
ip netns exec ns2.2 ip -6 route add 2001:AAAA:CCCC:1::0/64 via 2001:AAAA:CCCC:6::1
ip netns exec ns2.2 ip -6 route add 2001:AAAA:CCCC:2::0/64 via 2001:AAAA:CCCC:6::1
ip netns exec ns2.2 ip -6 route add 2001:AAAA:CCCC:3::0/64 via 2001:AAAA:CCCC:6::1
ip netns exec ns2.2 ip -6 route add 2001:AAAA:CCCC:4::0/64 via 2001:AAAA:CCCC:6::1
ip netns exec ns2.2 ip -6 route add 2001:AAAA:CCCC:5::0/64 via 2001:AAAA:CCCC:6::1
ip netns exec ns2.2 ip -6 route add 2001:AAAA::0/8 via 2001:AAAA:CCCC:6::1
ip netns exec ns2.3 ip -6 route add 2001:AAAA:CCCC:3::0/64 via 2001:AAAA:CCCC:4::2
ip netns exec ns2.3 ip -6 route add 2001:AAAA:CCCC:5::0/64 via 2001:AAAA:CCCC:4::2
ip netns exec ns2.3 ip -6 route add 2001:AAAA:CCCC:6::0/64 via 2001:AAAA:CCCC:2::2
ip netns exec ns2.3 ip -6 route add 2001:AAAA::0/8 via 2001:AAAA:CCCC:2::2
ip netns exec ns2.4 ip -6 route add 2001:AAAA:CCCC:1::0/64 via 2001:AAAA:CCCC:4::1
ip netns exec ns2.4 ip -6 route add 2001:AAAA:CCCC:2::0/64 via 2001:AAAA:CCCC:4::1
ip netns exec ns2.4 ip -6 route add 2001:AAAA:CCCC:6::0/64 via 2001:AAAA:CCCC:5::2
ip netns exec ns2.4 ip -6 route add 2001:AAAA::0/8 via 2001:AAAA:CCCC:5::2
```

Como modo de ejemplo, se hace una prueba de conectividad entre:

ns1.5 y ns2.5)

```
root@ivan-ubuntu:/home/ivanovic/scripts# ip netns exec ns1.5 ping6 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=60 time=0.424 ms
64 bytes from 2001:aaaa:cccc:1::10: icmp_seq=2 ttl=60 time=0.257 ms
64 bytes from 2001:aaaa:cccc:1::10: icmp_seq=3 ttl=60 time=0.375 ms
```

ns1.6 y ns2.6)

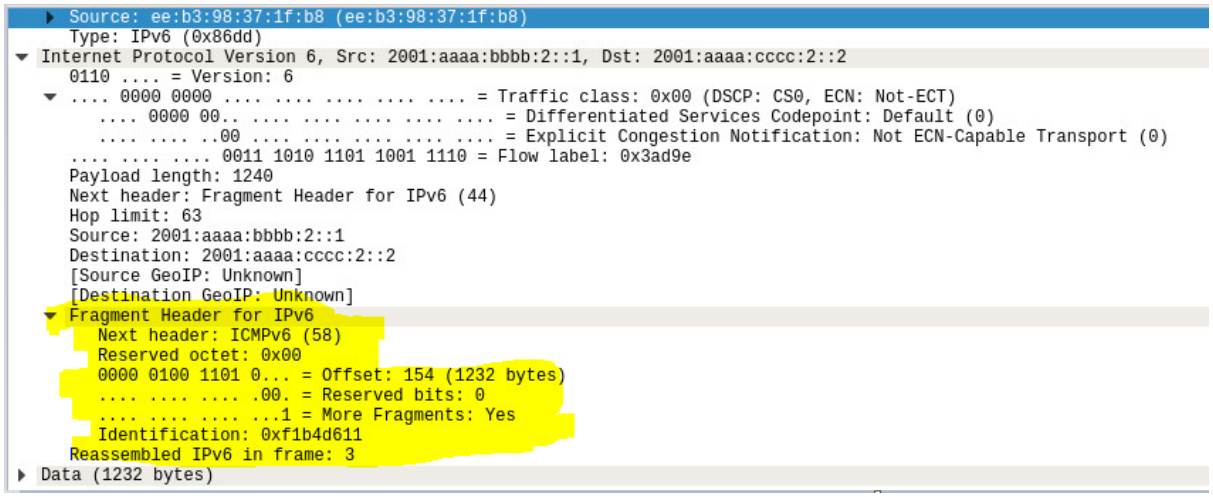
```
root@ivan-ubuntu:/home/ivanovic/scripts# ip netns exec ns1.6 ping6 2001:aaaa:cccc:3::20
PING 2001:aaaa:cccc:3::20(2001:aaaa:cccc:3::20) 56 data bytes
64 bytes from 2001:aaaa:cccc:3::20: icmp_seq=1 ttl=60 time=0.458 ms
64 bytes from 2001:aaaa:cccc:3::20: icmp_seq=2 ttl=60 time=0.384 ms
64 bytes from 2001:aaaa:cccc:3::20: icmp_seq=3 ttl=60 time=0.361 ms
```

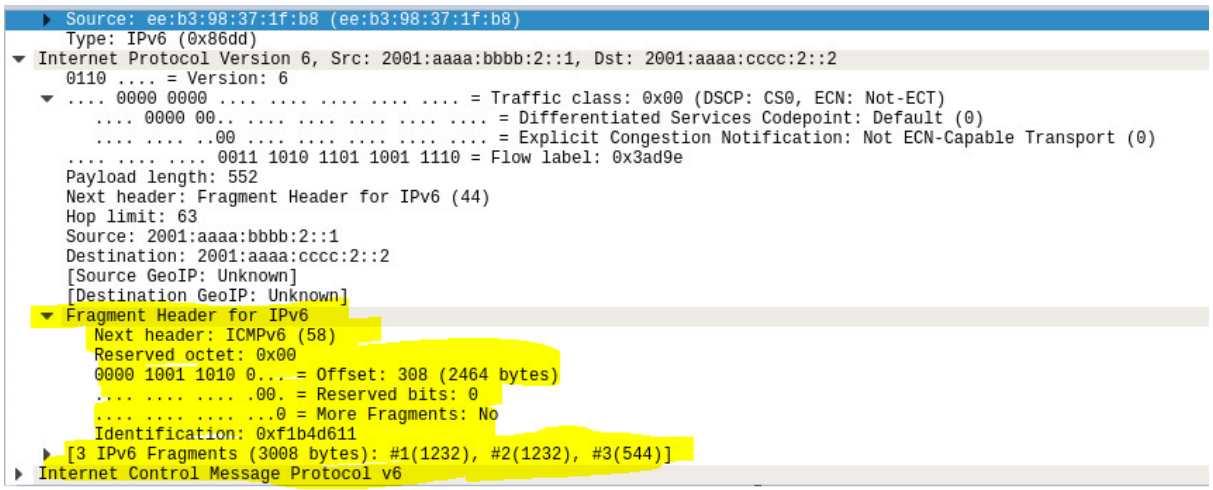
10. Para detectar la fragmentación, se hace una prueba de conectividad enviando paquetes ICMPv6 de 3000 bytes (utilizando el comando *ip netns exec ns1.3 ping6 2001:aaaa:cccc:2::2 -s 3000*) desde la PC1 a la PC2, donde en esta ultima se hacen capturas de trafico con wireshark. Como la MTU de las interfaces int11-4 (ns1.1) e int21-4 (ns2.4) es de 1280, y la del resto es de 1500 (por defecto), se produce la fragmentacion de los datagramas, en este caso, en 3 fragmentos como se ve en la siguiente captura:

lo.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	2001:aaaa:bbbb:2::1	2001:aaaa:cccc:2::2	IPv6	1294	IPv6 fragment (off=0 more=y ident=0xf1b4d611 nxt=58)
2	0.000021351	2001:aaaa:bbbb:2::1	2001:aaaa:cccc:2::2	IPv6	1294	IPv6 fragment (off=1232 more=y ident=0xf1b4d611 nxt=58)
3	0.000024742	2001:aaaa:bbbb:2::1	2001:aaaa:cccc:2::2	ICMPv6	606	Echo (ping) request id=0x1048, seq=1, hop limit=63 (reply in 6)

En las capturas siguientes se muestran en detalle cada uno de los fragmentos anteriores, remarcando la cabecera de fragmentacion:

```
▸ Source: ee:b3:98:37:1f:b8 (ee:b3:98:37:1f:b8)
  Type: IPv6 (0x86dd)
▾ Internet Protocol Version 6, Src: 2001:aaaa:bbbb:2::1, Dst: 2001:aaaa:cccc:2::2
  0110 .... = Version: 6
▾ .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... 0000 00.. .... .... .... .... .... = Differentiated Services Codepoint: Default (0)
    .... .... ..00 .... .... .... .... .... = Explicit Congestion Notification: Not ECN-Capable Transport (0)
  .... .... .... 0011 1010 1101 1001 1110 = Flow label: 0x3ad9e
  Payload length: 1240
  Next header: Fragment Header for IPv6 (44)
  Hop limit: 63
  Source: 2001:aaaa:bbbb:2::1
  Destination: 2001:aaaa:cccc:2::2
  [Source GeoIP: Unknown]
  [Destination GeoIP: Unknown]
▾ Fragment Header for IPv6
    Next header: ICMPv6 (58)
    Reserved octet: 0x00
    0000 0000 0000 0... = Offset: 0 (0 bytes)
    .... .... .... .00. = Reserved bits: 0
    .... .... .... ...1 = More Fragments: Yes
    Identification: 0xf1b4d611
  Reassembled IPv6 in frame: 3
▸ Data (1232 bytes)
```





donde, entre otras cosas, se puede observar el campo offset, que indica la porción de datos enviado en este paquete en relación al paquete original. El fragment offset (13 bit en el IP header) está indicado en bloques de 64 bits. Todos los fragmentos menos el último tienen el *more fragments* flag con valor "1".

11. La fragmentación es el mecanismo que permite separar un paquete ip en varios pedazos. En ipv4 la fragmentación puede darse en el host inicial o un router que está entre el emisor y receptor. Cuando el paquete es demasiado grande para el enlace donde será enviado, puede ser fragmentado por el remitente.

En IPv6 ya no se permite a los routers fragmentar los paquetes. El emisor siempre está informado con un mensaje ICMP cuando una fragmentación es necesaria. Así, el emisor puede bajar su tamaño de paquete para esta conexión y la fragmentación ya no es necesaria. En caso de requerirse una fragmentación, el host, es quien debe hacerla. El reensamblado se realiza siempre en el nodo de destino, usando como informacion el campo offset para que los fragmentos se ordenen correctamente.

REFERENCIAS:

http://www.tutorialspoint.com/unix_commands/ping6.htm
https://tools.ietf.org/html/rfc2460#section-5
https://es.wikipedia.org/wiki/Fragmentaci%C3%B3n_IP
https://www.ibm.com/support/knowledgecenter/es/ssw_ibm_i_73/rzai2/rzai2compipv4ipv6.htm#rzai2compipv4ipv6__compfragments
https://sites.google.com/site/tknikaipv6/2-direccionamiento/2-1-la-nueva-cabecera/2-1-3-las-cabeceras-de-extension
https://www.youtube.com/watch?v=_WgUwUf1d34
http://man7.org/linux/man-pages/man4/veth.4.html
https://wiki.debian.org/BridgeNetworkConnections
https://blog.scottlowe.org/2013/09/04/introducing-linux-network-namespaces/
https://es.wikipedia.org/wiki/Cabecera_IP