RSE 2021-2022 / Alejandro Albert Casañ

Memoria práctica 1

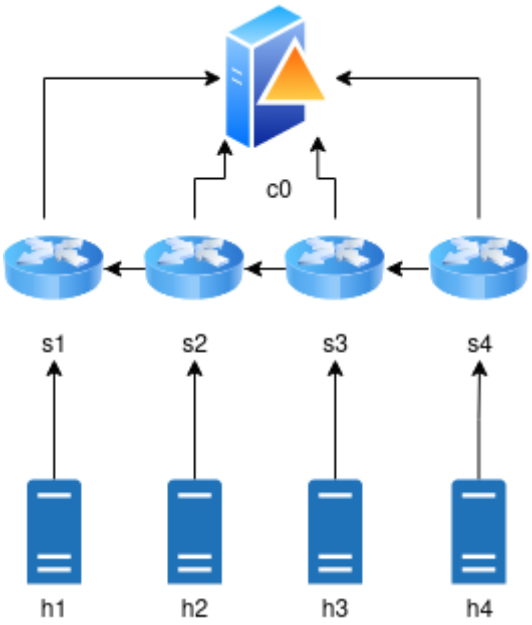
1.- Prueba a ejecutar el comando route en los diferentes dispositivos. Este comando se utiliza cuando se quiere trabajar con la tabla de enrutamiento IP/kernel. Se utiliza principalmente para configurar rutas estáticas a hosts o redes específicas a través de una interfaz. Se utiliza para mostrar o actualizar la tabla de enrutamiento IP/kernel.

Ejecutamos el comando especificado en cada uno de los nodos y el resultado es el siguiente:

```
mininet> c0 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         172.17.0.1      0.0.0.0         UG    0      0        0 eth0
172.17.0.0      0.0.0.0         255.255.0.0     U     0      0        0 eth0
mininet> h1 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.0.0.0       U     0      0        0 h1-eth0
mininet> h2 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.0.0.0       U     0      0        0 h2-eth0
mininet> s1 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         172.17.0.1      0.0.0.0         UG    0      0        0 eth0
172.17.0.0      0.0.0.0         255.255.0.0     U     0      0        0 eth0
```

2.- Dibuja la estructura de red que obtienes con este último.

La estructura de red resultante es la siguiente:



3.- Utilizando ping calcula la diferencia con el caso sin definir los "link parameters"

Sin los link parameters los pings tardan lo que podemos ver a continuación

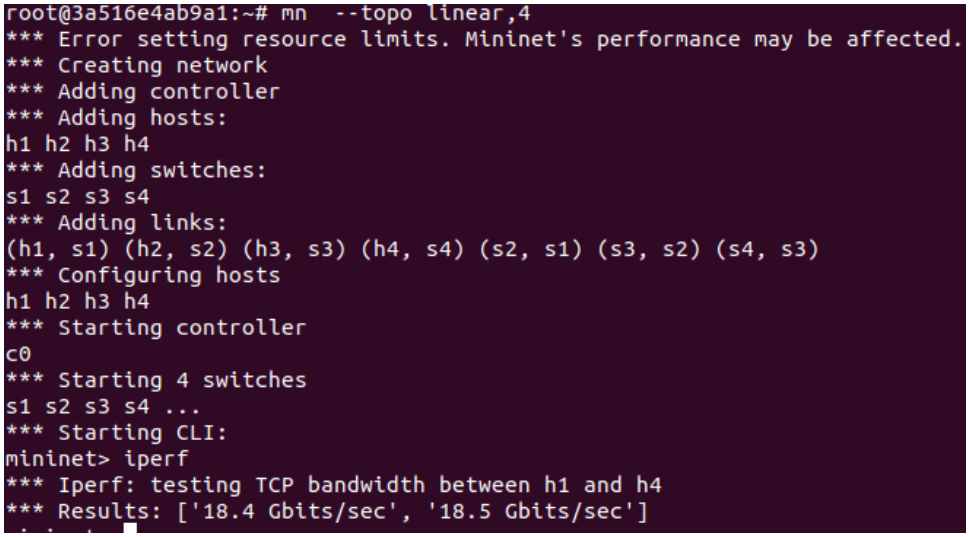
```
root@3a516e4ab9a1:~# mn  --topo linear,4
*** Error setting resource limits. Mininet's performance may be affected.
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (s2, s1) (s3, s2) (s4, s3)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Starting CLI:
mininet> h1 ping -c 1 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=9.33 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 9.338/9.338/9.338/0.000 ms
```

Mientras que con los link parameters obtenemos lo siguiente

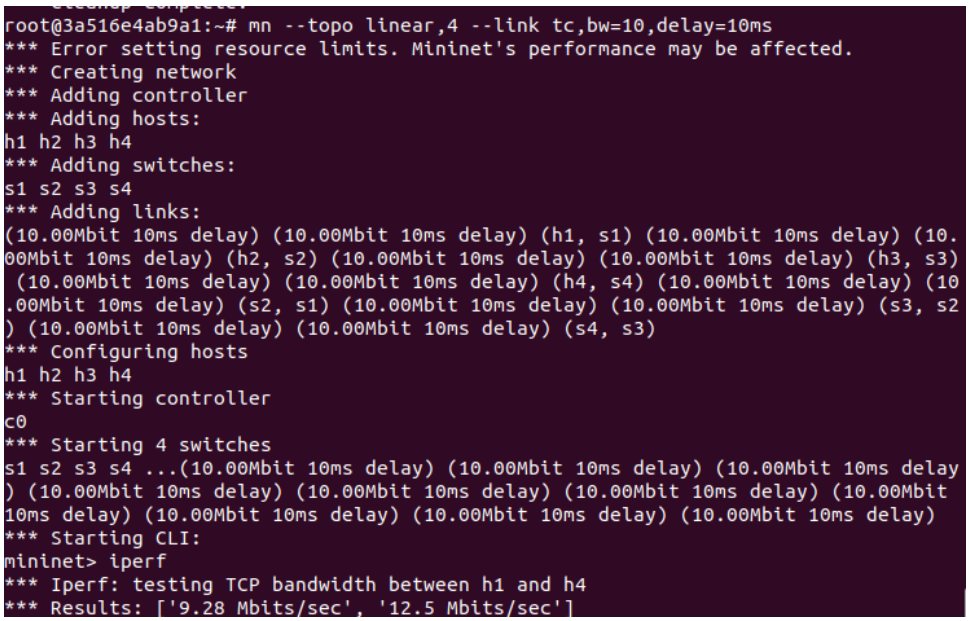
```
root@3a516e4ab9a1:~# mn --topo linear,4 --link tc,bw=10,delay=10ms
*** Error setting resource limits. Mininet's performance may be affected.
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(10.00Mbit 10ms delay) (10.00Mbit 10ms delay) (h1, s1) (10.00Mbit 10ms delay) (10.
00Mbit 10ms delay) (h2, s2) (10.00Mbit 10ms delay) (10.00Mbit 10ms delay) (h3, s3)
 (10.00Mbit 10ms delay) (10.00Mbit 10ms delay) (h4, s4) (10.00Mbit 10ms delay) (10
.00Mbit 10ms delay) (s2, s1) (10.00Mbit 10ms delay) (10.00Mbit 10ms delay) (s3, s2
) (10.00Mbit 10ms delay) (10.00Mbit 10ms delay) (s4, s3)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...(10.00Mbit 10ms delay) (10.00Mbit 10ms delay) (10.00Mbit 10ms delay
) (10.00Mbit 10ms delay) (10.00Mbit 10ms delay) (10.00Mbit 10ms delay) (10.00Mbit
10ms delay) (10.00Mbit 10ms delay) (10.00Mbit 10ms delay) (10.00Mbit 10ms delay)
*** Starting CLI:
mininet> h1 ping -c 1 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=130 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 130.671/130.671/130.671/0.000 ms
```

4.- Ahora repite lo mismo pero utilizando el comando iperf. ¿Como?

Ejecutando iperf en el caso en el que no se introducen los link parameters



Mientras que en el caso con los link parameters



Vemos una clara bajada respecto al caso que no tiene link parameters

5.- Indica la secuencia de órdenes para ejecutar el server HTTP en h1 y el cliente get en h2 en la topologia minima de Mininet.

Una vez inicializado mininet con su topología mínima haremos que h1 ejecute ifconfig para saber su ip

```
mininet> h1 ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.1  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::3453:eeff:fe25:9a2f  prefixlen 64  scopeid 0x20<link
        ether 36:53:ee:25:9a:2f  txqueuelen 1000  (Ethernet)
        RX packets 10  bytes 776 (776.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 11  bytes 866 (866.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Y ahora ejecutamos el HTTP server

```
mininet> h1 python2 -m SimpleHTTPServer 80&
```

Por último, ejecutamos el cliente desde h2

```
mininet> h2 wget -O - 10.0.0.1
--2021-12-21 23:45:47--  http://10.0.0.1/
Connecting to 10.0.0.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 414 [text/html]
Saving to: 'STDOUT'

-                   0%[                    ]       0  --.-KB/s               <!D
OCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"><html>
<title>Directory listing for /</title>
<body>
<h2>Directory listing for /</h2>
<hr>
<ul>
<li><a href=".bashrc">.bashrc</a>
<li><a href=".config/">.config/</a>
<li><a href=".mininet_history">.mininet_history</a>
<li><a href=".profile">.profile</a>
<li><a href=".Xresources">.Xresources</a>
<li><a href="mininet/">mininet/</a>
</ul>
<hr>
</body>
</html>
-                  100%[===================>]     414  --.-KB/s    in 0s

2021-12-21 23:45:47 (72.2 MB/s) - written to stdout [414/414]
```

Y vemos la página que está sirviendo h1

6.- Modifica el código de la red de arriba para que imprima la dirección IP y el MAC de cada host

Para que imprima las MAC de los hosts debemos añadir el siguiente código donde interese que se muestre.

```
info("MAC de los hosts\n")
info("MAC h1: "+h1.MAC()+"\n")
info("MAC h2: "+h2.MAC()+"\n")
info("MAC h3: "+h3.MAC()+"\n")
info("MAC h4: "+h4.MAC()+"\n")
```

7.- Modifica el código de la red de arriba, para que los enlaces sean de 10 Mbps y con un retardo de 10ms. Calcula las prestaciones del enlace entre h1 y h3

En el apartado donde creamos los links añadimos los parámetros necesarios en el método addlink

```
info( '*** Creating links\n' )
net.addLink( h1, s1, bw=10, delay='10ms', max_queue_size=1000, loss=10, use_htb=True)
net.addLink( h2, s1, bw=10, delay='10ms', max_queue_size=1000, loss=10, use_htb=True)
net.addLink( h3, s1, bw=10, delay='10ms', max_queue_size=1000, loss=10, use_htb=True)
net.addLink( h4, s1, bw=10, delay='10ms', max_queue_size=1000, loss=10, use_htb=True)
```

Para comprobar las prestaciones del enlace entre h1 y h3 solo tendríamos que ejecutar iperf h1 h3 con resultado ['49.5 Kbits/sec', '267 Kbits/sec']